REAL WORLD APPLICATIONS

Rajkumar Roy and David Davis, chairs

Designing Crushers with a Multi-Objective Evolutionary Algorithm

L. Barone **L. While** **P. Hingston**

Department of Computer Science & Software Engineering
The University of Western Australia
Nedlands, WA, Australia
luigi@cs.uwa.edu.au;lyndon@cs.uwa.edu.au

School of Computer and Information Science
Edith Cowan University
Mt Lawley, WA, Australia
p.hingston@ecu.edu.au

Abstract

This paper describes the use of a multi-objective evolutionary algorithm to solve an engineering design problem - determining the geometry and operating settings for a crusher in a comminution circuit for ore processing. The outcome is a tool for consulting engineers that can be used to create and explore candidate designs for various scenarios. The tool has proved capable of deriving designs that are clearly superior to existing designs, promising significant financial benefits. The approach is flexible enough to be applied to a variety of similar problems.

1 INTRODUCTION

Evolutionary algorithms are increasingly finding applications in engineering design tasks. In this paper we describe a study, supported by Rio Tinto Ltd, which uses evolutionary algorithms to optimise the performance of a comminution circuit for iron ore processing. In work reported earlier, (Hingston, 2002), a simple evolution strategy algorithm was used to solve this problem. We have restated the details of the problem description here for completeness. In the present study, we report on a further development - a multi-objective algorithm - and compare the two approaches.

The performance of a processing plant has a large impact on the profitability of a mining operation, and yet plant design decisions are often guided more by engineering intuition and previous experience than by analysis. This is because plants are extremely complex to model, so engineers often must rely on simulation tools to evaluate and compare alternative hand-crafted designs. This is a time-consuming process and the lack of an analytical model means that there is little theoretical guidance to narrow the search for better solutions. Evolutionary algorithms can be of great benefit here, providing a means to search large design spaces and present the engineer with superior designs optimised for different operating scenarios.

In order to test the applicability of evolutionary algorithms in this setting, a representative problem was chosen by Rio Tinto. The task was to find combinations of design variables (including geometric shapes and machine settings) to maximise the capacity of a simple comminution circuit, whilst also minimising the size of the product. Earlier work in (Hingston, 2002) showed the effectiveness of a single-objective evolution strategy algorithm for this task. However, the multi-objective approach described in this paper offers clear advantages over the single-objective algorithm.

We begin the paper with a description of the problem, including a brief background on crushers and comminution circuits. Section 3 describes our mapping of the problem to an evolutionary algorithm, including the genetic representation, genetic operators and selection methods. Section 4 presents some illustrative results. Finally, we discuss future enhancements to the system and plans to extend the work to include greater complexity in the simulation model, including circuits.

2 BACKGROUND

Crushing and grinding of rocks and other particles has many important applications, including coarse crushing mined ore and quarry rock, fine grinding of coal for power station boilers, and for production of paint, ceramics, cement and other materials. It has been estimated that several billion tons of material is crushed and ground annually (Hiorns, 1971). Thus optimisation of crushing operations offers large potential economic benefits. For example, in the area of energy savings, Napier-Munn et al ((Napier-Munn, 1996), p1) quote a report of the U.S. National Materials Advisory Board in 1981, which estimated that realistic improvements in crushing-related activities could result in energy savings of more than 20 billion kWh per annum. Other benefits of optimisation of crushing and grinding in mineral processing operations include reduced operating costs, increased throughput and thus value production, and improved downstream performance.

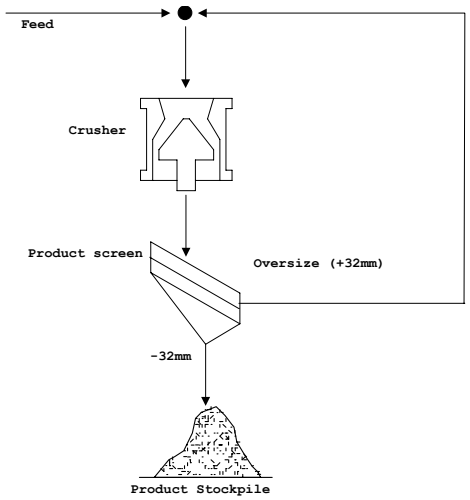


Figure 1 - The simple circuit used in this study

2.1 CRUSHERS AND CIRCUITS

In this section, we provide a brief background on crushers and how they are used in comminution circuits. The interested reader could consult, for example, (Napier-Munn, 1996) for more detailed information. "Comminution" refers to the collection of physical processes that can be applied to a stream of ore to change the size of the particles in the stream. Examples include crushing and grinding (which break ore particles into smaller particles), and screening (which separates ore into several streams of different particle sizes). The purpose of comminution is to transform raw ore into a more usable or more saleable product or to prepare it for further processing. A "comminution circuit" consists of a collection of processing units (crushers, screens etc) connected together (by conveyor belts, for example), possibly containing loops (hence the use of the word "circuit"). One or more streams of ore (the "feed") enter the circuit and one or more streams of transformed material (the "product") exit the circuit.

Figure 1 shows the simple circuit that was used in this study. The feed comes in on a conveyor from the top left and enters the crusher. The crushed ore is then passed through a screen that allows particles less than 32 mm to pass through and report to product. Particles larger than this (the "oversize") are recycled back to the crusher. Thus the input to the crusher is a combination of feed and recirculating oversize.

The type of crusher used here is a "cone" crusher. Figure 2 is a schematic diagram of a typical cone crusher. Material is introduced into the crusher from above, and is crushed as it flows downwards through the machine. The inner crushing surface, or "mantle", is mounted on the conical crushing head and is driven in an eccentric motion swivelling around the axis of the machine. The outer crushing surface, or "bowl", is held stationary. Material flows into the crushing chamber from above, and is crushed between the two surfaces by compressive forces due to the eccentric motion. After compression, the chamber widens and allows material to flow to lower parts of the crushing chamber, and eventually to fall through and exit the machine.

The gap between the bowl and the crushing head at the closest point in the cycle is called the "closed-side setting". This can be reduced to obtain a narrower chamber and finer crushing. The two crushing surfaces are covered by replaceable steel liners (shaded in Figure 2), which can be manufactured with different cross-sectional shapes. The eccentric angle and speed of revolution of the head can also be adjusted. These variables contribute to the performance characteristics of the crusher.

2.2 SIMULATING CRUSHERS

Fitness is evaluated using a simulation of a single cone crusher. The inputs to the simulation are the:

- Physical properties of the feed (composition, hardness etc);
- Size distribution of the feed (the proportion of particles in different size fractions);
- Geometry of the mantle and bowl liners;
- Closed-side setting;
- Rotational speed of the head; and
- Eccentric angle of the head.

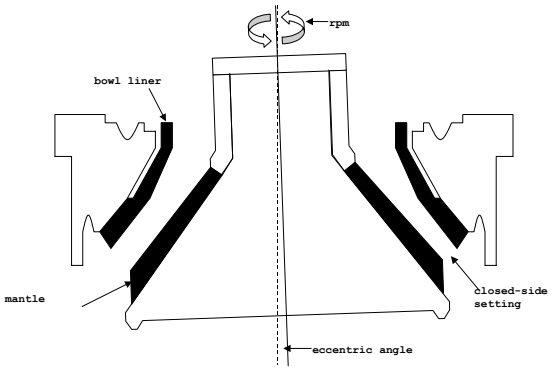


Figure 2 - Schematic diagram of a cone crusher (after (Napier-Munn, 1996) Figure 6.3)

The final four of these were selected as the design variables for the chosen problem. The outputs of the simulation are the:

- Size distribution of the product;
- Power needed to crush the feed; and
- Maximum amount of material that can flow through the crusher without overloading the crusher (its "capacity").

From these outputs it is possible to calculate the steady-state size distribution of the product and capacity of a circuit that includes the crusher. These data are used to evaluate the fitness of proposed designs. Each evaluation takes approx 300 ms on a 700MHz Pentium III.

3 ALGORITHM

The problem described above is well suited to an evolutionary algorithm approach. The problem cannot easily be described analytically, but a simulation is available that can be used to evaluate candidate solutions. The search space is large - too large for an exhaustive search - and there is little to guide an engineer in determining good designs for a given scenario. We chose an evolution strategy (ES) approach to tackle this problem, as it has similarities with other problems that have previously been successfully handled by ES. In particular, candidate designs can be described using a vector of real values, and the problem involves determining geometric shapes. Previously reported successful applications of this type include the design of a jet nozzle (Klockgether, 1970) and a flywheel (Eby, 1999).

The basic evolution strategy algorithm has the following steps:

1. Create an initial population of designs.
2. Evaluate the fitness of these designs.
3. Create a population of children by mutating the members of the current population.
4. Evaluate the fitness of these children.
5. Select the fittest designs from the parents and children together.
6. Repeat steps 3 to 5 until done.

To implement a specific instantiation of the algorithm, we must specify the representation scheme to be used, the method of fitness evaluation, the nature of the mutation operators, the selection mechanism, and the termination condition. It may be possible for infeasible designs to be generated by mutation, in which case we must also specify how to deal with these infeasible designs.

These specifications are detailed in the remainder of this section.

3.1 FITNESS

The principal objective that we are trying to maximise is the capacity of a circuit containing a given crusher. The placement of the crusher in a circuit is important because a crusher that itself has a high capacity may not be suitable if it generates a lot of oversize material: the presence of this recirculating material reduces the rate at which feed can be introduced into the circuit. We define "capacity ratio" to be the ratio of the amount of material entering the crusher to the amount of feed entering the circuit (at steady-state operation). A higher capacity ratio corresponds to more recirculating material.

The capacity of a circuit may be limited by one of three factors.

1. The capacity of the crusher. If a crusher has capacity *CAP* tons/hour and capacity ratio *CR*, the capacity of the circuit will be limited by

   $$CAP / CR$$

2. The power requirements of the crusher. A high rotational speed in particular delivers a lot of crushing but requires a lot of power. If a crusher with maximum power output *MP* kWh requires *P* kWh to process a circuit feed of F tons/hour, the capacity of the circuit will be limited by

   $$F \times (MP / P)$$

3. The capacity of the recirculation conveyor in the circuit. If a crusher has capacity ratio *CR* and the conveyor has a capacity of *MR* tons/hour, the capacity of the circuit will be limited by

   MR / (CR – 1)

Each of these factors potentially limits the capacity of the circuit, therefore the actual capacity will be the minimum of these values.

Notice the potential trade-offs for the various design variables. For example, a large closed-side setting will increase the capacity of the crusher, but will also increase the amount of recirculating material, raising the capacity ratio. Similarly, a high rotational speed will lead to more crushing in each pass through the chamber, but will also increase the power requirements of the crusher, possibly reducing the overall capacity.

Alongside maximising the capacity of the circuit, we also want to minimise the size of the product. Specifically, we define *P80* to be a measure of the size of the 80th percentile in the product (i.e. the size *k* mm such that 80% of the product is smaller than *k* mm). For technical reasons, a higher value of *P80* corresponds to a smaller product, so we want to maximise *P80*.

For the purpose of the experiments reported in this paper, we normalise both capacity and size figures by dividing by the figures for a standard design and settings.

In an earlier study, (Hingston, 2002), we combined the two objectives by defining the fitness of a design as a linear combination of them. The fitness function used in the earlier study was:

$$0.05 \times CAP + 0.95 \times P80$$

where *CAP* is the circuit capacity, *P80* is the size measure, and the constants are chosen to equalise the variability of the two components. Thus the fitness of the standard design is 1.0, and higher fitness is better.

In the present study, we use both objectives to define the Pareto ranking of a design relative to a set of potential designs. We use the ranking scheme proposed by Fonseca and Fleming (Fonseca, 1998), as described in (Veldhuizen, 2000). We define Pareto dominance for designs as follows:

> A design *u* is said to *dominate* a design *v* iff *CAP(u) > CAP(v)* and *P80(u) > P80(v)*
>
> A design *x* is *Pareto optimal* with respect to a set of designs Ω iff there is no design in Ω that dominates *x*. Thus a design that is Pareto optimal cannot be improved in any objective without degrading other objectives.
>
> Finally, the *Pareto rank* of a design *x*, with respect to a set of designs Ω, is the number of designs in Ω which dominate *x*.

Thus *x* is Pareto optimal iff *x* has a Pareto rank of 0. In this multi-objective approach, Pareto rank, rather than a combined fitness value, is used as the basis for selection.

3.2 INITIALISATION

The population is initialised with copies of the existing standard design and settings. These copies are quickly eliminated in the first few generations of a typical execution.

3.3 REPRESENTATION

The representation of the machine settings - closed-side setting, eccentric angle and rotational speed - is straightforward, these being real values within given ranges. The best way to represent the geometric shapes of the two liners is less clear. The shape of each liner is defined by its vertical cross-section. The shape of the machine structure dictates the shape of the "back" of each liner, so it is only the "front" of each liner (the actual crushing surface) that is represented.

We chose to describe each shape as a series of line segments, using a variable-length list of points, each represented by a pair of coordinates. The first coordinate pair for the first segment and the last coordinate pair for the last segment are fixed, but each other coordinate is another real-valued object variable. Thus, if there are n line segments on the mantle and m line segments on the bowl liner, then the genotype consists of a vector of

$$3 + 2(n-1) + 2(m-1)$$

real-valued object variables.

Figure 3 shows a series of liner pairs evolved during a typical run.

3.4 MUTATION

When a parent is mutated to produce a child, each object variable is mutated independently using self-adaptive mutation rates as described in (Back, 1997). Specifically, each object variable is mutated using the formula

$$X_i' = X_i + \sigma_i' \cdot N_i(0,1)$$

where $N_i(0,1)$ is a normally distributed random value with mean 0 and standard deviation 1, and each strategy parameter σ_i is mutated using the formula

$$\sigma_i' = \sigma_i \cdot \exp\left(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)\right)$$

where τ and τ' are constants set to 0.25 and 0.1 respectively, and $N(0,1)$ is sampled once for each individual.

In addition, we provided mutation operators to increase or reduce the number of segments in a liner. Whether to apply these operators is determined randomly with a fixed probability. The mutation to reduce the number of segments randomly selects two adjacent segments to merge and discards the common end point. The operator to increase the number of segments randomly selects a segment to split into two, using the segment midpoint as the common end point. This was done to allow the algorithm to generate more complex or simpler liner shapes as required.

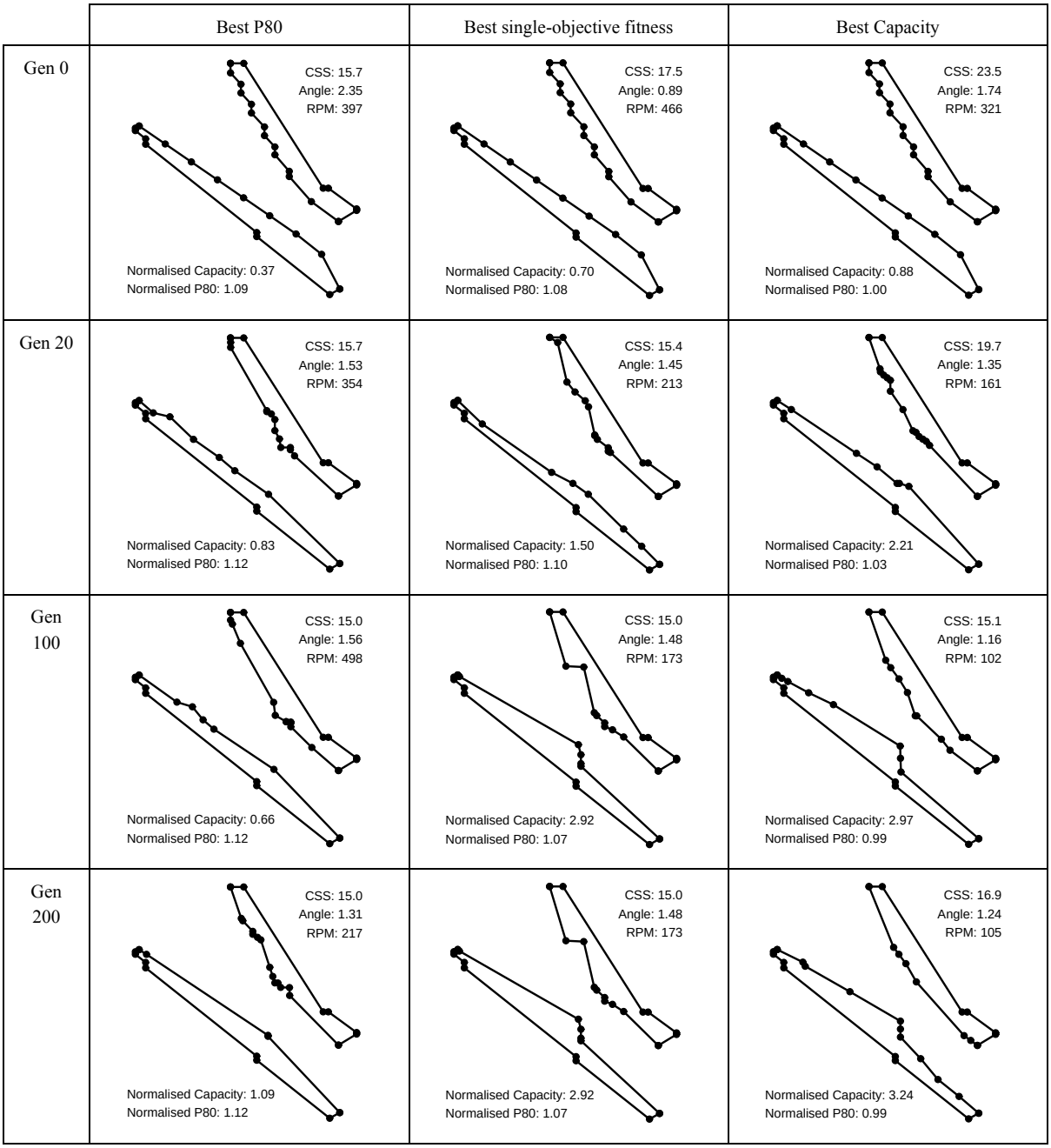


Figure 3 - A sample of evolved liner pairs

3.5 CONSTRAINTS

There are a variety of feasibility constraints upon potential designs. These can be categorised as follows:

Physical constraints The sequences of coordinate pairs must describe shapes that make sense operationally. In particular, the liners must have at least a certain thickness to be practical. We found that this constraint was violated so rarely that it is not worth the computational expense to do the checking. If the final solution returned violates this constraint, the algorithm can simply be re-run.

Setting constraints Each machine setting must be confined to a given range. This is done by repair — any value that is too low is set to the minimum value for that setting, and any that is too high is set to the maximum value.

Modeling constraints The crusher simulation is very complex and assumes (sometimes implicitly) that liners have "sensible" shapes. To keep our designs in the "sensible" region, we imposed a heuristic constraint that the sequence of x-coordinates and the sequence of y-coordinates for each liner always change monotonically.

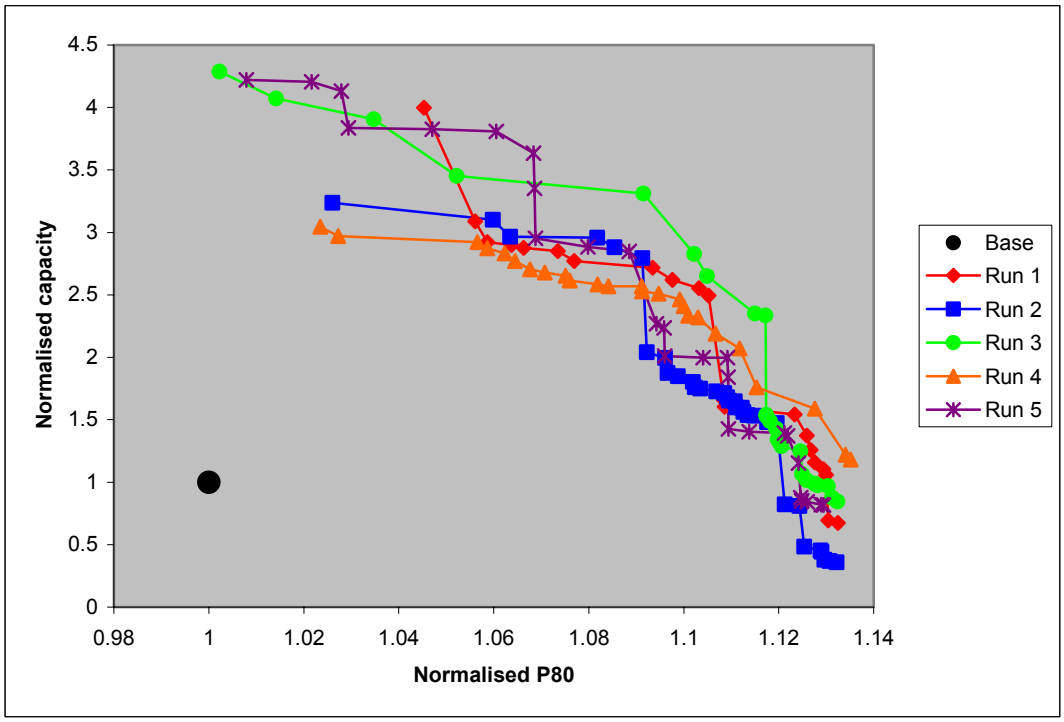


Figure 4 - Final Pareto fronts from five runs of the system

This constraint is enforced by repairing any coordinate that violates the constraint, at the time of creation. Even so, the simulation occasionally fails. In these cases, the design is assumed to be nonsensical and both capacity and *P80* are assigned an abysmal value of 0.

3.6 SELECTION

Selection is done using the standard $(\lambda + \mu)$-selection mechanism of evolution strategies, with $\lambda = \mu = 1$. Each member of the current generation becomes the parent of one child, and those with lowest rank are selected from the parents and children combined become the next generation. That is, each member of the current generation becomes the parent of one child, and the best individuals selected from the combined parents and children become the next generation.

4 RESULTS AND DISCUSSION

In this section, we describe an example set of runs of the system that is indicative of the performance on test problems. We ran the system five times with a population size of 50 for 200 generations on each run.

Figure 4 shows the final Pareto fronts for the sample runs. In all cases a good range of designs is found, showing different tradeoffs of the two objectives.

Figure 5 shows the movement of the Pareto front during Run 5 from Figure 4. Note that while the fronts for Generations 100 and 200 appear to cross, no design in Generation 200 is actually dominated by one in Generation 100. Some designs in Generation 100 are still present in Generation 200 (indeed one design in Generation 20 is still present), and the use of lines to interpolate between the population members creates the illusion of a cross-over. The situation is exacerbated by the difficulty in improving *P80* values beyond a certain level: this has been confirmed by experiments where maximising *P80* was the sole objective.

Figure 3 shows a sample of evolved liner pairs and settings from another run. The first row shows liners from Generation 0, a selection of random mutations on the standard design. The middle rows show liners from part way through the run, and the final row shows liners from the final Pareto front. The first column shows the design with the best *P80*, while the last column shows the design with the best capacity. The second column shows the design with the highest fitness according to the composite measure used in (Hingston, 2002).

Figure 6 shows the user interface during the execution of a typical run. The top right corner shows a scatter plot of the current generation in objective-space. The user can select a particular design in the plot to view its details elsewhere on the screen. The top left corner depicts the selected crusher in a circuit: it shows the liner shape and the material flows through each part of the circuit. The user can click on one of these flows to view a graph of the size distribution of the corresponding ore stream. The bottom left corner shows the settings and fitness for the selected design. The bottom right corner has various controls for the parameters of the system.

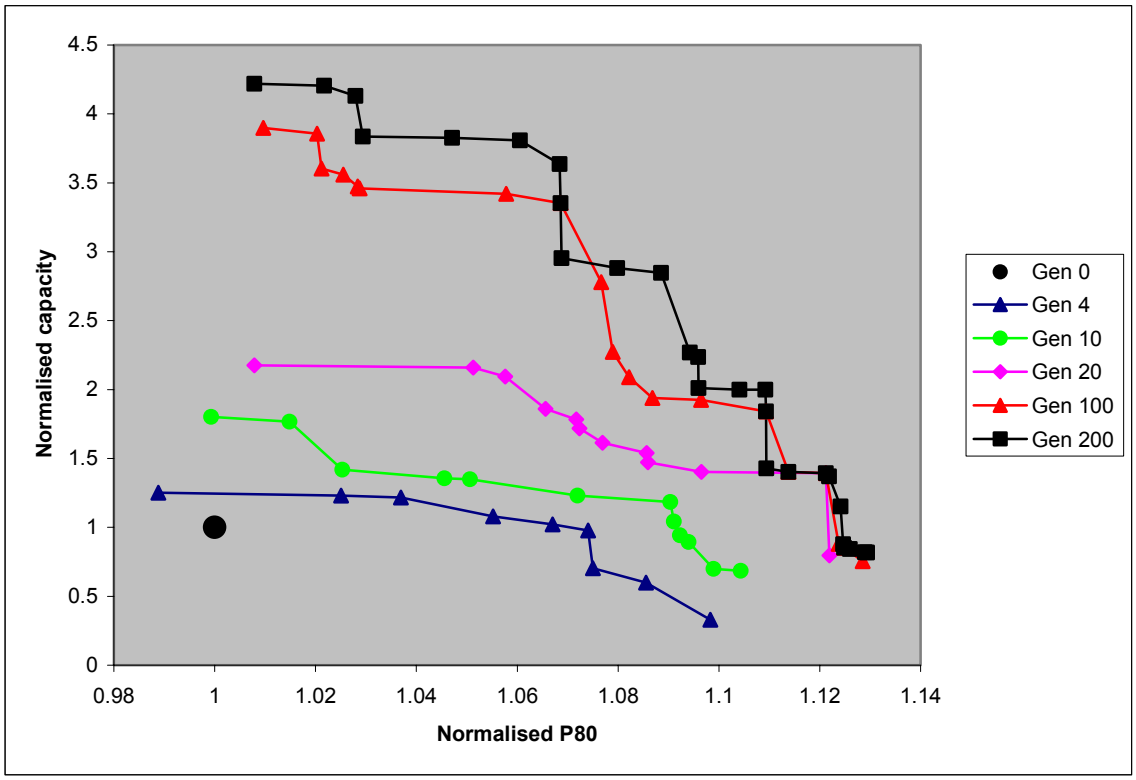


Figure 5 - Progressive Pareto fronts from Run 5 in Figure 4

The true multi-objective approach used in this study offers clear benefits in this application over the simpler approach of using a combined fitness function as in (Hingston, 2002). It removes the need for arbitrary weightings, which engineers have trouble specifying in advance. There is no need to separately apply capacity constraints, as non-dominated solutions inevitably satisfy the constraints anyway. The user interface provides an intuitive visualisation for engineers, enabling them to see the effects of trading off the different objectives on the evolved designs.

5 FUTURE WORK

The work reported here is still in the early stages of its development. While the results obtained so far are excellent, many enhancements and extensions are envisaged.

Planned enhancements to the crusher simulation are likely to make it run an order of magnitude slower. We may then need to develop special strategies to speed up the evolutionary algorithm. One possibility is to use faster, more approximate models early in the search, using a scheme similar to the injection island genetic algorithm described in (Eby, 1999).

Another aim is to include, as part of the task, the design of the circuit itself - that is, to co-evolve crushers, screens and other processing units and their settings, as well as the pattern of conveyors connecting them together. This brings in elements of network design, another application area in which evolutionary algorithms have been successful (see e.g. (Gross, 1996)). The concurrent design of this network and the machines within it will be challenging, but the potential rewards are huge.

6 CONCLUSIONS

In this paper we have described a study in the application of multi-objective evolutionary algorithms to a difficult practical engineering design problem. Our system determines the liner profiles and operating settings for a crusher in a comminution circuit. Initial results promise significant financial benefits.

In many ways, this problem is an ideal application for evolutionary algorithms - the pay-off is high; the problem is too complex to solve analytically; the search space is too large to explore unaided; we have a well defined evaluation function and a straightforward representation scheme, suitable for manipulation by genetic operators. Many challenges remain in incorporating more realism in the problem definition (for example, including variety in feed properties, interactions with other plant etc) and validating the predicted performance with field trials.

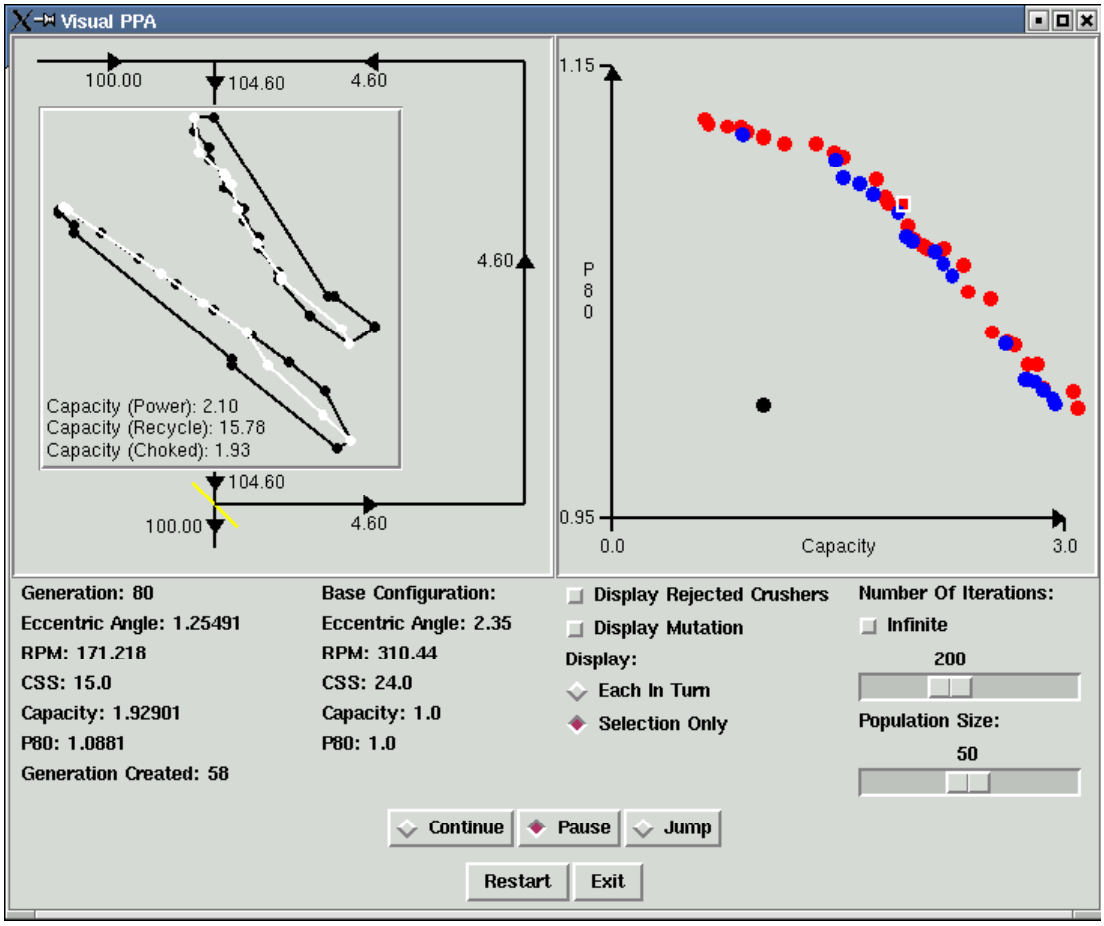


Figure 6 - The user interface of the system

References

Back, T., Ulrich, H. & Schwefel, H-P. (1997). Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation, 1*(1), 3-16.

Eby, D., Averill, R.C., Punch, W.F. & Goodman, E.D. (1999). Optimal Design of Flywheels Using an Injection Island GA. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing, 13*, 327-340.

Fonseca, C. M., & Fleming, P.J. (1998). Multiobjective Optimisation and Multiple Constraint Handling with Evolutionary Algorithms - Part I: Unified Formulation. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 28*(1), 26-37.

Gross, B., Hammel, U., Maldaner, P., Meyer, A., Roosen, P. & Schutz, M. (1996). *Optimization of Heat Exchanger Networks by Means of Evolution Strategies.* Paper presented at the Sixth Conference on Parallel Problem Solving from Nature.

Hingston, P., Barone, L. & While, L. (2002). *Evolving Crushers.* Paper presented at the Conference on Evolutionary Computation, Hawaii, 2002.

Hiorns, F. J. (1971). Wear measurements in size reduction machinery. *Chemical Process Engineering*(January), 47-49.

Klockgether, J. S., H-P. (1970). *Two-phase Nozzle and Hollow Core Jet Experiments.* Paper presented at the 11th Symposium on Engineering Aspects of Magnetohydrodynamics.

Napier-Munn, T. J., Morrell, S., Morrison, R.D. & Kojovic, T. (1996). *Mineral Comminution Circuits*. Brisbane: Julius Kruttschnitt Mineral Research Centre.

Veldhuizen, D., & Lamont, G. (2000). Multiobjective Evolutionary Algorithms: Analysing the State-of-the-Art. *Evolutionary Computation, 8*(2), 125-147.

Learning Composite Operators for Object Detection

Bir Bhanu and **Yingqiang Lin**

Center for Research in Intelligent Systems
University of California, Riverside, CA, 92521, USA
Email: {bhanu, yqlin}@vislab.ucr.edu
Tel: 909-787-3954

Abstract

In this paper, we learn to discover composite operators and features that are evolved from combinations of primitive image processing operations to extract regions-of-interest (ROIs) in images. Our approach is based on genetic programming (GP). The motivation for using GP is that there are a great many ways of combining these primitive operations and the human expert, limited by experience, knowledge and time, can only try a very small number of conventional ways of combination. Genetic programming, on the other hand, attempts many unconventional ways of combination that may never be imagined by human experts. In some cases, these unconventional combinations yield exceptionally good results. Our experimental results show that GP can find good composite operators, that consist of primitive operators designed in this paper, to effectively extract the regions of interest in images and the learned composite operators can be applied to extract ROIs in other similar images.

1 INTRODUCTION

Object detection is an important intermediate step to object recognition. The task of object detection is to locate and extract regions from an image that may contain potential objects. These regions are called regions of interest (ROIs) or object chips. The quality of object detection is dependent on the kind and quality of features extracted from an image. There are many kinds of features that can be extracted. The question is what are the appropriate features or how to synthesize features, particularly useful for detection, from the primitive features extracted from an image. The answer to these questions is largely dependent on the intuitive instinct, knowledge, previous experience and even the bias of human image experts.

In this paper, we use genetic programming (GP) to synthesize composite features, which are the output of composite operators, to perform object detection. A composite operator consists of primitive operators and it can be viewed as a combination of primitive operations on images. The basic approach is to apply a composite operator on the original image or primitive feature images generated from the original one, then the output image of the composite operator (called composite feature) is segmented to obtain a binary image or mask to extract the region containing the object from the original image. The individuals in our GP based learning are composite operators represented by binary trees whose internal nodes represent the pre-specified primitive operators and the leaf nodes represent the original image or the primitive feature images. The primitive feature images are predetermined, and they are not the output of the prespecified primitive operators.

2 MOTIVATION AND RELATED RESEARCH

2.1 MOTIVATION

In most imaging applications, an expert designs an approach to extract ROIs from images. The approach can often be dissected into some primitive operations on the original image or a set of related feature images obtained from the original one. It is the expert who, relying on his/her rich experience, figures out a smart way to combine these primitive operations to achieve good results. The task of finding a good approach is equivalent to finding a good point in the search space of *composite operators* formed by the combination of primitive operators.

The number of ways of combining primitive operators is almost infinite. The human expert can only try a very limited number of combinations and typically only the conventional ways of combination are tried. However, a GP may try many unconventional ways of combining primitive operations that may never be imagined by human experts. In some cases, it is the unconventional ways of combination that yield exceptionally good results. The inherent parallelism of GP and the speed of computers allow the portion of the search space explored by GP to be much larger than that by human experts. Although only a very small portion of the space is tried by GP, the search

performed by GP is not a random search. It is guided by the goodness of composite operators in the population. As the search goes on, GP will gradually shift the population to the portion of the space containing good operators.

2.2 RELATED RESEARCH AND OUR CONTRIBUTION

Genetic programming, an extension of genetic algorithm, was first proposed by Koza in [1]. In GP, the individuals can be binary trees, graphs or some other complicated structures of dynamically varying size. Poli [2] used GP to develop effective image filters to enhance and detect features of interest or to build pixel-classification-based segmentation algorithms. Stanhope and Daida [3] used GP paradigms for the generation of rules for target/clutter classification and rules for the identification of objects. To perform these tasks, previously defined feature sets are generated on various images and GP is used to select relevant features and methods for analyzing these features. Howard et al. [4] applied GP to automatic detection of ships in low-resolution SAR imagery using an approach that evolves detectors. Roberts and Howard [5] used GP to develop automatic object detectors in infrared images.

Unlike the work of Stanhope and Daida [3], Howard et al. [4] and Roberts and Howard [5], the input and output of each node of the tree in our system are images, not real numbers. Also, the primitive features defined in this paper are more general and easier to compute than those used in [5]. In summary, the primitive operators and primitive features designed by us are very basic and domain-independent, not specific to a kind of imagery. Thus, our system can be applied to a wide variety of images.

3 TECHNICAL APPROACH

In our GP based approach, individuals are composite operators, which are represented by binary trees. The search space of GP is the space of all possible composite operators. The space is very large. In order to illustrate this, consider only a special kind of binary tree, where each tree has exactly 30 internal nodes and one leaf node and each internal node has only one child. For 17 primitive operators and only one primitive feature image, the total number of such trees is 17^{30}. It is extremely difficult to find good operators from this vast space unless one has a smart search strategy.

3.1 DESIGN CONSIDERATIONS

There are five major design considerations, which involve determining the set of terminals, the set of primitive operators, the fitness measure, the parameters for controlling the run, and the criterion for terminating a run.

- **The Set of Terminals:** The set of terminals used in this paper are seven primitive feature images generated from the original image: the first one is the original image; the others are mean and standard deviation images obtained by applying templates of sizes 3×3, 5×5 and 7×7. These images are the input to the composite operators. GP determines which operations are applied on them and how to combine the results. To get the mean image, we translate the template across the original image and use the average pixel value of the pixels covered by the template to replace the pixel value of the pixel covered by the central cell of the template. To get the standard deviation image, we just compute the square root of the pixel value difference between the pixel in the original image and its corresponding pixel in the mean image.

- **The Set of Primitive Operators:** A primitive operator takes one or two input images, performs a primitive operation on them and stores the result in a resultant image. Currently, 17 primitive operators are used by GP to compose composite operators.

In the following, A and B are images of the same size and c is a constant. For operators such as ADD_OP, SUB_OP, MUL_OP, etc that take two images as input, the operations are performed on the pixel-by-pixel basis.

1. ADD_OP: A + B. Add two images pixel by pixel.
2. SUB_OP: A – B. Subtract image B from image A.
3. MUL_OP: A * B. Multiply images A and B.
4. DIV_OP: A / B. Divide image A by image B (If the pixel in B has value 0, the corresponding pixel in the resultant image takes the maximum pixel value in A).
5. MAX2_OP: A max B. The pixel in the resultant image takes the larger pixel value of images A and B.
6. MIN2_OP: A min B. The pixel in the resultant image takes the smaller value of pixels in images A and B.
7. ADD_CONST_OP: A + c. Increase pixel value by c.
8. SUB_CONST_OP: A - c. Decrease pixel value by c.
9. MUL_CONST_OP: A * c. Multiply pixel value by c.
10. DIV_CONST_OP: A / c. Divide pixel value by c.
11. SQRT_OP: sqrt(A). For each pixel p with value v, if $v \geq 0$, change its value to $\sqrt{v}$. Otherwise, to $-\sqrt{-v}$.
12. LOG_OP: log(A). For each pixel p with value v, if $v \geq 0$, change its value to log(v). Otherwise, to –log(-v).
13. MAX_OP: max(A). Replace the pixel value by the maximum pixel value in a 3×3, 5×5 or 7×7 neighborhood.
14. MIN_OP: min(A). Replace the pixel value by the minimum pixel value in a 3×3, 5×5 or 7×7 neighborhood.
15. MED_OP: med(A). Replace the pixel value by the median pixel value in a 3×3, 5×5 or 7×7 neighborhood.
16. REVERSE_OP: rev(A). Reverse the pixel value. Suppose the maximum and minimum pixel values of

image A are Vmax and Vmin respectively. If a pixel has value v, change its value to Vmax – v + Vmin.

17. STDV_OP: stdv(A). Obtain standard deviation image of image A by applying a template of size 3×3, 5×5 or 7×7.

- **The Fitness Measure:** The fitness value of a composite operator is computed in the following way. Suppose *G* and *G'* are foregrounds in the ground truth image and the resultant image of the composite operator respectively. Let *n(X)* denote the number of pixels within region *X*, then *Fitness = n(G∩G') / n(G ∪ G')*. The fitness value is between 0 and 1. If *G* and *G'* are completely separated, the value is 0; if *G* and *G'* are completely overlapped, the value is 1.

- **Parameters and Termination:** The key parameters are the population size M, the number of generations N, the crossover rate and the mutation rate.

The GP stops whenever it finishes the pre-specified number of generations or whenever the best operator in the population has fitness value greater than the fitness threshold.

3.2 REPRODUCTION, CROSSOVER AND MUTATION

The GP searches through the space of composite operators to generate new operators, which may be better than the previous ones. By searching through the composite operator space, GP gradually adapts the population of composite operators from generation to generation and improves the overall fitness of the whole population. More importantly, GP may find an exceptionally good operator during the search. The search is done by performing reproduction, crossover and mutation operations. The initial population is randomly generated and the fitness of each individual is evaluated.

The reproduction operation involves selecting a composite operator from the current population. In this research, we use tournament selection, where a number of individuals are randomly selected from the current population and the one with the highest fitness value is copied into the new population.

To perform crossover, two composite operators are selected on the basis of their fitness values. These two composite operators are called parents. One internal node in each of these two parents is randomly selected, and the two subtrees with these two nodes as root are exchanged between the parents. In this way, two new composite operators, called offspring, are created.

In order to avoid premature convergence, mutation is introduced to randomly change the structure of some of the individuals to help maintain the diversity of the population. Once a composite operator is selected to perform mutation operation; an internal node of the binary tree representing this operator is randomly selected, then the subtree rooted at this node is deleted, including the node selected. Another binary tree is randomly generated and this tree replaces the previously deleted subtree. The resulting new binary tree represents a new composite operator. This new composite operator replaces the old one in the population.

3.3 STEADY_STATE AND GENERATIONAL GENETIC PROGRAMMING

In *steady-state GP*, two parental composite operators are selected on the basis of their fitness for crossover. The children of this crossover, perhaps mutated, replace a pair of composite operators with the smallest fitness values. The two children are executed immediately and their fitness values are recorded. Then another two parental composite operators are selected for crossover. This process is repeated until crossover rate is satisfied. In *generational GP*, two composite operators are selected on the basis of their fitness values for crossover. Then, two composite operators with the smallest fitness values, among those that have not been selected for replacement, are selected. They will be replaced by the children of the crossover. At this time, the replacement has not occurred. The above process is repeated until crossover rate is satisfied. A composite operator may be repeatedly selected for crossover, but it cannot be repeatedly selected for replacement. After crossover operations are finished, all the children resulted from the crossover operations replace all the composite operators selected for replacement at once. In addition, we adopt an elitism replacement method that copies the best composite operator from generation to generation. The steady state and generational genetic programming algorithms are given in the following.

- **Steady-state Genetic Programming:**

```
0.  randomly generate population P and evaluate each
    composite operator in P.
1.  for gen = 1 to generation_num do
2.      keep the best composite operator in P.
3.      perform reproduction to generate population P'
        from P.
4.      number_of_crossover = population_size * cross
        over_rate / 2.
5.      for i = 1 to number_of_crossover do
6.          select 2 composite operators from P' based on
            their fitness values for crossover.
7.          select 2 composite operators with the lowest
            fitness values in P' for replacement.
8.          perform crossover operation and let the 2
            offspring composite operators replace the 2
            composite operators selected for replacement.
9.          if mutation is performed on the composite
                operators from the crossover then
10.             perform mutation on the 2 offspring
                operators with probability mutation_rate.
            end.
```

11. execute the 2 offspring composite operators and evaluate their fitness values.
end // loop 5
12. if mutation is performed on the composite operators from the whole population P' then
13. perform mutation on each composite operator with probability mutation_rate.
14. execute and evaluate mutated composite operators.
end
15. let the best composite operator from population P replace the worst composite operator in P'.
16. let P = P'
17. if the fitness value of the best composite operator in P is above fitness threshold value then
18. stop.
end
end // loop 1

- **Generational Genetic Programming:**

0. randomly generate population P and evaluate each composite operator in P.
1. for gen = 1 to generation_num do
2. keep the best composite operator in P.
3. perform reproduction to generate population P' from P. (crossover and mutation are performed on population P')
*4. number_of_crossover = population_size * crossover_rate / 2.*
*5. perform crossover number_of_crossover times and record 2 * number_of_crossover composite operators to be replaced.*
6. perform mutation on the composite operators generated from crossover or on the composite operators from the whole population. If a composite operator is mutated, recorded it for later execution.
7. execute offspring composite operators from crossover and the mutated composite operators and evaluate their fitness values.
8. put offspring composite operators from crossover in P' and remove the composite operators selected for replacement from P'.
9 let the best composite operator from population replace the worst composite operator in P'.
10. let P = P'
11. if the fitness value of the best composite operator in P is above fitness threshold value then
12. stop.
end
end // loop 1

4 EXPERIMENTS

Various experiments were performed to test the efficacy of genetic programming in extracting regions of interest from real SAR (synthetic aperture radar) images and color images. In this paper, we show some selected examples. It is to be noted that the training and testing images are different and the ground truth is used only during training. In all the experiments, the maximum size of composite operator is 30 and the threshold value used in segmentation is 0.

4.1 REAL SAR IMAGES

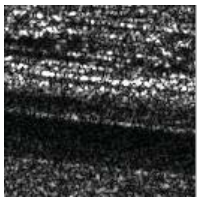
In the four experiments with real SAR images, the population size is 100, the number of generations is 100, the crossover rate is 0.6, the mutation rate is 0.1 and the selection type is tournament selection. In each experiment, GP is invoked ten times with the same parameters and the experimental results from one run and the average performance of ten runs are reported in Table 1. We select the run in which GP finds the best composite operator among the composite operators found in all ten runs to report. The first two rows show the fitness value of the best composite operator and population fitness value (average fitness value of all composite operators in the population) in the initial and final generations in the selected run. The numbers in the parenthesis in the "Best fitness" columns are the fitness values of the best composite operators on the testing SAR images. The last two rows show the average values of the above fitness values over all ten runs. The regions extracted during the training and testing by the best composite operator from the selected run are shown in the following examples.

- **Example 1. Road Extraction:** Three images contain road. The first one contains horizontal paved road and field; the second one contains vertical paved road and grass; the third one contains unpaved road and field. Training is done using the image shown in Figure 1(a) and testing is performed on images shown in Figure 3(a) and 3(c). Figure 1(b) show the ground truth provided by the user, and the white region corresponds to the road.

The generational GP was used to generate a composite operator to extract the road. The fitness threshold value is 0.90. Figure 1(c) shows the output image (corresponding to training image 1(a)) of the best composite operator in the initial population, and Figure 1(d) shows the binary image after segmentation. The output image has both positive pixels in lighter shade and negative pixels in darker shade. Positive pixels belong to the region to be extracted. The fitness value of the best composite operator in the initial population is 0.47 and the population fitness value is 0.19. Figure 1(e) shows the output image of the best composite operator after 100 generations and Figure 1(f) shows the binary image after segmentation. The fitness value of the best composite operator in the final population is 0.92 and the population fitness value is 0.89. The best composite operator has 30 internal nodes and its depth is 21. It has eight leaf nodes, two contains the original image and the other six contain 5×5 mean images,

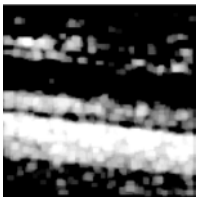
Table 1. The Performance of Genetic Programming on Various Examples of SAR Images.

	Road		Lake		River		Field	
	Best fitness	Population fitness	Best fitness	Population fitness	Best fitness	Population fitness	Best fitness	Population fitness
Initial fitness	0.47	0.19	0.65	0.42	0.43	0.21	0.62	0.44
Final fitness	0.92 (0.92, 0.89)	0.89	0.93 (0.92)	0.92	0.74 (0.84)	0.68	0.87 (0.68)	0.86
Ave. Inital fitness	0.47	0.18	0.73	0.39	0.37	0.11	0.65	0.41
Ave. Final fitness	0.81	0.76	0.92	0.87	0.68	0.58	0.84	0.77



(a) paved road vs. field

(b) ground truth



(c) best initial image

(d) segmented image

(e) best final image

(f) segmented image

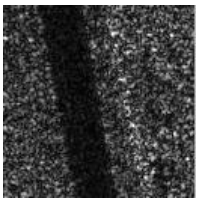
Figure 1. Training real SAR image containing road.

```
(LOG_OP (MIN2_OP (MED_OP (MAX_OP (MAX_OP
(MAX_OP (MAX_OP (MUL_CONST_OP
(DIV_CONST_OP (MAX_OP (MAX_OP
(DIV_CONST_OP (MAX_OP (MAX_OP
(MUL_CONST_OP (MAX_OP (MUL_CONST_OP
(ADD_OP (SUB_CONST_OP (MAX2_OP PF_IM3
PF_IM3)) (LOG_OP (MIN2_OP PF_IM3 (STDV_OP
PF_IM0)))))))))))))))) (DIV_CONST_OP (ADD_OP
(SUB_CONST_OP (MAX2_OP PF_IM3 PF_IM3))
(LOG_OP (MIN2_OP PF_IM3 (STDV_OP
PF_IM0)))))))
```

Figure 2. Learned composite operator tree in LISP notation.

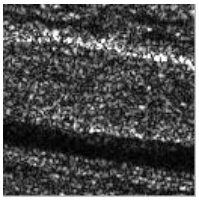
which are very useful in the noise reduction. It is shown in Figure 2, where PM_IM0 is original image and PF_IM3 is 5×5 mean image. It is possible to have a more compact tree representation of this composite operator.

We applied the composite operator obtained in the above training to the other two real SAR images shown in Figure 3(a) and 3(c). Figure 3(b) shows the region extracted by the composite operator from Figure 3(a) and the fitness value of the region, which is 0.92. Figure 3(d) shows the region extracted by the composite operator from Figure 3(c) and the fitness value of the region, which is 0.89.



(a) paved road vs. grass

(b) road detection (fitness 0.92)



(c) unpaved road vs. field

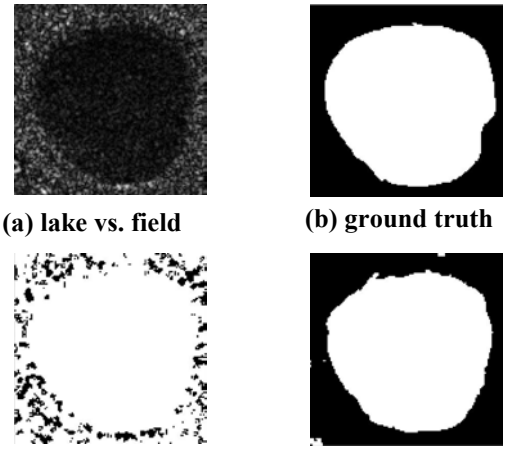
(d) road detection (fitness 0.89)

Figure 3. Testing real SAR images and corresponding road detection results.

- **Example 2. Lake Extraction:** Two SAR images contain lake. The first one contains a lake and field, and the second one contains a lake and grass. Figure 4(a) shows the original image containing lake and field. Figure 4(b) shows the ground truth provided by the user, and the white region corresponds to the lake to be extracted. Figure 5(a) shows the image containing lake and grass.

We used the SAR image containing the lake and field as the training image and applied the composite operator generated by GP to the SAR image containing the lake and grass. The steady-state GP was used to generate the composite operator and the fitness threshold value is 0.95. Figure 4(c) shows the region extracted by the best composite operator in the initial population after segmentation. The fitness value of the best composite operator in the initial population is 0.65 and the population fitness value is 0.42. Figure 4(d) shows the region extracted by the best composite operator in the final population (it is found after 65 generations) after segmentation. The fitness value of the best composite operator in the final population is 0.93 and the population fitness value is 0.92.

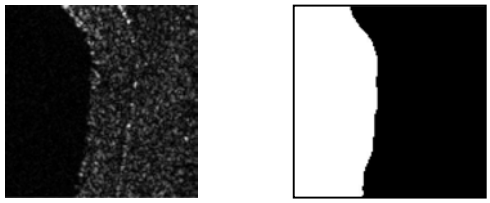
We then applied the composite operator to the image containing a lake and grass. Figure 5(b) shows region extracted and its fitness value 0.92.



(a) lake vs. field **(b) ground truth**

(c) initial segmented image **(d) final segmented image**

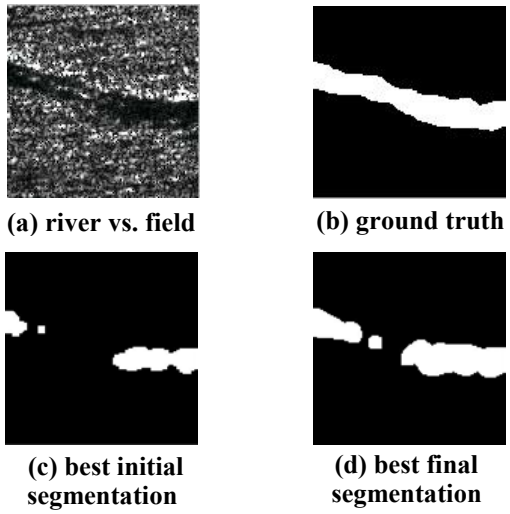
Figure 4. Real SAR image containing lake and field.



(a) lake vs. grass **(b) segmented image (fitness 0.92)**

Figure 5. Testing Real SAR image containing lake and grass.

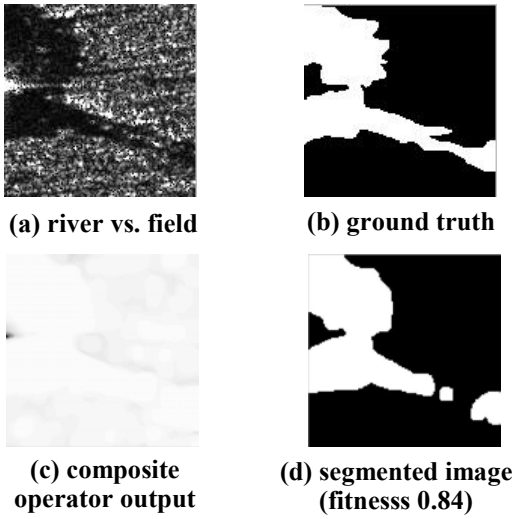
- **Example 3. River Extraction:** We have two SAR images containing river and field. Figure 6(a) and 7(a) show the original images and Figure 6(b) and 7(b) show the ground truth provided by the user. The white region in Figure 6(b) and 7(b) corresponds to the river to be extracted. The SAR image shown in Figure 6(a) was used as the training image by GP. GP generated a composite operator to extract the river in the image. Then the composite operator was applied to the SAR image shown in Figure 7(a) to test its efficacy in extracting the river.



(a) river vs. field **(b) ground truth**

(c) best initial segmentation **(d) best final segmentation**

Figure 6. Training real SAR images containing river.



(a) river vs. field **(b) ground truth**

(c) composite operator output **(d) segmented image (fitnesss 0.84)**

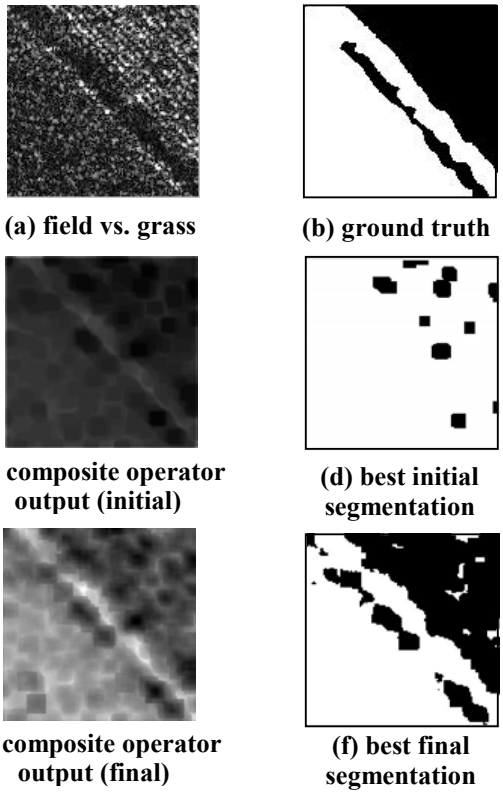
Figure 7. Testing real SAR images containing river.

The steady-state GP was used to generate the composite operator and the fitness threshold value is 0.85. Figure 6(c) shows the region extracted by the best composite operator in the initial population after segmentation. The fitness value of the best composite operator in the initial population is 0.43 and the population fitness value is 0.21. Figure 6(d) shows the region extracted by the best composite operator in the final population (it was found after 40 generations) after segmentation. The fitness value of the best composite operator in the final population is 0.74 and the population fitness value is 0.68. The fitness value of the best composite operator in the final population is not very good. Two reasons account for this. First, the river in Figure 6(a) accounts for only a small percentage of the total area in the image. Second, there are some islands in the river. These islands are similar to the field,

i.e., pixels belong to the islands have similar pixel values to those belong to the field, but they are not excluded from the ground truth.

We applied the composite operator to the image shown in Figure 7(a). Figure 7(c) shows the output image of the composite operator. Figure 7(d) shows the region extracted after segmentation and its fitness value 0.84. This number is larger than the fitness value in the training. The main reason is that the river in Figure 7(a) accounts for a much larger percentage of the total area of the image than that in Figure 6(a).

- **Example 4. Field Extraction:** Two SAR images contain field and grass. Figure 8(a) and 9(a) show the original images and Figure 8(b) and 9(b) show the ground truth. The white region in Figure 8(b) and 9(b) corresponds to the field to be extracted. We consider extracting field from a SAR image containing field and grass as the most difficult task among the four experiments with the SAR images, since the grass and field are similar to each other. We used the SAR image in Figure 8(a) as the training image and applied the composite operator generated by GP to the SAR image in Figure 9(a).



(a) field vs. grass **(b) ground truth**

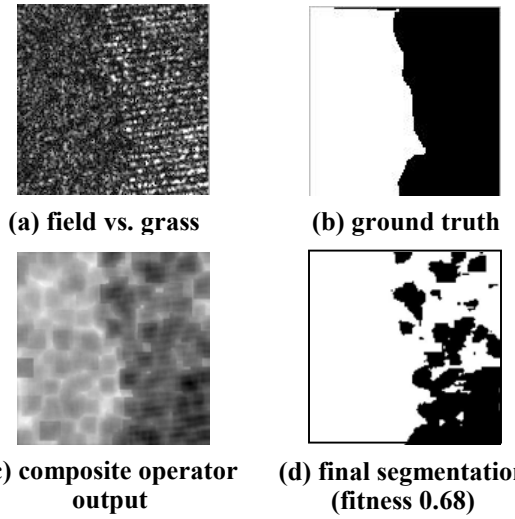
(c) composite operator output (initial) **(d) best initial segmentation**

(e) composite operator output (final) **(f) best final segmentation**

Figure 8. Training real SAR image containing field and grass.

The generational genetic programming was used to generate the composite operator and the fitness threshold value is 0.85. Figure 8(c) shows the output image of the best composite operator in the initial population. The fitness value of the best composite operator in the initial population is 0.62 and the population fitness value is 0.44. Figure 8(d) shows the region extracted after segmentation. Figure 8(e) shows the output image of the best composite operator in the final population and Figure 8(f) shows the region extracted after segmentation. The fitness value of the best composite operator in the final population is 0.87 and the population fitness value is 0.86. Figure 8(c) is very dark. One may not see anything meaningful in this image. The reason is that almost all the pixels in this image have very low pixel values. Some pixels have positive pixel values, but the pixel values are close to 0.

We applied the composite operator to the image in Figure 9(a). Figure 9(c) shows the output image of the composite operator. Figure 9(d) shows the region extracted after segmentation and its fitness value 0.68.



(a) field vs. grass **(b) ground truth**

(c) composite operator output **(d) final segmentation (fitness 0.68)**

Figure 9. Testing real SAR image containing field and grass.

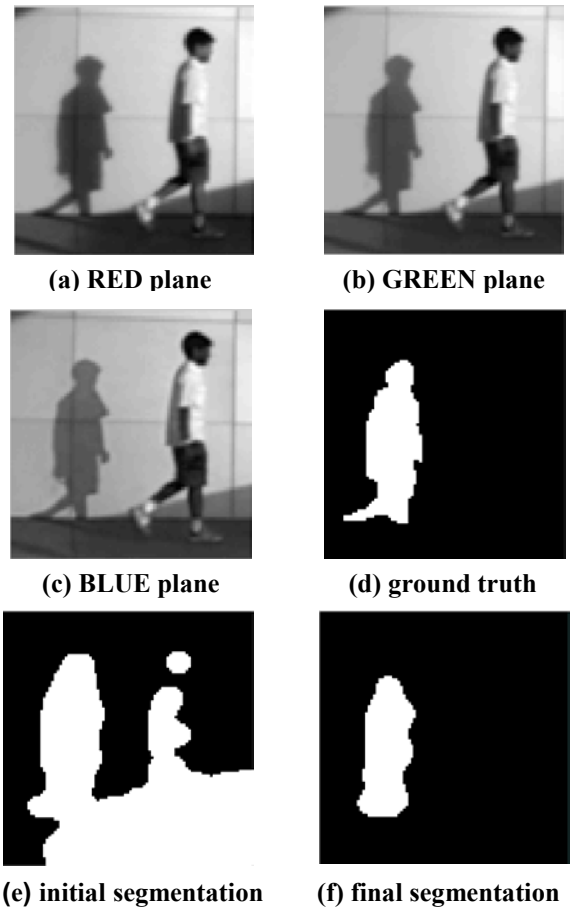
4.2 COLOR IMAGES

In this subsection, we attempt to generate a composite operator to extract the shadow of a person from an RGB color image. The generated composite operator was then tested on two other similar images.

Figure 10 shows the image used for training and the ground truth provided by the user. We don't show a color image, rather the RED, GREEN and BLUE planes of the color image in Figure 10(a), 10(b), 10(c) respectively. The RED, GREEN and BLUE planes of the color image are gray scale intensity images and they are used as primitive feature images in this experiment.

The generational genetic programming was used to generate the composite operator. The population size is 200, the number of generation is 200, the fitness threshold value is 0.80, the crossover rate is 0.1 and the mutation rate is 0.05.

Figure 10(e) shows the region extracted by the best

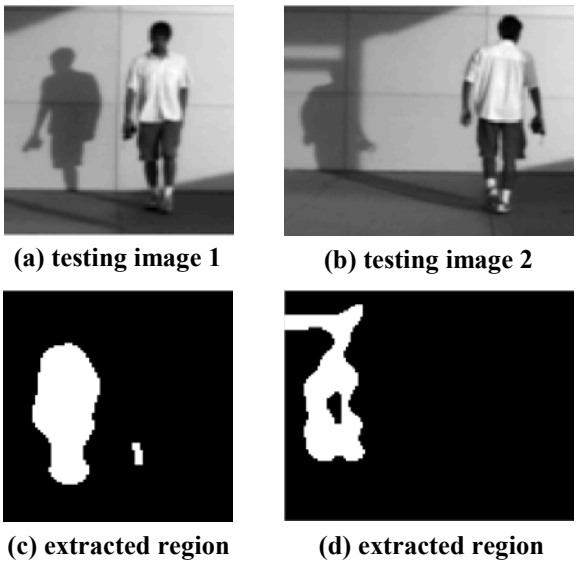


(a) RED plane (b) GREEN plane

(c) BLUE plane (d) ground truth

(e) initial segmentation (f) final segmentation

Figure 10. RED, GREEN and BLUE planes of RGB color image used in training.



(a) testing image 1 (b) testing image 2

(c) extracted region (d) extracted region

Figure 11. GREEN planes of testing RGB color images.

composite operator in the initial population after segmentation. The fitness value of the best composite operator in the initial population is 0.28 and the population fitness value is 0.16. Figure 10(f) shows the region extracted by the best composite operator in the final population after segmentation. The fitness value of the best composite operator in the final population is 0.80 and the population fitness value is 0.76. GP found a good composite operator to extract the shadow.

The composite operator generated by GP was then applied to another two similar color images to test its efficacy in extracting the shadow. The GREEN planes of these two color images are shown in Figures 11(a) and 11(b). When the composite operator is applied to extract shadow regions in these two color images, the RED, GREEN and BLUE planes of the color images are the primitive feature images used by the composite operator. The testing results are shown in Figure 11(c) and Figure 11(d). The fitness values for these two results were 0.76 and 0.54 respectively. It can be seen from these images that the composite operator generated by GP is capable of extracting shadows in the color images similar to the color image used in training.

5 CONCLUSIONS

Our experimental results show that the primitive operators selected by us are effective. GP can find good composite operators to extract the regions of interest in an image and the composite operators can be applied to extract ROIs in other similar images. In our experiments, we did not find any significant difference between the steady-state and generational genetic programming algorithms. In the future, we plan to extend this work by designing smart crossover and mutation operators and discovering new features within the regions of interest for automated object recognition.

Acknowledgement: This research is supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. government.

References

[1] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

[2] R. Poli, "Genetic programming for feature detection and image segmentation," in *Evolutionary Computation*, T. C. Forgarty Ed., pp. 110-125, 1996.

[3] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," *Proceeding Conference. Evolutionary Programming VII*, pp. 735-744, 1998.

[4] D. Howard, S. C. Roberts, and R. Brankin, "Target detection in SAR imagery by genetic programming," *Advances in Engg. Software*, 30(5), pp. 303-311, May 1999.

[5] S. C. Roberts and D. Howard, "Evolution of vehicle detectors for infrared line scan imagery," *Proceeding Workshop, Evolutionary Image Analysis, Signal Processing and Telecommunications*, pp. 110-125, 1999.

Grammatical Evolution and Corporate Failure Prediction

Anthony Brabazon
Dept. Of Accountancy,
University College Dublin, Ireland.

Michael O'Neill
Dept. Of Computer Science And Information Systems,
University of Limerick, Ireland.

Robin Matthews
Center for International Business Policy
Kingston Business School, London

Conor Ryan
Dept. Of Computer Science And Information Systems,
University of Limerick, Ireland.

Abstract

This study examines the potential of Grammatical Evolution to uncover a series of useful rules which can assist in predicting corporate failure using information drawn from financial statements. A sample of 178 publically quoted, failed and non-failed US firms, drawn from the period 1991 to 2000 are used to train and test the model. The preliminary findings indicate that the methodology has much potential.

1 Introduction

The objective of this study is to determine whether an evolutionary automatic programming methodology, Grammatical Evolution, is capable of uncovering useful structure in financial ratio information which can be used to predict corporate failure.

Corporate failure is an essential component of an efficient market economy, allowing the recycling of financial, human and physical resources into more productive organisations [10] [32]. However, many parties including shareholders, providers of debt finance, employees, suppliers, customers, managers and auditors have an interest in the financial health of organisations as corporate failure can impose significant private costs on all these groups. Even where total failure can be averted by firm reorganization, the costs of major restructuring can be as high as 12% to 19% of firm value [41]. If a trajectory leading to corporate failure can be identified sufficiently early to allow successful intervention, these costs can be reduced. [14] suggest that indicators of corporate failure can be present up to ten years prior to final failure, providing an opportunity for construction of models which predict corporate failure.

Corporate failure can arise for many reasons. It may occur due to a single catastrophic event or it may be the end result of a lengthy process of decline. Under the second perspective, corporate failure is a process which starts with management defects, leading to poor decisions, leading to financial deterioration and finally resulting in corporate collapse [3] [20]. Most attempts to predict corporate failure implicitly assume that management decisions critically impact on firm performance [5] [20]. The premise of this paper is that a series of poor decisions lead to a deterioration in the financial health of the firm and finally to its demise. Although the decisions are not directly observable, their consequent affect on the financial health of the firm can be observed.

Previous studies have utilised a wide variety of explanatory variables in the construction of corporate distress models[1], including variables drawn from the financial statements of firms, from financial markets, general macro-economic variables [29], and non-financial, firm-specific information, including director turnover [27]. In this study, we limit our attention to information drawn from the financial statements of firms.

1.1 Potential for application of evolutionary automatic programming

There are a number of reasons to suppose that the use of an evolutionary automatic programming (EAP) approach such as Genetic Programming (GP) or GE can prove fruitful in the prediction of corporate failure. The problem domain is characterised by a lack of a strong theoretical framework, with many plausible, competing explanatory variables. The selection of quality explanatory variables and model form represents a high-dimensional combinatorial problem, giving rise to potential for an EAP methodology. Use of EAP also facilitates the development of complex fitness functions including discontinuous, non-differentiable functions. This is of particular importance in a prediction domain as fitness criteria may be complex. Generally, the cost of misclassifications of failing / non-failing firms will be asymmetric. Another useful feature of an EAP approach is that it can produce human-readable rules that have the potential to enhance understanding of the problem domain.

[1] [3] and [22] provide good reviews of the development of empirical research in bankruptcy prediction.

1.2 Motivation for study

This study was motivated by a number of factors. Although a substantial volume of research utilising traditional statistical modelling techniques has been undertaken in the corporate failure domain, to date only a limited number of studies have applied GA / GP methodologies [39] [19] [6]. This study builds on these initial studies and adopts a novel evolutionary automatic programming approach.

1.3 Structure of paper

This contribution is organised as follows. Section 2 provides a short discussion of prior literature in the corporate failure domain and outlines the definition of corporate failure employed in this study. Section 3 provides an introduction to Grammatical Evolution. Section 4 describes both the data utilised, and the model development process adopted in this paper. Section 5 provides the results of the constructed model. Finally, conclusions and a discussion of the limitations of the contribution are provided in Section 6.

2 Background

Formal research into the prediction of corporate failure has a long history [12] [35] [16]. Early statistical studies such as [7], adopted a univariate methodology, identifying which accounting ratios had greatest classification accuracy when identifying failing and non-failing firms. Although this approach did demonstrate classification power, it suffers from the shortcoming that a single weak financial ratio may be offset (or exacerbated) by the strength (or weakness) of other financial ratios. [1] addressed this issue by employing a linear discriminant analysis (LDA) model, which utilised both financial and market data concerning a firm, and this was found to improve the classification accuracy of the developed models. The discriminant function which produced the best classification performance in Altman's 1968 study was:

$$Z = .012X_1 + .014X_2 + .033X_3 + .006X_4 + .999X_5$$

where:
X_1 = working capital to total assets
X_2 = retained earnings to total assets
X_3 = earnings before interest and taxes to total assets
X_4 = market value of equity to book value of total debt
X_5 = sales to total assets

LDA assumes both multi-variate normality and the equality of the covariance matrices of each classification group. Generally, these assumptions do not hold for financial ratio data. Other statistical methodologies which have been applied include logit and probit regression models [13] [42] [24]. In recent times, methodologies applied to this problem domain have included neural networks [34] [33] [40], genetic algorithms [39] [19] and hybrid neural network / genetic algorithm models [6].

2.1 Definition of Corporate Failure

No unique definition of corporate failure exists [3]. Possible definitions range from failure to earn an economic rate of return on invested capital given the risk of the business, to legal bankruptcy followed by liquidation of the firm's assets. Any attempt to uniquely define corporate failure is likely to prove problematic. While few publicly quoted companies fail in any given year[2], poorer performers are liable to acquisition by more successful firms. Thus, two firms may show a similar financial trajectory towards failure, but one firm may be acquired and 'turned-around' whilst the other may fail.

The definition of corporate failure adopted in this study is the entry of a firm into Chapter 7 or Chapter 11 of the US Bankruptcy code. The selection of this definition provides an objective benchmark as the occurrence, and date of occurrence, of either of these events can be determined through examination of regulatory filings. Chapter 7 covers corporate liquidations and Chapter 11 covers corporate reorganizations, which usually follow a period of financial distress. Under Chapter 11, management is required to file a reorganisation plan in bankruptcy court and seek approval for this plan. When the court grants approval for the plan the firm is released from Chapter 11 bankruptcy and continues to trade. In most cases, Chapter 11 reorganisations involve significant financial losses for both shareholders [30] and creditors [11] of the distressed firm. [23], in a study of the outcomes of Chapter 11 filings, found that 'there were few sucessful reorganisations' (p. 125), despite a perception that some management teams were using Chapter 11 filings as a deliberate strategy for dealing with certain firm specific events such as onerous labor contracts or produce liability claims[3].

2.2 Explanatory variables utilised in prior literature

Five groupings of explanatory variables, drawn from financial statements, are given prominence in prior literature [4]:

i. Liquidity
ii. Debt
iii. Profitability
iv. Activity / Efficiency
v. Size

[2] [42] reports that this rate is less than 0.75% in the US and [22] suggests that the rate is below 2% in the UK.

[3] [23] report that out of a sample of 73 firms entering Chapter 11 between 1980 and 1986, only 44 were sucessfully reorganized with only 15 of these firms emerging from Chapter 11 with more than 50% of their prebankruptcy assets.

Liquidity refers to the availability of cash resources to meet short-term cash requirements. Debt measures focus on the relative mix of funding provided by shareholders and lenders. Profitability considers the rate of return generated by a firm, in relation to its size, as measured by sales revenue and/or asset base. Activity measures consider the operational efficiency of the firm in collecting cash, managing stocks and controlling its production or service process. Firm size provides information on both the sales revenue and asset scale of the firm and also provides a proxy metric on firm history. The groupings of potential explanatory variables can be represented by a wide range of individual financial ratios, each with slightly differing information content. The groupings themselves are interconnected, as weak (or strong) financial performance in one area will impact on another. For example, a firm with a high level of debt, may have lower profitability due to high interest costs. Whatever modelling methodology is applied, the initial problem is to select a quality set of model inputs from a wide array of possible financial ratios, and then to combine these ratios using suitable weightings in order to construct a high quality classifier. Given the large search space, an evolutionary automatic programming methodology has promise.

2.3 Results of Prior Literature

Earlier studies [1] [2] [3] [4] [9] [6] have suggested that the classification accuracy of failure models increases rapidly as the date of final failure approaches. Generally, results indicate that the most significant deterioration in financial ratios occurs in the third year prior to eventual failure. Although sample sizes, dates and methodologies differ between studies, these findings have been replicated in a broad series of studies. In Altman's 1968 study [1], the developed LDA model correctly identified (in-sample) 95% of failing firms one year prior to failure. The classification accuracy fell to 72% and 48% in the second and third year prior to failure. [2] demonstrated a classification accuracy (in-sample) of approximately 93% in the year prior to failure declining to 68% four years prior to failure. [40] report in-sample classifications of 98.7% for a neural network model, one-year prior to failure, and 95% for a logit model on the same data. [39] reports in-sample classification accuracy for a GA based model of approximately 97%, one year prior to failure. In-sample classification accuracies provide a limited assessment of model generalisability. Enhanced in-sample classification accuracies could result from data-mining. Hence, in this study, developed models are solely assessed based on classification performance on out-of-sample data.

A description of the evolutionary automatic programming system used to evolve rules for prediction of corporate failure is provided in the next section.

3 Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary algorithm that can evolve computer programs in any language. Rather than representing the programs as syntax trees, as in traditional GP [18], a linear genome representation is adopted. A genotype-phenotype mapping process is used to generate the output program for each individual in the population. Each individual, a variable length binary string, contains in its codons (groups of n-bits, where n equals 8 here) the information to select production rules from a Backus Naur Form (BNF) grammar. The BNF is a plug-in component to the mapping process, that represents the output language in the form of production rules. It is comprised of a set of non-terminals that can be mapped to elements of the set of terminals, according to the production rules.

An example excerpt from a BNF grammar is given below. These productions state that S can be replaced with either one of the non-terminals `expr`, `if-stmt`, or `loop`.

```
S ::= expr      (0)
    | if-stmt   (1)
    | loop      (2)
```

The grammar is used in a generative process to construct a program by applying production rules, selected by the genome, beginning from the start symbol of the grammar.

In order to select a rule in GE, the next codon value on the genome is read and placed in the following formula:

$$Rule = Codon\ Value\ MOD\ \#Rules$$

If the next codon integer value was 4, given that we have 3 rules to select from as in the above example, we get $4\ MOD\ 3\ =\ 1$. S will therefore be replaced with the non-terminal `if-stmt`.

Beginning from the left hand side of the genome codon integer values are generated and used to select rules from the BNF grammar, until one of the following situations arise:

i. A complete program is generated. This occurs when all the non-terminals in the expression being mapped, are transformed into elements from the terminal set of the BNF grammar.

ii. The end of the genome is reached, in which case the *wrapping* operator is invoked. This results in the return of the genome reading frame to the left hand side of the genome once again. The reading of codons will then continue unless an upper threshold representing the maximum number of wrapping events has occurred during this individual's mapping process. This threshold is currently set to ten events.

iii. In the event that a threshold on the number of wrapping events is exceeded and the individual is still incom-

pletely mapped, the mapping process is halted, and the individual assigned the lowest possible fitness value.

GE uses a steady state replacement mechanism, such that, two parents produce two children the best child replacing the worst individual in the current population if the child has a greater fitness. In the case where both children have the same fitness and are better than the current population worst, a child is chosen at random. The standard genetic operators of point mutation, and crossover (one point) are adopted. It also employs a duplication operator that duplicates a random number of codons and inserts these into the penultimate codon position on the genome. A full description of GE can be found in [26] [25] [31].

4 Problem Domain & Experimental Approach

This section describes both the data utilised by, and the model development process adopted in, this study.

4.1 Sample Definition and Model Data

A total of 178 firms were selected judgementally (89 failed, 89 non-failed), from the Compustat Database [8][4]. The criteria for selection of the failed firms were:

i. Inclusion in the Compustat database in the period 1991-2000

ii. Existence of required data for a period of three years prior to entry into Chapter 7 or Chapter 11

iii. Sales revenues must exceed $1M

The first criterion limits the study to publicly quoted, US corporations. The second criterion injects an element of bias into the sample in that companies without a three year financial history prior to entering Chapter 7 or Chapter 11 are omitted. Twenty-two potential explanatory variables, are collected for each firm for the three years prior to entry into Chapter 7 or Chapter 11[5]. For every failing firm, a matched non-failing firm is selected. They are matched both by industry sector and size (sales revenue three years prior to failure)[6]. The set of 178 matched firms are randomly divided into model building (128 firms) and out-of-sample (50 firms) datasets. The dependant variable is binary (0,1), representing either a non-failed or a failed firm.

[4]Firms from the financial sector were excluded on grounds of lack of comparability of their financial ratios with other firms in the sample.

[5]The date of entry into Chapter 7 or Chapter 11 was determined by examining regulatory filings for each firm.

[6]It is recognised that the use of an equalised, matched sample entails sampling bias and eliminates firm size and industry nature as potential explanatory variables (see [22] for a detailed discussion of these points). It is noted that utilising an unmatched sample imposes its own bias.

The choice of explanatory variables is hindered by the lack of a clear theoretical framework which explains corporate failure [5] [38] [40]. Most empirical work on corporate failure adopts an ad-hoc approach to variable selection. Prior to the selection of the potential explanatory variables for inclusion in this study, a total of ten previous studies were examined [7] [1] [2] [9] [24] [33] [17] [6] [37] [21]. These studies employed a total of 58 distinct ratios divided amongst the five classifications noted by [4]. A subset of 22 of the most commonly used financial ratios was selected for this study. The selected ratios were:

i. EBIT / Sales
ii. EBITDA / Sales
iii. EBIT / Total Assets
iv. Gross Profit / Sales
v. Net Income / Total Assets
vi. Net Income / Sales
vii. Return on Assets
viii. Return on Equity
ix. Return on Investment
x. Cash / Sales
xi. Sales / Total Assets
xii. Inventory / Cost of Goods Sold
xiii. Inventory / Working Capital
xiv. Fixed Assets / Total Assets
xv. Retained Earnings / Total Assets
xvi. Cash from Operators / Sales
xvii. Cash from Operations / Total Liabilities
xviii. Working Capital / Total Assets
xix. Quick Assets / Total Assets
xx. Total Liabilities / Total Assets
xxi. Leverage
xxii. EBIT / Interest

5 Results

Accuracy of the developed models is assessed based on the overall classification accuracy arising in both the model-building and out-of-sample datasets. For simplicity, the cost of each type of classification error is assumed to be symmetric in this study. The fitness function could be easily altered to bias the model development process to minimise a specific type of classification error if required, and later studies will address this issue.

The classification problem which plays an important role in decision-making, consists of assigning observations to disjoint groups [28]. The decision scenario faced in this study

comprises a binary classification. In general, the construction of classifier systems such as linear discriminant analysis, logit or ANN models consists of two components, the determination of a valuation rule which is applied to each observation, and the determination of a 'cut-off' value. The grammar adopted in this study is given below and its output is interpreted using a fixed 0.5 cut-off value to produce a classification.

```
lc : output = expr ;
expr : ( expr ) + ( expr )
     | coeff * var
var : var1[index]  | var2[index]  | var3[index]
    | var4[index]  | var5[index]  | var6[index]
    | var7[index]  | var8[index]  | var9[index]
    | var10[index] | var11[index] | var12[index]
    | var13[index] | var14[index] | var15[index]
    | var16[index] | var17[index] | var18[index]
    | var19[index] | var20[index] | var21[index]
    | var22[index]
coeff :  ( coeff ) op ( coeff )
      | float
op : + | - | *
float : 20 | -20 | 10 | -10 | 5 | -5 | 4 | -4
      | 3 | -3 | 2 | -2 | 1 | -1 | .1 | -.1
```

The above grammar generates classifiers of the form:

$$output = (< some\ expression > *varX) \ + \ ..$$
$$+ (< some\ expression > *varY)$$

Any combination of the twenty two explanatory variables can be exploited by an evolved classifier, including zero or more occurrences of any one variable. This is in contrast to an LDA approach where classifiers would generally utilise all the explanatory variables within the expression. In the LDA case, of course some of those variables could be *switched off* by multiplying their value by zero.

Three series of models were constructed using explanatory variables drawn from one, two and three years (T1, T2 and T3) prior to failure. For each set of models, 30 runs were conducted using population sizes of 500, running for 100 generations, adopting one-point crossover at a probability of 0.9, and bit mutation at 0.01, along with a steady state replacement strategy.

A plot of the mean average and mean best fitness values over the 30 runs for each time period can be seen in Figure 1.

Years Prior to Failure	In Sample	Out Of Sample
1	85.9%	80%
2	82.8%	80%
3	75.8%	70%

Table 1: The best classification accuracies reported for each of the three years prior to failure.

The best individuals evolved for each period are reported in Table[7]1. In-sample it can be seen that the performance of the models generated falls off gracefully as we move out each year. It is interesting to note that out-of-sample there is no performance difference between the evolved models in periods T1 and T2, both giving 80% correct classifications.

The best classifiers evolved for each period are given in Table 2.

5.1 Discussion

The classification results of the evolved models show promise. Despite drawing a sample from a wide variety of industrial sectors, the models demonstrate a high classification accuracy in and out-of-sample, which degrades gracefully rather than suddenly in the third year prior to failure. Although the evolved models were free to select from twenty-two potential explanatory variables, it is notable that each model only employed a small subset of these. This lends support to the proposition that many financial ratios have similar information content and that classification accuracy is not enhanced through the construction of models with a large number of these ratios. It is also notable that each model has (approximately) included one variable drawn from the four main categories of explanatory variables suggested in the corporate failure literature (Liquidity, Debt, Profitability, and Activity/Efficiency), lending empirical support to earlier work[8].

The risk factors suggested by each model differ somewhat and contain some less-intuitive but nonetheless plausible findings.

Examining the best classifer evolved for T1 suggests that risk factors include low return on assets, low retained earnings and a high ratio of total liabilities to total assets, which concords with financial intuition. Less obviously, a high ratio of inventory to net liquid assets (inventory+receivables+cash-payables) is also a risk factor, possibly resulting from depletion of cash or build-up of inventories as failure approaches.

Risk factors for firms at T2 include low return on assets and a low ratio of earnings to interest costs. Less intuitive risk factors indicated are a low ratio of fixed assets to total assets and a high ratio of sales to total assets. The former could indicate firms with a lower safety cushion of saleable resources which could be sold to stave-off collapse, the latter could be serving as a proxy variable for firms with rapid sales growth. Over-rapid sales growth can be a danger signal, indicating that management resources are being spread too-thinly.

Finally, risk factors indicated for firms at T3 include low return on assets, a low ratio of profit to interest charge, a low level of cash generated from operations and as for T2, a high ratio of sales to total assets.

Although each model is evolved seperately, the general form of each model appears consistent with the hypothesis that

[7]Calculation of Press's Q statistic [15] for each of these models rejects a null hypothesis, at the 5% level, that the out-of-sample classification accuracies are not significantly better than chance.

[8]Size, the fifth category, is not considered in this study due to the matching process utilized.

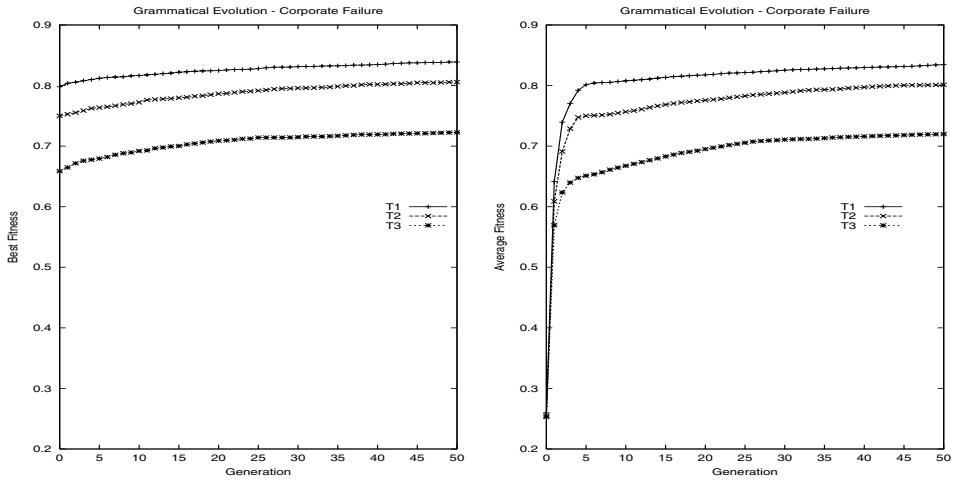


Figure 1: A comparison of the mean best fitness between T1, T2, and T3 (left), and of the mean average fitness values (right) for the same time periods.

Years Prior to Failure	Best Classifier
1	output = -3*var7-5*var8+3*var17-20*var19+4*var24
2	output = -2*var8+10*var15-10*var18-2*var25
3	output = -4*var8+20*var15-72.9*var20-10*var25

Table 2: The best classifiers evolved for each of the years analysed.

there is a financial trajectory towards failure. Low profits and high interest payments as a percentage of profits in periods T3 and T2 indicate a firm in financial difficulties, with an erosion of the safety cushion provided by high levels of (saleable) fixed assets indicated in the risk factors at T2. The final year prior to failure sees additional risk factors indicated by high levels of debt and reducing cash balances / inventory build-up.

6 Conclusions & Future Work

GE was shown to successfully evolve useful rules for prediction of corporate failure with a performance equivalent to that reported in prior studies. In assessing the performance of the developed models, a number of caveats must be borne in mind. The premise underlying this paper (and all empirical work on corporate failure prediction) is that corporate failure is a process, commencing with poor management decisions, and that the trajectory of this process can be tracked using accounting ratios. This approach does have inherent limitations. It will not forecast corporate failure which results from a sudden environmental event. It is also likely that the explanatory variables utilised contain noise. Commentators [5] [36] have noted that managers may attempt to utilise creative accounting practices to manage earnings and / or disguise signs of distress. Additionally, financial data is produced on a time-lagged basis. Although not undertaken in this preliminary study, the incorporation of non-financial qualitative explanatory variables or variables related to the firm's share price performance could further improve classification accuracy. Another limitation of all models of corporate distress is that the underlying relationships may not be stationary [4] [17]. Accounting standards and economic environment faced by firms will vary over time. Finally, the firms sampled in this study are relatively large and are publically quoted. Thus, the findings of this study may not extend to small businesses.

Despite these limitations, the high economic and social costs of corporate failure imply that models which can indicate declining financial health will have utility. Given the lack of a clear theory underlying corporate failure, empirical modelling usually adopts a combinatorial approach, a task for which GE is well suited. The results of this preliminary study indicate that GE has useful potential for the construction of corporate failure models.

References

[1] Altman, E. (1968). 'Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy', *Journal of Finance*, 23:589-609.

[2] Altman, E., Haldeman, R. and Narayanan, P. (1977). 'ZETA Analysis: A new model to identify bankruptcy risk of corporations', *Journal of Banking and Finance*, 29-54.

[3] Altman, E. (1993). *Corporate Financial Distress and Bankruptcy*, New York: John Wiley and Sons Inc.

[4] Altman, E. (2000). 'Predicting Financial Distress of Companies: Revisiting the Z-score and Zeta models', *http://www.stern.nyu.edu/ ealtman/Zscores.pdf, October 2001*.

[5] Argenti, J. (1976). *Corporate Collapse: The Causes and Symptoms*, London: McGraw-Hill.

[6] Back, B., Laitinen, T., Sere, K. and van Wezel, M. (1996). 'Chosing Bankruptcy Predictors using Discriminant Analysis, Logit Analysis and Genetic Algorithms', *Technical Report no. 40, Turku Centre for Computer Science, Turku School of Economics and Business Administration.*.

[7] Beaver, W. (1968). 'Financial Ratios as Predictors of Failure', *Journal of Accounting Research - Supplement: Empirical Research in Accounting*, 71-102.

[8] Compustat Database. http://www.compustat.com

[9] Dambolena, I. and Khoury, S. (1980). 'Ratio Stability and Corporate Failure', *Journal of Finance*, 35(4):1017-1026.

[10] Easterbrook, F. (1990). 'Is Corporate Bankruptcy Efficient?', *Journal of Financial Economics*, 27:411-417.

[11] Ferris, S., Jayaraman, N. and Makhija, A. (1996). 'The Impact of Chapter 11 filings on the Risk and Return of Security Holders, 1979-1989', *Advances in Financial Economics*, 2:93-118.

[12] Fitzpatrick, P. (1932). *A Comparison of the Ratios of Successful Industrial Enterprises with Those of Failed Companies*, Washington: The Accountants' Publishing Company.

[13] Gentry, J., Newbold, P. and Whitford, D. (1985). 'Classifying Bankrupt Firms with Funds Flow Components', *Journal of Accounting Research*, 23(1):146-160.

[14] Hambrick, D. and D'Aveni, R. (1988). 'Large Corporate Failures As Downward Spirals', *Administrative Science Quarterly*, 33:1-23.

[15] Hair, J., Anderson, R., Tatham, R. and Black, W. (1998). *Multivariate Data Analysis*, Upper Saddle River, New Jersey: Prentice Hall.

[16] Horrigan, J. (1965). 'Some empirical bases of financial ratio analysis', *The Accounting Review*, July 1965, 558-568.

[17] Kahya, E. and Theodossiou, P. (1996). 'Predicting Corporate Financial Distress: A Time-Series CUSUM Methodology', *Review of Quantitative Finance and Accounting*, 13:71-93.

[18] Koza, J. (1992). *Genetic Programming*. MIT Press.

[19] Kumar, N., Krovi, R. and Rajagopalan, B. (1997). 'Financial decision support with hybrid genetic and neural based modelling tools', *European Journal of Operational Research*, 103:339-349.

[20] McRobert, A. and Hoffman, R. (1997). *Corporate Collapse: An early warning system for lenders, investors and suppliers*, Roseville:NSW, McGraw-Hill (Australia).

[21] Moody's (2000). 'RiskCalc For Private Companies: Moody's Default Model', *http://riskcalc.moodysrms.com/us/research/crm/56402.pdf*, July 2001.

[22] Morris, R. (1997). *Early Warning Indicators of Corporate Failure: A critical review of previous research and further empirical evidence*, London: Ashgate Publishing Limited.

[23] Moulton, W. and Thomas, H. (1993). 'Bankruptcy As a Deliberate Strategy: Theoretical Considerations and Empirical Evidence', *Strategic Management Journal*, 14:125-135.

[24] Ohlson, J. (1980). 'Financial Ratios and the Probabilistic Prediction of Bankruptcy', *Journal of Accounting Research*, 18:109-131.

[25] O'Neill, M. (2001). Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution. PhD thesis, University of Limerick, 2001.

[26] O'Neill M., Ryan C. (2001) Grammatical Evolution, *IEEE Trans. Evolutionary Computation*. 2001.

[27] Peel, M., Peel, D. and Pope, P. (1986). 'Predicting Corporate Failure: Some Results for the UK Corporate Sector', *Omega International Journal of Management Science*, 14:5-12.

[28] Pendharkar, P. (2001). An empirical study of design and testing of hybrid evolutionary-neural approach for classification, *Omega*, 29:361-374.

[29] Rose, P., Andrews, W.and Giroux, G. (1982). 'Predicting Business Failure: A Macroeconomic Perspective', *Journal of Accounting, Auditing and Finance*, Fall:20-31.

[30] Russel, P., Branch, B. and Torbey, V. (1999). 'Market Valuation of Bankrupt Firms: is there an anomaly?', *Quarterly Journal of Business and Economics*, 38:55-76.

[31] Ryan C., Collins J.J., O'Neill M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming*, 83-95, Springer-Verlag.

[32] Schumpeter, J. (1934). *The Theory of Economic Development*, Cambridge, MA: Harvard Business Press.

[33] Serrano-Cina, C. (1996). 'Self organizing neural networks for financial diagnosis', *Decision Support Systems*, 17:227-238.

[34] Shah, J. and Murtaza, M. (2000). 'A Neural Network Based Clustering Procedure for Bankruptcy Prediction', *American Business Review*, 18(2):80-86.

[35] Smith, R. and Winakor, A. (1935). 'Changes in the Financial Structure of Unsuccessful Corporations', *University of Illinois, Bureau of Business Research, Bulletin No. 51*.

[36] Smith, T. (1992). *Accounting for Growth*. London: Century Business.

[37] Sung, T., Chang, N. and Lee, G. (1999). 'Dynamics of Modelling in Data Mining: Interpretative Approach to Bankruptcy Prediction', *Journal of Management Information Systems*, 16(1):63-85.

[38] Trigueiros, D. and Taffler R. J. (1996). 'Neural Networks and Empirical Research in Accounting', *Accounting and Business Research*, 26(4):347 - 355.

[39] Varetto, F. (1998). 'Genetic algorithms in the analysis of insolvency risk', *Journal of Banking and Finance*, 22(10):1421 - 1439.

[40] Wilson, N., Chong, K. and Peel, M. (1995). 'Neural Network Simulation and the Prediction of Corporate Outcomes: Some Empirical Findings', *International Journal of the Economics of Business*, 2(1):31-50.

[41] Wruck, K. (1990). 'Financial Distress, Reorganization and Organizational Efficiency', *Journal of Financial Economics*, 27(2):419-444.

[42] Zmijewski, M. (1984). 'Methodological Issues Related to the Estimation of Financial Distress Prediction Models', *Journal of Accounting Research - Supplement*, 59-82.

Evolving Neural Networks for the Classification of Galaxies

Erick Cantú-Paz and Chandrika Kamath
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551
cantupaz,kamath2@llnl.gov

Abstract

The FIRST survey (Faint Images of the Radio Sky at Twenty-cm) is scheduled to cover 10,000 square degrees of the northern and southern galactic caps. Until recently, astronomers classified radio-emitting galaxies through a visual inspection of FIRST images. Besides being subjective, prone to error and tedious, this manual approach is becoming infeasible: upon completion, FIRST will include almost a million galaxies. This paper describes the application of six methods of evolving neural networks (NNs) with genetic algorithms (GAs) to identify bent-double galaxies. The objective is to demonstrate that GAs can successfully address some common problems in the application of NNs to classification problems, such as training the networks, choosing appropriate network topologies, and selecting relevant features. The results indicate that most of the methods we tried performed equally well on our data, but using a GA to select features produced the best results.

1 INTRODUCTION

The Faint Images of the Radio Sky at Twenty-cm (FIRST) survey (Becker et al., 1995) started in 1993 with the goal of producing the radio equivalent of the Palomar Observatory Sky Survey. Using the Very Large Array at the National Radio Astronomy Observatory, FIRST is scheduled to cover more than 10,000 square degrees of the northern and southern galactic caps. At present, FIRST has covered about 8,000 square degrees, producing more than 32,000 two-million pixel images. At a threshold of 1 mJy, there are approximately 90 radio-emitting galaxies, or radio sources, in a typical square degree.

Radio sources exhibit a wide range of morphological types that provide clues to the source's class, emission mechanism, and properties of the surrounding medium. Sources with a bent-double morphology are of particular interest as they indicate the presence of clusters of galaxies, a key project within the FIRST survey. FIRST scientists currently identify the bent-double galaxies by visual inspection, which—besides being subjective, prone to error and tedious—is becoming increasingly infeasible as the survey grows.

Our goal is to bring automation to the classification of galaxies using techniques from data mining, such as neural networks. Neural networks (NNs) have been used successfully to classify objects in many astronomical applications (Odewahn et al., 1992; Storrie-Lombardi et al., 1992; Adams & Woolley, 1994). However, the success of NNs largely depends on their architecture, their training algorithm, and the choice of features used in training. Unfortunately, determining the architecture of a neural network is a trial-and-error process; the learning algorithms must be carefully tuned to the data; and the relevance of features to the classification problem may not be known a priori. Our objective is to demonstrate that genetic algorithms (GAs) can successfully address the topology selection, training, and feature selection problems, resulting in accurate networks with good generalization abilities. This paper describes the application of six combinations of genetic algorithms and neural networks to the identification of bent-double galaxies.

This study is one of a handful that compares different methods to evolve neural nets on the same domain (Roberts & Turenga, 1995; Siddiqi & Lucas, 1998; Grönross, 1998). In contrast with other studies that limit their scope to two or three methods, we compare six combinations of GAs and NNs against hand-

designed networks. Most of the methods we tried performed equally well on our data, but using a GA to select features yielded the best results. The experiments also show that most of the GA and NN combinations produced significantly more accurate classifiers than we could obtain by designing the networks by hand.

The next section outlines the problem of detecting bent-double galaxies in the FIRST data. Section 3 describes several existing combinations of GAs and NNs. Section 4 presents our experiments and reports the results. The paper concludes with our observations and plans for future work.

2 FIRST SURVEY DATA

Figure 1 has several examples of radio sources from the FIRST survey. While some bent-double galaxies are relatively simple in shape (examples (a) and (b)), others, such as the ones in examples (e) and (f), can be rather complex. Note the similarity between the bent-double in example (a) and the non-bent-double in example (c).

Data from FIRST are available on the FIRST web site (sundog.stsci.edu). There are two forms of data available: image maps and a catalog. The images in figure 1 are close-ups of galaxies. The catalog (White et al., 1997) is obtained by fitting two-dimensional Gaussians to each radio source on an image map. Each entry in the catalog corresponds to a single Gaussian.

We decided that, initially, we would identify the radio sources and extract the features using only the catalog. The astronomers expected that the catalog was a good approximation to all but the most complex of radio sources, and several of the features they thought were important in identifying bent-doubles were easily calculated from the catalog.

We identified the features for the bent-double problem through extensive conversations with FIRST astronomers. When they justified their decisions of identifying a radio source as a bent-double, they placed great importance on spatial features such as distances and angles. Frequently, the astronomers would characterize a bent-double as a radio-emitting "core" with one or more additional components at various angles.

In the past, we have concentrated our work on instances described by three catalog entries, because we have more labeled examples of this type. Our previous experience with this data suggested that the best accuracies are usually achieved using features extracted considering triplets of catalog entries (as opposed to pairs or single entries). Therefore, in the remainder of this paper we focus on the 20 triplet features that we extracted. A full list of features is described elsewhere (Fodor et al., 2000).

Unfortunately, our training set is relatively small, containing 195 examples for the three-catalog entry sources. Since the bent- and non-bent-doubles must be manually labeled by FIRST scientists, putting together an adequate training set is non-trivial. Moreover, scientists are usually subjective in their labeling of galaxies, and the astronomers often disagree in the hard-to-classify cases. There is also no ground truth we can use to verify our results. These issues imply that the training set itself is not very accurate, and there is a limit to the accuracy we can obtain.

Among the 195 labeled examples of 3-entry sources, 28 are non-bent and 167 are bent-double galaxies. This unbalanced distribution in the training set presents problems in estimating the accuracy of the NNs, which are discussed in section 4.

3 GENETIC NEURAL NETWORKS

Genetic algorithms and neural networks have been used together in several ways, and this section presents a brief review of previous work. In particular, GAs have been used to search for the weights of the network and to select the most relevant features of the training data. GAs have also been used to design the structure of the network. It is well known that to solve nonlinearly separable problems, the network must have at least one hidden layer between the inputs and outputs; but determining the number and the size of the hidden layers is mostly a matter of trial and error. GAs have been used to search for these parameters, as well as for the pattern of connections and for developmental instructions to generate a network. The interested reader may consult the reviews by Branke (1995), Schaffer (1994) and Yao (1999).

3.1 TRAINING NETWORKS WITH GAs

Training a NN is an optimization task with the goal of finding a set of weights that minimizes an error measure. The search space is high dimensional and, depending on the error measure, it may contain numerous local optima. Some network training algorithms, such as backpropagation (BP), use some form of gradient search, and may get trapped in local optima.

A straightforward combination of genetic algorithms and neural networks is to use the GA to search for weights that make the network perform as desired. The architecture of the network is fixed by the user

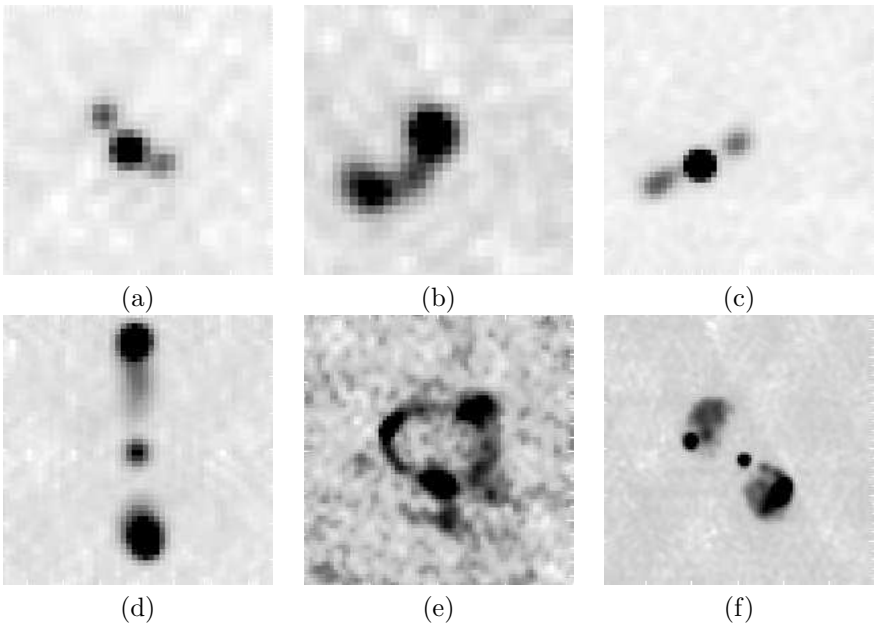


Figure 1: Example radio sources: (a)-(b) Bent-doubles, (c)-(d) Non-bent doubles, (e)-(f) Complex Sources

prior to the experiment. In this method, each individual in the GA represents a vector with all the weights of the network. There are two popular variations:

- Use the weights found by the GA without any further refinement (Caudell & Dolan, 1989; Montana & Davis, 1988; Whitley & Hanson, 1989).
- Use the GA to find a promising set of weights from which a gradient-based method can quickly reach an optimum (Skinner & Broughton, 1995). The motivation is that GAs quickly identify promising regions of the search space, but they may not fine-tune parameters very fast.

These approaches are straightforward and numerous studies report good results. However, since adjacent layers in a network are usually fully connected, the total number of weights is $O(n^2)$, where n is the number of units. Longer individuals usually require larger populations, which in turn result in higher computational costs. Therefore, the GA may be efficient for small networks, but this method may not scale up well. Another drawback is the so-called permutations problem (Radcliffe, 1990). The problem is that by permuting the hidden nodes of a network, the representation of the weights in the chromosome would change, but the network is functionally the same. Some permutations may not be suitable for GAs because crossover might easily disrupt favorable combinations of weights. To ameliorate this problem, Thierens et al. (1991) suggested to place incoming and outgoing weights of a hidden node next to each other, which was the encoding we used.

3.2 FEATURE SELECTION

Besides searching for weights, GAs may be used to select the features that are input to the NNs. The training examples may contain irrelevant or redundant features, but it is generally unknown a priori which features are relevant. Avoiding irrelevant or redundant features is desirable not only because they increase the size of the network and the training time, but also because they may reduce the accuracy of the network.

Applying GAs to select features is straightforward using what is referred to as the wrapper approach: the chromosome of the individuals contains one bit for each feature, and the value of the bit determines whether the feature will be used in the classification (Brill, Brown, & Martin, 1990; Brotherton & Simpson, 1995). The individuals are evaluated by training the networks (that have a predetermined structure) with the subset of features indicated by the chromosome. The resulting accuracy estimate is used to calculate the fitness.

3.3 DESIGNING NETWORKS WITH GAs

As mentioned before, the topology of a network is crucial to its performance. If a network has too few nodes and connections, it may not be able to learn the re-

quired concept. On the other hand, if a network has too many nodes and connections, it may overfit the training data and have poor generalization. GAs have been used successfully to design the topology of NNs. There are two basic approaches for applying GAs to the design of NNs: use a direct encoding to specify every connection of the network or evolve an indirect specification of the connectivity.

The key idea behind direct encodings is that a neural network can be regarded as a directed graph where each node represents a neuron and each edge is a connection. A common method of representing directed graphs is with a binary connectivity matrix: the i, j-th element of the matrix is one if there is an edge between nodes i and j, and zero otherwise. The connectivity matrix can be represented in a GA simply by concatenating its rows or columns (Miller et al., 1989; Belew et al., 1990). Using this method, Whitley et al. (1990) showed that the GA can find topologies that learn faster than the typical fully-connected feedforward network. The GA can be explicitly biased to favor smaller networks, which can be trained faster.

A simple method to avoid specifying all the connections is to commit to a particular topology and learning algorithm, and then use the GA to find the parameter values that complete the network specification. For example, with a fully-connected feedforward topology, the GA may search for the number of layers and the number of neurons per layer. Another example would be to code the parameters of a particular learning algorithm, such as the momentum and the learning rate of BP (Belew et al., 1990; Marshall & Harrison, 1991). Of course, this method is constrained by the initial choice of topology and learning algorithm.

Another approach is to use a grammar to encode rules that govern the development of a network. Kitano (1990) introduced the earliest grammar-based approach. He used a connectivity matrix to represent the network, but instead of encoding the matrix directly in the chromosome, the matrix is generated by a graph-rewriting grammar. The chromosomes contain rules that rewrite scalar elements into 2×2 matrices.

In this grammar, there are 16 terminal symbols that are 2×2 binary matrices. There are 16 non-terminal symbols, and the rules have the form $n \rightarrow m$, where n is one of the scalar non-terminals, and m is a 2×2 matrix of non-terminals. There is an arbitrarily designated start symbol, and the number of rewriting steps is fixed by the user.

To evaluate the fitness, the rules are decoded and the connectivity matrix is developed by applying all the rules that match non-terminal symbols. Then, the connectivity matrix is interpreted and the network is constructed and trained with BP.

Other examples of grammar-based developmental systems are the work of Boers and Kuiper (1992) with Lindenmayer systems, Gruau's "cellular encoding" method (Gruau, 1992), and the system of Nolfi, Elman, and Parisi (1994) that simulates cell growth, migration, and differentiation.

4 EXPERIMENTS

This section details the experimental methods and the results that we obtained with six combinations of neural networks and genetic algorithms.

The programs were written in C++ and compiled with g++ version 2.96. The experiments were executed on a single processor of a Linux (Red Hat 7.1) workstation with dual 1.5 GHz Intel Xeon processors and 512 Mb of memory. The programs used a Mersenne Twister random number generator.

All the GAs used a population of 50 individuals. We used a simple GA with binary encoding, pairwise tournament selection, and multi-point crossover. The number of crossover points was varied in each experiment according to the length of the chromosomes, l. In all cases, the probability of crossover was 1, and the probability of mutation was set to $1/l$. The initial population was initialized uniformly at random.

The experiments used feedforward networks with one hidden layer. All neurons are connected to a "bias" unit with constant output of 1.0. Unless specified otherwise, the output units are connected to all the hidden units, which in turn are connected to all the inputs. In feedforward operation, the units compute their net activation as

$$\text{net} = \sum_{i=1}^{d} x_i w_i + w_0,$$

where d is the number of inputs to the neuron, x_i is an input and w_i is the corresponding weight, w_0 is the weight corresponding to the "bias" unit. Each unit emits an output according to $f(\text{net}) = \tanh(\beta * \text{net})$, where β is a user-specified coefficient. Simple backpropagation was used in some of the experiments. The weights from the hidden to the output layer were updated using $\Delta w_{kj} = \eta \delta_k y_j = \eta(t_k - z_k) f'(\text{net}_\text{k}) y_j$, where η denotes the learning rate, k indexes the output units, t_k the desired output, z_k the actual output, f' is the derivative of f, and y_j is the output of the j-th hidden unit. The weights from the i-th input to the

hidden layer were updated using

$$\Delta w_{ji} = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f'(\text{net}_j) x_i.$$

In all experiments, each feature in the data was linearly normalized to the interval $[-1, 1]$. The type of galaxy was encoded in one output value (-1 for bent and 1 for non-bent). When backpropagation was used, the examples were presented in random order for 20 epochs. All the results reported are averages over 10 runs of the algorithms. Comparisons were made using standard t-tests with 95% confidence.

4.1 FITNESS CALCULATION

One of the crucial design decisions for the application of GAs is the calculation of fitness values for each member of the population. Since we are interested in networks that predict accurately the type of galaxies not used in training, the fitness calculation must include an estimate of the generalization ability of the networks.

There are multiple ways to estimate generalization. Since we do not have much training data, hold-out methods (dividing the data into training and testing sets and perhaps an additional validation set) are not practical. To calculate the fitness, we used the accuracy estimate of five-fold crossvalidation trials. In this method, the data D is divided into five non-overlapping sets, $D_1, ..., D_5$. At each iteration i (from 1 to 5), the network is trained with $D \backslash D_i$ and tested on D_i. The average of the five tests was used as the fitness. A better estimate of accuracy would be to use an average of multiple crossvalidation experiments, but we found the cost excessive.

To correct for the unbalanced distribution of bent and non-bent examples in our training data, we calculate the accuracy as the geometric mean of the accuracies of each class of galaxy (bent and non-bent) (Kubat & Matwin, 1997). Using the geometric mean gives equal weight to the accuracies on both types of galaxies in the overall performance.

4.2 TRAINING NETWORKS WITH GAs

We implemented the first of the methods described in section 3.1: the GA was used to find the network's weights. The network had 20 inputs that correspond to each of the features in the data, 25 hidden nodes, and one output. Each weight was represented with 10 bits, and the range of possible weights was $[-10, 10]$.

For this experiment, the GA used a population of 50 individuals, each with a length of $l = 5510$ bits (there are 551 total weights). The number of crossover points was set at 25, and the mutation rate was 0.00018 ($\approx 1/l$). As in all experiments, pairwise tournament selection without replacement was used.

The second training method described in section 3.1 is to run BP using the weights represented by the individuals in the GA to initialize the network. We implemented this method and used the same network architecture and GA parameters as in the first experiment. Each network was trained using 20 epochs of BP with a learning rate η of 0.1 and β of 0.4.

The entries WEIGHTS and WEIGHTS+BP in table 1 present the average accuracy of the best networks found in each run of the GA for these two sets of experiments. The results highlighted in bold in the table are the best results and those not significantly worse than the best (according to the t-test, which may detect more differences than there actually exist). The addition of BP produces a significant improvement in the bent-double accuracy rate, which is of primary importance to the astronomers. However, the improvement in the overall accuracy is not significant.

4.3 FEATURE SELECTION

The next combination of GAs and NNs is to use the GA to select the features that will be used to train the networks, as described in section 3.2. As in the previous experiment, we set the number of hidden units to 25, the learning rate η to 0.1 and β to 0.4. The networks were trained with 20 epochs of BP.

Our data has 20 features, and therefore the chromosomes in the GA are 20 bits long. The GA used one-point crossover and the same parameters as in previous experiments. The accuracy results are labeled FEATURE SEL and are significantly better than the other results in table 1.

The GAs consistently selected about half of the features, and frequently selected features that appear to be relevant to the identification of bent-double galaxies, such as symmetry measures and angles.

4.4 DESIGNING NETWORKS WITH GAs

For our first application of GAs to network design, the GA was used to find the number of hidden units, the parameters for BP, and the range of initial weights as described in section 3.3. The learning rate was encoded with four bits and the range of possible values was $[0, 1]$. The coefficient β for the activation function was also encoded with four bits and its range was $[0, 1]$. The upper and lower ranges for the initial weights were

encoded with five bits each and were allowed to vary in $[-10, 0]$ and $[0, 10]$, respectively. Finally, the number of hidden units was represented with seven bits and could take values in $[0, 127]$.

After extracting the parameters from a chromosome, a network was built and initialized according to the parameters and trained with 20 epochs of BP. There is no explicit bias to prefer smaller networks, but there is an implicit bias toward networks that can learn quickly, since we are using only 20 epochs of BP. It is probable that small networks learn faster than larger ones, and so it is likely that the GA favors small networks.

The GA used two-point crossover and the same parameters as in previous experiments. The accuracy results are labeled PARAMETERS in table 1. On average, the best learning rate found by the GA was 0.82 (with 0.06 std. error), which is higher than the usual recommendation of 0.1–0.2 (Duda, Hart, & Stork, 2001). Perhaps the learning rate is high because of the implicit bias for learning quickly. This bias may also explain the average number of hidden units being relatively small at 15.6 (std. error 2.8). The average β was 0.16 (0.01), and the range of initial weights was $[-3.51, 3.45]$ (both with std. errors of 0.4).

The next experiment used the GA to search for a connectivity matrix as described in section 3.3. We fixed the number of hidden units to 25, the learning rate to 0.1 and β to 0.4. The neurons are numbered consecutively starting with the inputs and followed by the hidden units and outputs. The connectivity matrix is encoded by concatenating its rows. Since we allow direct connections between the inputs and the outputs, the string length is $(hidden + outputs) * inputs + hidden * outputs = (26*20)+(25*1) = 545$ bits. For this longer string, we use 10 crossover points, and the same GA parameters as before. The results corresponding to this method are labeled MATRIX in table 1.

We also implemented Kitano's graph rewriting grammar method. We limited the number of rewriting steps to 6, resulting in networks with at most 64 units. Since the chromosomes encode four 2×2 binary matrices for each of the 16 rules, the string length is 256 bits. The GAs used five crossover points. The results obtained with this method are labeled GRAMMAR in table 1.

4.5 COMPARISON AND DISCUSSION

Table 1 summarizes the results obtained with each method. The results show few differences among the various methods in the accuracy rate for bent-doubles. While the direct encoding of connections (MATRIX) has the best accuracy, four other methods do not appear to be significantly less accurate. In terms of the accuracy on the non-bents and the overall accuracy, it is clear that the feature selection method obtained the best results.

We also performed numerous experiments with networks designed by hand. The best parameters that we could find for 20 epochs of backpropagation were those used in the experiments with the GAs: $\beta = 0.1$, the learning rate was 0.4, and the number of hidden was 25. The average of ten 10-fold crossvalidation experiments resulted in an accuracy on the non-bents of only 16.4% (with std. error of 1.7) and on the bents of 99.69% (0.16). The overall accuracy estimated with the geometric mean is a disappointing 23.41% (2.02).

Increasing the number of training epochs to 100 raised the standard the geometric mean accuracy to 72.69% (0.32). The accuracy on the non-bents also improved to 56.7%, while the accuracy on the bents decreased slightly to 94.38%.

5 CONCLUSIONS

This paper presented a comparison of six combinations of GAs and NNs for the identification of bent-double galaxies in the FIRST survey. Our experiments suggest that, for this application, some combinations of GAs and NNs can produce accurate classifiers that are competitive with networks designed by hand. For our application, we found few differences among the GA and NN combinations that we tried. The only consistently best method was to use the GA to select the features used to train the networks, which suggests that some of the features in the training set are irrelevant or redundant.

There are several avenues to extend this work. The highly unbalanced training set presents some difficulties that could be avoided or ameliorated by including more examples of the minority class. However, extending the training set is non-trivial, because the labeling is subjective and disagreements among the experts are common.

Other optimization techniques, evolutionary and traditional, can be used to train NNs. In this paper we used a simple genetic algorithm with a binary encoding, but other evolutionary algorithms operate on vectors of real numbers that can be directly mapped to the network's weights or the BP parameters (but not to a connectivity matrix, a grammar, or a feature selection application). There are other combinations of GAs and NNs that we did not include in this study, but appear promising. For instance, since evolutionary algorithms use a population of networks, a natural

Method	Bent-Doubles	Non-Bent	Overall
Weights	86.34 (2.83)	**78.01 (4.13)**	80.98 (2.41)
Weights+BP	**91.89 (0.67)**	75.23 (0.87)	81.68 (0.53)
Feature Sel	**92.99 (0.55)**	**83.65 (1.41)**	**87.51 (0.77)**
Parameters	**92.35 (0.89)**	69.13 (1.56)	78.76 (0.57)
Matrix	**93.58 (0.46)**	70.77 (1.34)	80.22 (0.69)
Grammar	**92.84 (0.69)**	73.73 (1.40)	81.78 (0.72)

Table 1: Mean accuracies on the bent and non-bent doubles and overall accuracy for different combinations of GAs and NNs using the *geometric* mean of class-wise accuracies as fitness. The numbers in parenthesis are the standard errors, and the results in bold are the best and those not significantly worse than the best.

extension of this work would be to use evolutionary algorithms to create ensembles that combine several NNs to improve the accuracy of classifications.

A disadvantage of using genetic algorithms in combination with neural networks is the long computation time required. This can be an obstacle to applying these techniques to larger data sets, but there are numerous alternatives to improve the performance of genetic algorithms. For instance, we could approximate the fitness evaluation using sampling or we can exploit the inherently parallel nature of GAs using multiple processors.

Acknowledgments

We gratefully acknowledge our FIRST collaborators Robert Becker, Michael Gregg, David Helfand, Sally Laurent-Muehleisen, and Richard White for their technical interest and support of this work. We would also like to thank Imola K. Fodor and Nu Ai Tang for useful discussions and computational help.

UCRL-JC-147020. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

Adams, A., & Woolley, A. (1994). Hubble classification of galaxies using neural networks. *Vistas in Astronomy*, *38*, 273–280.

Becker, R. H., White, R., & Helfand, D. (1995). The FIRST survey: Faint images of the radio sky at twenty-cm. *Astrophysical Journal*, *450*, 559.

Belew, R., McInerney, J., & Schraudolph, N. (1990). *Evolving networks: Using the genetic algorithm with connectionist learning* (Tech. Rep. No. CS90-174). San Diego: University of California, Computer Science and Engineering Department.

Boers, J. W., & Kuiper, H. (1992). *Biological metaphors and the design of modular artificial neural networks*. umt, Leiden University, The Netherlands.

Branke, J. (1995). *Evolutionary algorithms for neural network design and training* (Technical Report). Karlsruhe, Germany: Institute AIFB, University of Karlsruhe.

Brill, F. Z., Brown, D. E., & Martin, W. N. (1990). *Genetic algorithms for feature selection for counterpropogation networks* (Tech. Rep. No. IPC-TR-90-004). Charlottesville: University of Virginia, Institute of Parallel Computation.

Brotherton, T. W., & Simpson, P. K. (1995). Dynamic feature set training of neural nets for classification. In McDonnell, J. R., Reynolds, R. G., & Fogel, D. B. (Eds.), *Evolutionary Programming IV* (pp. 83–94). Cambridge, MA: MIT Press.

Caudell, T. P., & Dolan, C. P. (1989). Parametric connectivity: Training of constrained networks using genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 370–374). San Mateo, CA: Morgan Kaufmann.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. New York, NY: John Wiley & Sons.

Fodor, I. K., Cantú-Paz, E., Kamath, C., & Tang, N. (2000). Finding bent-double radio galaxies: A case study in data mining. In *Interface: Computer Science and Statistics*, Volume 33.

Grönross, M. A. (1998). *Evolutionary design of neural networks*. Unpublished master's thesis, University of Turku.

Gruau, F. (1992). Genetic synthesis of boolean neural networks with a cell rewritting developmental process. In Whitley, D., & Schaffer, J. D. (Eds.), *Proceedings of the International Workshop on Com-*

binations of Genetic Algorithms and Neural Networks (pp. 55–74). Los Alamitos, CA: IEEE Computer Society Press.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, *4*(4), 461–476.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 179–186). San Francisco, CA: Morgan Kaufmann.

Marshall, S. J., & Harrison, R. F. (1991). Optimization and training of feedforward neural networks by genetic algorithms. In *Proceedings on the Second International Conference on Artifical Neural Networks and Genetic Algorithms* (pp. 39–43). Springer Verlag.

Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379–384). San Mateo, CA: Morgan Kaufmann.

Montana, D. J., & Davis, L. (1988). *Training feedforward neural networks using genetic algorithms.* unpublished manuscript.

Nolfi, S., Elman, J. L., & Parisi, D. (1994). *Learning and evolution in neural networks* (Tech. Rep. No. 94-08). Rome, Italy: Institute of Psychology, National Research Council.

Odewahn, S., Stockwell, E., Pennington, R., Humphreys, R., & Zumach, W. (1992). Automated star/galaxy discrimination with neural networks. *The Astronomical Journal*, *103*(1), 318–331.

Radcliffe, N. J. (1990). *Genetic neural networks on MIMD computers.* Unpublished doctoral dissertation, University of Edinburgh, Scotland.

Roberts, S. G., & Turenga, M. (1995). Evolving neural network structures. In Pearson, D., Steele, N., & Albrecht, R. (Eds.), *International Conference on Genetic Algorithms and Neural Networks* (pp. 96–99). New York: Springer-Verlag.

Schaffer, J. D. (1994). Combinations of genetic algorithms with neural networks or fuzzy systems. In Zurada, J. M., Marks, II, R. J., & Robinson, C. J. (Eds.), *Computational Intelligence Imitating Life* (pp. 371–382). New York, NY: IEEE Press.

Siddiqi, A. A., & Lucas, S. M. (1998). A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *Proceedings of 1998 IEEE Iternational Conference on Evolutionary Computation* (pp. 392–397). Piscataway, NJ: IEEE Service Center.

Skinner, A., & Broughton, J. Q. (1995). Neural networks in computational material science: training algorithms. *Modelling and Simulation in Material Science and Enginnering*, *3*, 371–390.

Storrie-Lombardi, M., Lahav, O., Sodre, L., & Storrie-Lombardi, L. (1992). Morphological classification of galaxies by artificial neural networks. *Mon. Not. R. Astron. Soc.*, *259*, 8–12.

Thierens, D., Suykens, J., Vandewalle, J., & Moor, B. D. (1991). Genetic weight optimization of a feedforward neural network controller. In *Proceedings of the Conference on Neural Nets and Genetic Algorithms* (pp. 658–663). Springer Verlag.

White, R. L., Becker, R., Helfand, D., & Gregg, M. (1997). A catalog of 1.4 GHz radio sources from the FIRST survey. *Astrophysical Journal*, *475*, 479.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 391–397). San Mateo, CA: Morgan Kaufmann.

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, *14*, 347–361.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447.

Alignment of Protein Structures with a Memetic Evolutionary Algorithm

B. Carr, W. Hart*
Sandia National Laboratories
PO Box 5800, MS1110
Albuquerque NM87185
USA

N.Krasnogor, J. Hirst, E. Burke†
ASAP Group &
Computational Biophysics Group
University of Nottingham
Nottingham, NG72RD
United Kingdom

J. Smith‡
Intelligent Computer System Centre
University of the West of England
Bristol, BS161QY
United Kingdom

Abstract

CATEGORY: Real-World Applications

Structural comparison of proteins is a core problem in modern biomedical research. Identifying structural similarities is essential for the assessment of the relationship between structure and function in proteins, and structural comparison techniques play a key role in applications like rational drug design. In this paper we consider a technique for protein structure comparison known as the maximum contact map overlap problem. In this problem, the similarity between two protein structures is computed by aligning the proteins to maximize the number of shared contacts in their corresponding contact maps.

We present a new approach to this problem that uses a Multimeme evolutionary algorithm. The best solution found by our algorithm provides a lower bound on the value of the optimal structural alignment between the proteins. We have evaluated the Multimeme algorithm on a range of benchmark problems and compared with previous heuristics. We apply a linear programming method, which provides an upper bound, to assess the accuracy of our solutions. Our experiments show that the Multimeme evolutionary algorithm represents a significant improvement on the current state of the art in metaheuristics for this problem.

*{rdcarr,wehart}@sandia.gov
† {natalio.krasnogor,jhirst}@nottingham.ac.uk , ekb@cs.nottingham.ac.uk
‡james.smith@uwe.ac.uk

1 Introduction

Structural comparison of proteins is a central task in biomedical research. Identifying structural similarities can provide significant insights into the relation between structure and function in proteins. Reliable and efficient structural matching plays a key role in rational drug design and in assessing the quality of structure prediction methods. A variety of structure comparison methods have been developed, such as SCOP [14], DALI [6], and LGA [17, 18]. However, no one technique has proven robust across a wide range of applications.

One of the emerging approaches for solving this problem is to evaluate the *alignment (or overlap) of contact maps* between proteins [6, 8, 11]. In its simplest form, a contact map is a matrix of all pairwise distances within a protein's components [12, 7]; these components can be atoms, residues, etc, depending on the resolution of the model employed. The distances in a contact map typically are computed by considering either the distance between the C_α atoms in a pair of residues, or the minimum distance between *any* two atoms belonging to those residues. Thus a contact map provides a simple representation of a protein's native three dimensional structure.[1]

In this paper we reconsider the use of metaheuristics for the Contact Map Overlap (*Max CMO*) problem [11]. For this problem, the distances in the contact map are discretized to zero or one, depending on whether the pairwise distances between residues are within a specified threshold. Although this discretization would seem to be easier than aligning matrices with real values, the problem is in fact NP-complete [4, 5, 9]. We have previously proposed a

[1] A protein's native state is associated with its minimal free energy configuration. The biological function of a protein is achieved in this state.

rigorous approach to *Max CMO* [11]. This approach employs an integer programming (IP) formulation for *Max CMO*, which is solved using a branch-and-cut algorithm. The branch-and-cut algorithm uses a Linear Programming (LP) relaxation of the IP to produce the upper bounds, and a Genetic Algorithm (GA) is used to provide lower bounds at the branch nodes.

The aim of the present research is to investigate the use of more sophisticated evolutionary algorithms: Multimeme memetic evolutionary algorithms [9], which integrate multiple local search strategies with a standard evolutionary search. We employ the LP relaxation of the *Max CMO* IP to provide upper bounds on the quality of the alignment of two proteins' structures, and thus we can empirically evaluate the quality of the solutions that we generate. Further, we compare the results of the Multimeme algorithm with a standard GA as well as the LGA protein structure comparison algorithm.

2 The Maximum Contact Map Overlap Problem

2.1 0-1 Contact Maps

Although contact maps are generally represented as distance matrices, one way of simplifying this representation of a protein's structure is to define a contact as a pair of residues that are closer than a given threshold, θ. Typically, θ ranges between 2 and 9 Angstroms. This gives a 0-1 contact map, where the matrix has the form

$$S_{i,j} = \begin{cases} 1 & \text{if residue i and j are within distance } \theta \\ 0 & \text{otherwise} \end{cases}$$

The advantage of this representation is that structural properties of proteins can be more easily visualized and compared [16, 15]. Figure 1 is the graphic representation of the 0-1 contact map for protein 1C7W shown in Figure 2(a).[2] In this figure, the α-helices are represented by wide bands along the main diagonal, while β-sheets manifest themselves as bands parallel or perpendicular to the diagonal.[3]

A 0-1 contact map can also be represented as an undirected graph. In this graph, each residue is a node and there exists an edge between nodes i and j if these residues are in contact (i.e. if $S_{i,j} = 1$). Figure 2

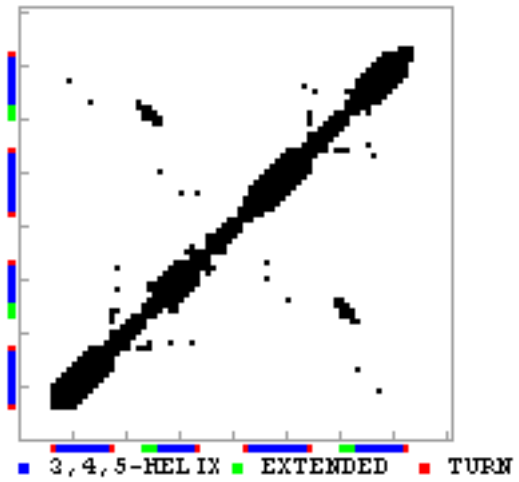


Figure 1: A 0-1 contact map comparing protein 1C7W with itself.

shows the native structures of two proteins, and Figure 3 shows the graphs corresponding to their contact maps. Note the long range interactions of residues that are far away in the sequence but close in the three dimensional structure adopted by the native state.

2.2 Problem Formulation

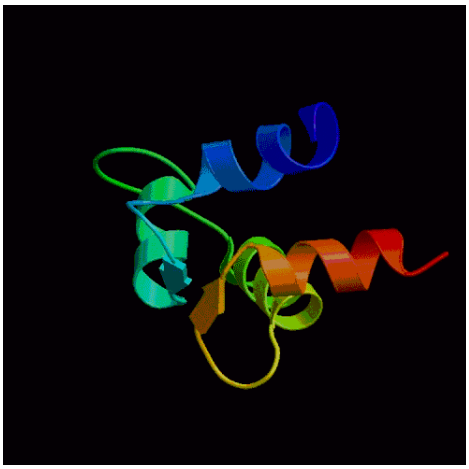
The *alignment* between two contact maps is an assignment of residues in the first contact map to residues on the second contact map. Residues that are thus aligned are considered equivalent. Further, consider a pair of contacts, one from each protein. We say that such a pair of contacts is equivalent if the pairs of residues that define the end-points of these contacts are equivalent. In the *Max CMO* problem, the value of an alignment between a pair of proteins is the number of equivalent contacts between these proteins. This number is called the *overlap* of the contact maps and the goal is to maximize this value. The *Max CMO* problem was first discussed by Godzik et al. [3], and it has been proven NP-complete [5, 9].

Lancia et al. [11] describe an IP approach for the MAX CMO problem, which builds upon a polynomial reduction from *Max CMO* to Maximum Independent Set (MIS). The size of the converted instances is the product of the number of contacts of the two maps (around 10000 nodes for a pair of proteins of 100 residues each). To solve MIS instances of this size, the authors exploit specific characteristics of the MIS instances.

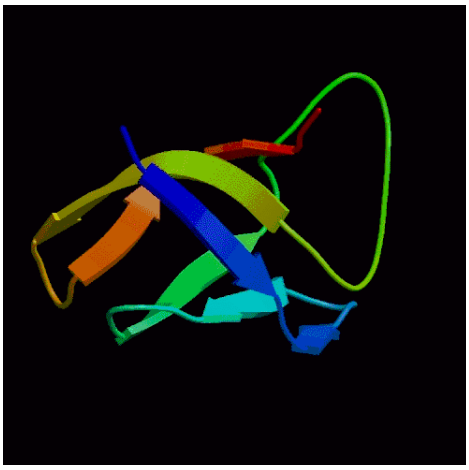
Let $G_1 = (E_1, V_1)$ and $G_2(E_2, V_2)$ be the two graphs that correspond to two 0-1 contact maps, where E_i are the edges in these graphs and V_i the vertices. The IP

[2] The proteins used in this paper are taken from the Protein Data Bank [1], and the labels we use are the labels provided by the PDB.

[3] $\alpha-$helices and $\beta-$sheets are elements of a protein's secondary structure. See [2] for a description of protein secondary structure.



(a)



(b)

Figure 2: Ribbon representation of structures for proteins (a) 1C7W and (b) 1NMG. Arrow shows $\beta-$strand and spiral depicts helix.

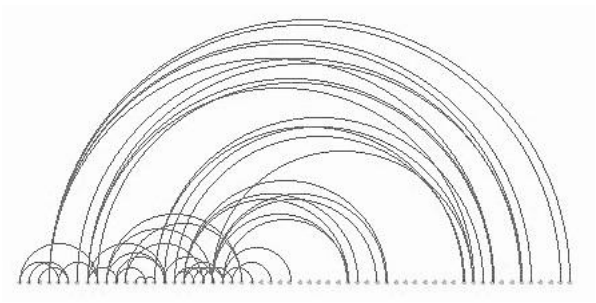
formulation proposed by Lancia et al. is:

$$\max \sum_{e \in E_1, f \in E_2} y_{e,f}$$

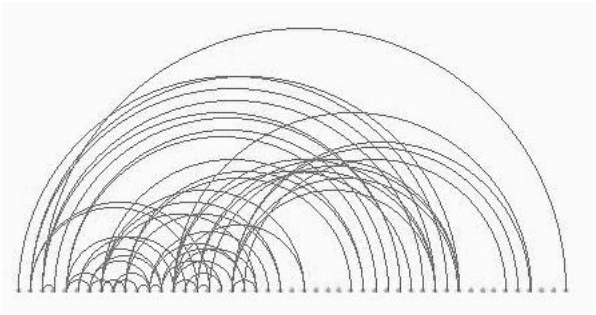
subject to the constraints

$$\begin{array}{ll}
x_{i,u} + x_{j,v} \leq 1 & \forall (i,u),(j,v) \text{ crossed} \\
\sum_{i<j} y_{(i,j)(u,v)} \leq x_{i,u} & \forall i \in V_1, (u,v) \in E_2 \\
\sum_{i>j} y_{(j,i)(u,v)} \leq x_{i,v} & \forall i \in V_1, (u,v) \in E_2 \\
\sum_{u<v} y_{(i,j)(u,v)} \leq x_{i,u} & \forall u \in V_2, (i,j) \in E_1 \\
\sum_{u>v} y_{(i,j)(v,u)} \leq x_{j,u} & \forall u \in V_2, (i,j) \in E_1 \\
x, y \in \{0,1\} &
\end{array}$$

The binary variable $x_{i,u}$ for $i \in V_1$ and $u \in V_2$ has a value of 1 if i is aligned with u and 0 otherwise. The binary variable $y_{e,f}$ has a value of 1 if the edges e, f are shared in a feasible solution and 0 otherwise.



(a)



(b)

Figure 3: Graphical representation of the contact maps of proteins (a) 1C7W and (b) 1NMG.

Hence the first equation is the statement of the goal of maximizing the shared edges (contacts). We say that (i,u) and (j,v) are *crossed* if both of these assignments are not feasible within a single alignment; these form crossed assignment lines in the alignment graphs below.

Lancia et al. [11] discuss the solution of this IP with a branch-and-cut algorithm. Note that if the last constraint in the IP is removed then this problem is an LP, so it can be solved in polynomial time. Further, the solution to this relaxation of the IP provides an upper bound on the globally optimal solution of the IP. These LP solutions are a critical element of the branch-and-cut algorithm described by Lancia et al. Further, they can be used to benchmark heuristic solvers like the EA we describe in the next section.

3 Multimeme Algorithms

Memetic algorithms [13] are evolutionary algorithms that include, as part of the "standard" evolutionary cycle of crossover-mutation-selection, a local search stage. They have been extensively used and studied on a wide range of problems. Multimeme evolutionary algorithms are introduced in Krasnogor et al. [10, 9]. The distinction between memetic and Multimeme al-

gorithms is the use of a family of local searchers. A memetic algorithm employs a single local search heuristic, while a Multimeme algorithm relies on a set of simple local searchers. Multimeme algorithms self-adaptively select which heuristic to use for different instances, stages of the search or individuals in the population.

In a Multimeme algorithm an individual is composed of its genetic material (that represents the solution to the problem being solved) and its memetic material (that defines the kind of local searcher to use). The mechanisms of genetic exchange and variation are the usual crossover and mutation operators but tailored for the specific problem one wants to solve. Memetic transmission is done during crossover as follows. If the two parents use the same local searcher then the offspring will inherit that local searcher. However, if the local searchers are different then the offspring inherits the one associated with the fittest parent. Otherwise (the heuristics used by both parents are different but the fitnesses are the same) a random choice between both local searchers is made.

The rational behind this criterion is to propagate local searchers that are associated with fit individuals (as it is hoped that those individuals were improved by their respective local searchers). Also, during mutation, the meme of an individual can be overridden and a local searcher assigned at random (uniformly from the set of all available local searchers) with the probability specified by the innovation rate parameter.

4 A Multimeme Algorithm for *Max CMO*

We extend here the work on the *Max CMO* initiated by Lancia et.al. [11], who employed a standard GA with specially tailored genetic operators. We briefly describe those operators and explain how we enlarged that set for use in our Multimeme approach.

In a GA for *Max CMO* a chromosome is represented by a vector c of dimension n, for which each position can take values in the $[-1, \ldots, m-1]$ domain. Here, m is the length of the longer protein and n the length of the shorter. A position j in c, $c[j]$, specifies that the j^{th} residue in the longer protein is aligned to the $c[j]^{th}$ residue in the shorter. A value of -1 in that position will signify that residue j is not aligned to any of the residues in the other protein. Unfeasible configurations are not allowed, that is, if $i < j$ and $v[i] > v[j]$ or $i > j$ and $v[i] < v[j]$ (e.g. a crossing alignment) then the chromosome is discarded. That is, our algorithms work only with feasible solutions. It is simple to define genetic operators that preserve feasibilities based on this representation. Two-point crossover with boundary checks was used to mate individuals and create one offspring. Although both parents are feasible valid alignments, the newly created offspring can result in invalid (crossed) alignments. After constructing the offspring, feasibility is restored by deleting any alignment that crosses other alignments. Figure 4 shows a two point crossing over with an unfeasible intermediate offspring. At the later stage it is repaired and completed, i.e. all unassigned vertices are randomly assign to a vertex on the other protein if no new violations are produced (not shown in the picture).

The mutation move employed in the experiments is called a sliding mutation. It selects a consecutive region of the chromosome vector and adds, slides right, or subtracts, slides left, a small number. The phenotypic effect produced is the tilting of the alignments. In Figure 5 an example is shown. Again, alignments that violate the feasibility of the solution are discarded. Lancia et al. [11] describes a few variations on the sliding mutation.

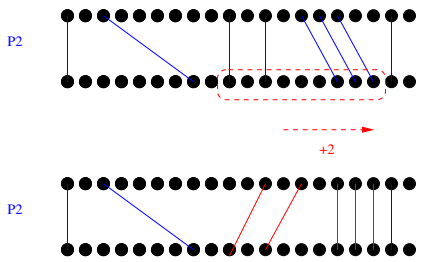


Figure 5: Sliding mutation under the vector representation for *Max CMO*. In this example a window size of 9 residues was chosen together with a right sliding by 2 residues.

In this paper we employ a Multimeme algorithm that, besides using the same mutation and crossover as the mentioned GA, has a set of 6 local search operators. Four of the local searchers implemented are parameterized variations of the sliding operator. The direction of movement, left or right sliding, and the tilting factor, i.e. the number added or subtracted, were chosen at random in each local search stage. The size of the window was taken from the set $\{2, 4, 8, 16\}$. Two new operators were also defined: a "wiper" move and a "split" move. The wiper move is depicted in Figure 6. At every iteration of the operator two alignments, represented by x and y in the lower protein of the picture, are chosen. The feasible regions of alignment for x and y are determined (marked with dotted line rectangles $R1$ and $R2$ in the graph). Subsequently all the residues within those regions are tested as candidate

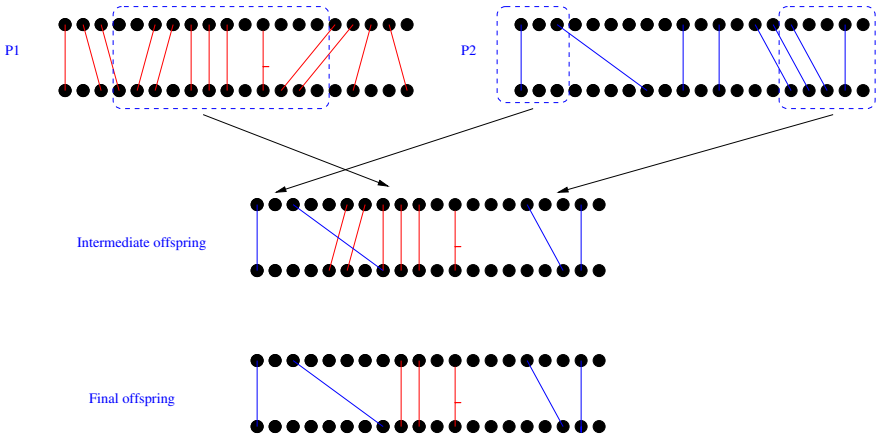


Figure 4: Two-point crossover with boundary checks for a vector representation of *Max CMO*.

alignments for x and y. The best alignment is chosen. In the graph this is represented by the vertices that are end points of the upper contact edge.

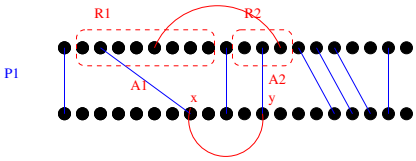


Figure 6: Wiper move under the vector representation for *Max CMO*. Two alignments of the lower protein are selected and tested exhaustively against all the possible feasible and compatible configurations around them.

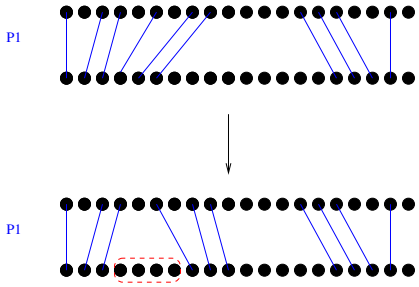


Figure 7: Split operator under the vector representation for *Max CMO*. This operator splits regions of consecutive alignments.

During our investigations, it became evident that some sort of redistribution of consecutive alignments might be beneficial. We implemented a split operator to accomplish this. The split move, depicted in Figure 7, tries to rearrange regions of consecutive alignments. In the example, the first section of six consecutive alignments is broken into two regions of three alignments each. Note that the end points of the alignments are not changed in contrast with the sliding and wiper moves.

5 Experiments and Results

In order to evaluate our Multimeme algorithm we first implemented a GA, following as closely as possible the GA described by Lancia et al. [11].[4] We were able to reproduce Lancia et al.'s results and, although we found a small improvement of the final-values in our implementation, they were minor and we consider both GA's implementations to be very similar.

[4]Some extra experimental details were kindly given to us in private communications with the authors.

The GA used a population of size 300. The mutation rate was 0.15 per individual and crossover probability was set to 0.75. Fitness proportional selection was used to select the mating pool. An elitist (elite set size of 1) $(300, 300)$ selection strategy was employed. These parameters were selected after an initial assessment of parameter values with a few pairs of proteins. The Multimeme algorithm used the same basic GA setting and parameters, but also employed as memes the four variations of the sliding move, a split move and a wiper move as described in the previous section. The probability of local search was set to one, i.e., local search was applied to every individual in every generation. Each meme was iterated two times (short local searches). The values of mutation and crossover probabilities were not optimized for the Multimeme code but as mentioned before, taken from the GA setting. The innovation rate for the Multimeme algorithm was 1.0 and 0.15 (see below).

We performed two experiments on a set of 18 pairs of protein structures from the Protein Data Bank [1]. For these 18 pairs we had the upper bound values ob-

Instance	GA	Multimeme IR=1.0	LP
1a8o-1f22	25	23	28
1avy-1bct	19	22	25
1b6w-1bw5	23	23	24
1bct-1bw5	20	17	20
1bct-1f22	20	18	22
1bct-1ilp	18	18	23
1c7v-1c7w	62	62	62
1c9o-1kdf	29	29	40
1df5-1f22	21	22	27
1hlh-1hrf	19	21	24
1hlh-1nmf	22	22	27
1kst-2new	20	22	26
1nmf-2new	23	23	27
1nmg-1wdc	18	19	23
1pfn-1svf	16	16	16
1utr-1wdc	16	24	28
1vnb-1bhb	17	21	27
2new-3mef	21	19	26

Table 1: Maximum Contact Map Overlap values for several protein pairs. A GA, a Multimeme algorithm with innovation rate 1.0 are compared. The value of the LP results are also displayed to the right.

Instance	GA	Multimeme IR=0.15	LP
1a8o-1f22	25	25	28
1avy-1bct	22	22	25
1b6w-1bw5	23	24	24
1bct-1bw5	17	20	20
1bct-1f22	16	21	22
1bct-1ilp	18	19	23
1c7v-1c7w	62	62	62
1c9o-1kdf	31	34	40
1df5-1f22	24	24	27
1hlh-1hrf	20	22	24
1hlh-1nmf	22	23	27
1kst-2new	22	23	26
1nmf-2new	23	25	27
1nmg-1wdc	18	19	23
1pfn-1svf	16	16	16
1utr-1wdc	26	26	28
1vnb-1bhb	19	23	27
2new-3mef	23	22	26

Table 2: Maximum Contact Map Overlap values for several protein pairs. A GA, a Multimeme algorithm with innovation rate = 0.15 compared. The value of the LP results are also displayed to the right.

tained by the LP formulation described earlier. It is important to remark that, the LP gives estimations (i.e. upper bounds) on the possible maximum objective value for a particular instance of the problem. It does not produce (explicit) solutions to the problem instances.

The metric used in the experiments was the value of best alignment obtained out of 5 runs for each pair of proteins.

In the first experiment we assessed the performance of a Multimeme algorithm with an innovation rate set to 1 in a relatively fast experiment. For both the Multimeme and the GA the maximum number of function evaluations was $3 * 10^6$. The results are presented in Table 1.

From the table we can see that the two algorithms produce the same results in 7 cases, the GA outperforms the Multimeme in four cases and the Multimeme outperforms the GA in 7 cases. We can thus say that the Multimeme algorithm with innovation rate of 1.0 generates similar or better results than the GA (both algorithms using the same number of fitness evaluations) in 14 out of 18 cases.

The second experiment was meant to test the behavior of both the GA and the Multimeme algorithms in the same set of 18 pairs of proteins but employing more fitness evaluations, $5 * 10^6$ in this case. Also the innovation rate was reduced to 0.15. The alignment values obtained are presented in Table 2.

From inspection of the table, and comparing it with the previous one, we can see that both algorithms profit from longer runs. However, the difference between the two approaches is more noticeable in this case. Out of 18 protein pairs the GA outperformed the Multimeme in just one case, instance 2new-3mef, as opposed to four in Table 1. The Multimeme produced better results in 11 cases while for the remaining pairs, 6 instances, the values obtained with both algorithms were equivalent.

The Multimeme algorithm was able to match 4 of the optimum bounds produced by the LP. In the 4 instances where the GA and the Multimeme achieve similar results, i.e. pairs 1a8o-1f22, 1avy-1bct, 1df5-1f22 and 1utr-1wdc, the values obtained are below the LP bounds. However, we speculate that actually those alignments, i.e. the ones produced with the metaheuristics, are indeed optimal and that the LP program is able to obtain higher values by using fractional solutions that cannot possibly have physical meaning. Also, it is important to note that the gap between the Multimeme results and the LP bounds is in all cases smaller than 4 except in the case of the pair 1c9o-1kdf for which the gap is 6.

Other experiments were performed with different genetic operators, like DPX crossover and different mutation moves, but the results were not particularly better than the ones discussed here; hence they are omitted.

6 Comparison with LGA

In the previous section we verified that the Multimeme algorithm introduced in this paper produces optimal and almost optimal, i.e. with respect to the LP bounds, results. In this section we assess whether the alignments generated for CMO are qualitatively similar to other well known methods of structural alignment. To accomplish this aim, we will compare our alignments with those obtained with LGA [18, 17]. The later is a state of the art, publicly available program for the comparative analysis of protein structures.

LGA can be run in two modes, protein sequence aware mode and sequence independent mode. The former is suitable when the two proteins to be compared have the same number of residues and the later for the case when the two proteins are not necessarily of the same length. We use LGA in the sequence independent mode as the illustrative comparison we run was made with proteins of different size. The parameters used to run the LGA program were $-4\ -sia\ -o1\ -d\_6.5$. Please refer to Zemla [17] for details. The pair of proteins studied was 1c9o and 1kdf (from the Protein Data Bank). This pair is the one that produces the biggest gap between the solutions returned by the Multimeme algorithm and the LP upper bound[5]. Protein 1c9o is a cold shock protein from the genome of *Bacillus Caldolyticus* and 1kdf is an antifreeze protein from *Macrozoarses Americanus*. Because the functions are similar, it is expected that the structures of the two will have some resemblance and that either algorithm (LGA or the Multimeme) will be able to capture it.

Figure 8 plots the alignments obtained by our method and the LGA program. Axis X and Y are indexed by residues id, where X represents the residues of protein 1c9o and Y that of 1kdf. A mark, circle or square, in coordinates (x, y) should be interpreted as the alignment of residue number x in 1c9o with residue y in 1kdf. The closer to the diagonal the full alignment is the more similar are the structures. As it is possible to see from the graph there is only one mismatched region between the two alignments, the area between residue 14 and 20. In that window the difference between the two alignment is considerable. In the rest of the protein the two algorithms produce strikingly similar alignments where some perfect matching regions are visible. The overall shape of both alignments is also similar. To elucidate which of the two algorithms calculates the best alignment in the region of discrepancy (i.e better preserves secondary structure features like beta sheets, alpha helices, etc), we carried out a secondary structure analysis in this region. The analysis performed allows us to conclude that both algorithms produced results of similar quality as the proteins differ substantially on their secondary structure contents for the region studied.

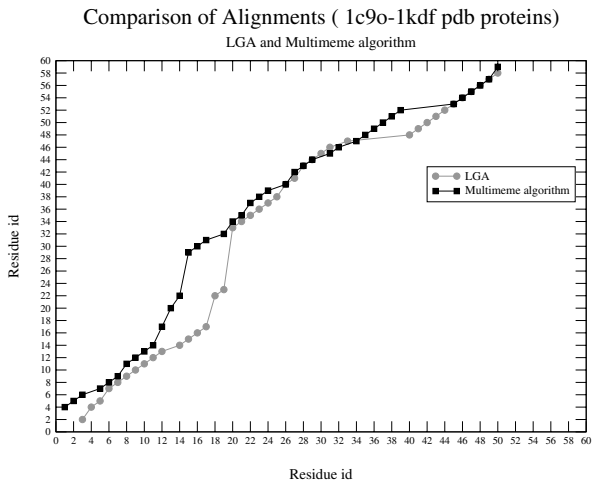


Figure 8: Comparison of the structural alignments obtained by LGA and the Multimeme algorithm for proteins 1c9o and 1kdf

7 Conclusion and Future Work

In this paper we reproduced the results of Lancia et al. for the *Max CMO* problem [11]. Their results provide the first application of a GA for this problem. We used a Multimeme algorithm with an architecture similar to that used in Krasnogor et al. [9, 10] to obtain results that improve over those produced by the standard GA. No exhaustive testing of parameters for the Multimeme algorithm was carried out, but rather the same setting as those produced for the GA were employed. Furthermore, our method gives results that are compatible with those obtained with a state of the art structure comparison algorithm [17, 18]. One immediate advantage of our method over, e.g., LGA is that being a population based approach it can potentially return not only one "best alignment" but a variety of alternative alignments. Moreover, these set of candidate alignments can be analyzed for biological

[5] It is a worst case comparison as it represents our poorest result.

relevance at a later stage by a human expert.

As an immediate follow up of this work a much larger set of protein pairs is being analyzed and the biological significance of the alignments obtained with our method will be assessed on those pairs. A Master-Worker parallel version of the LP-Multimeme integrated approach is under development. That platform will enable one to perform genome scale structures comparisons.

Acknowledgments We thank B. Walenz for his aid at several stages of this project and C. Papadimitriou for sending us several reprints. We also thank M. Collins for help performing the LP calculations. We acknowledge the funding of the British research council BBSRC (Bioinformatics Initiative, ref:42/BIO14458). This work was performed in part at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

[1] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissing, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[2] T.E. Creighton. *PROTEINS: Structures and Molecular Properties*. W.H. Freeman and Company, Publishers, 1993 (2nd edition).

[3] A. Godzik, J. Skolnick, and A. Kolinski. A topology fingerprint approach to inverse protein folding problem. *Journal of Molecular Biology*, 227:227–238, 1992.

[4] D. Goldberg. Phd thesis. *Department of Computer Sciences, UC Berkeley*, 2000.

[5] D. Goldman, S. Istrail, and C. Papadimitriou. Algorithmic aspects of protein structure similarity. *Proceedings of the 40th Annual Symposium on Foundations of Computer Sciences*, pages 512–522, 1999.

[6] L. Holm and C. Sander. Protein-structure comparison by alignment of distance matrices (DALI). *Journal of Molecular Biology*, 233:123–138, 1993.

[7] C. Hue-Sun and K.A. Dill. Origins of structure in globular proteins. In *Proc. Natl. Acad. Sci. USA*, volume 87, pages 6388–6392, August 1990.

[8] R.K. Kincaid. A molecular structure matching problem. *Computers Ops Res.*, 24:25–35, 1997.

[9] N. Krasnogor. *Studies on the Theory and Design Space of Memetic Algorithms*. Ph.D. Thesis, Faculty of Engineering, Computer Science and Mathematics. University of the West of England. Bristol, United Kingdom. http://dirac.chem.nott.ac.uk/~natk/Public/, 2002.

[10] N. Krasnogor and J.E. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2001.

[11] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. *Proceedings of The Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001*, 2001.

[12] S. Lifson and C. Sander. Antiparallel and parallel beta-strands differ in amino acid residue preferences. *Nature*, 282:109–11, 1979.

[13] P. A. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Caltech, Pasadena, California, 1989.

[14] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[15] I.N. Shindyalov and P.E. Bourne. WPDB a PC-based tool for analyzing protein structures. *Journal of Applied Crystallography*, 28:847–852, 1995.

[16] E.L.L. Sonnhammer and J.C. Wooton. Dynamic contact maps of protein structures. *Journal of Molecular Graphics and Modelling*, 16:1–5, 1998.

[17] A. Zemla. LGA program: A method for finding 3-D similarities in protein structures. *http://PredictionCenter.llnl.gov/local/lga*, 2000.

[18] A. Zemla, C. Venclovas, J. Moult, and K. Fidelis. Processing and analysis of CASP3 protein structure predictions. *PROTEINS: Structure, Function, and Genetics Suppl*, 3:22–29, 1999.

A genetic algorithm with sequential niching for discovering small-disjunct rules

Deborah R. Carvalho

Pontificia Universidade Católica do Paraná (PUCPR)
Postgraduate program in applied computer science
R. Imaculada Conceicao, 1155. Curitiba – PR
80215-901. Brazil

Universidade Tuiuti do Paraná (UTP)
Computer Science Dept.
Av. Comendador Franco, 1860. Curitiba-PR
80215-090 Brazil
deborah@ipnet.com.br

Alex A. Freitas

Pontificia Universidade Católica do Paraná (PUCPR)
Postgraduate program in applied computer science
R. Imaculada Conceicao, 1155. Curitiba – PR
80215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex
Tel./Fax: (55) (41) 330-1669

Abstract

This work addresses the well-known classification task of data mining. In this context, small disjuncts are classification rules covering a small number of examples. One approach for coping with small disjuncts, proposed in our previous work, consists of using a decision-tree/genetic algorithm method. The basic idea is that examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm (C4.5), while examples belonging to small disjuncts are classified by a genetic algorithm (GA) designed for discovering small-disjunct rules. In this paper we follow this basic idea, but we propose a new GA which consists of several major modifications to the original GA used for coping with small disjuncts. The performance of the new GA is extensively evaluated by comparing it with two versions of C4.5, across several data sets, and with several different sizes of small disjuncts.

1 INTRODUCTION

This paper addresses the well-known classification task of data mining (Hand, 1997). In this task, the discovered knowledge is often expressed as a set of rules of the form: IF <conditions> THEN <prediction (class)>.

This knowledge representation has the advantage of being intuitively comprehensible for the user, and it is the kind of knowledge representation used in this paper.

From a logical viewpoint, typically the discovered rules are in disjunctive normal form, where each rule represents a disjunct and each rule condition represents a conjunct. A small disjunct can be defined as a rule which covers a small number of training examples (Holte et al., 1989).

In general rule induction algorithms have a bias that favors the discovery of large disjuncts, rather than small disjuncts. This preference is due to the belief that it is better to capture generalizations rather than specializations in the training set, since the latter are unlikely to be valid in the test set (Danyluk & Provost, 1993).

Hence, at first glance, small disjuncts are not important, since they tend to be error prone. However, small disjuncts are actually quite important in data mining. The main reason is that, even though each small disjunct covers a small number of examples, the set of all small disjuncts can cover a large number of examples. For instance (Danyluk & Provost, 1993) report a real-world application where small disjuncts cover roughly 50% of the training examples. In such cases we need to discover accurate small-disjunct rules in order to achieve a good classification accuracy rate.

One approach for coping with small disjuncts, proposed in our previou work (Carvalho & Freitas 2000a, 2000b), consists of using a decision-tree/genetic algorithm method. The basic idea is that examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm (C4.5), while examples belonging to small disjuncts are classified by a genetic algorithm (GA) designed for discovering small-disjunct rules.

In this paper we follow this basic idea, but we propose a new GA which consists of several major modifications to the original GA proposed for coping with small disjuncts.

The rest of this paper is organized as follows. Section 2 describes the hybrid decision tree/genetic algorithm

method proposed in our previous work. This section assumes that the reader is familiar with decision trees, a well-known kind of data mining algorithm. Section 3 describes the new GA proposed in this paper for discovering small-disjunct rules. In that section we explain the motivation for the design of this new GA and describe in detail the major modifications that it introduces, by comparison with original GA used in our hybrid method. Section 4 reports the results of extensive experiments evaluating the performance of the proposed method. Finally, section 5 concludes the paper.

2 THE BASIC HYBRID DECISION-TREE / GENETIC-ALGORITHM METHOD FOR RULE DISCOVERY

We have previously proposed a hybrid method for rule discovery that combines decision trees and genetic algorithms (GAs) (Carvalho & Freitas, 2000a; 2000b). The basic idea is to use a decision-tree algorithm to classify examples belonging to large disjuncts and use a GA to discover rules classifying examples belonging to small disjuncts. Decision-tree algorithms have a bias towards generality that is well suited for large disjuncts, but not for small disjuncts. On the other hand, GAs are robust, flexible algorithms which tend to cope well with attribute interactions (Dhar et al, 2000), (Freitas, 2001; 2002), and can be more easily tailored for coping with small disjuncts.

The method discovers rules in two training phases. In the first phase it runs C4.5, a well-known decision tree induction algorithm (Quinlan, 1993). The induced, pruned tree is transformed into a set of rules (or disjuncts). Each of these rules is considered either as a small disjunct or as a "large" (non-small) disjunct, depending on whether or not its coverage (the number of examples covered by the rule) is smaller than or equal to a given threshold.

The second phase consists of using a GA to discover rules covering the examples belonging to small disjuncts. In the previous version of our method, each run of the GA discovers rules classifying examples belonging to a separated small disjunct. In this paper we introduce a major modification of this phase: all small disjuncts are grouped together into a single training set and given to the GA, so that a single run of the GA discovers rules classifying examples belonging to the total set of small-disjunct examples. This new approach (as well as the motivation for it) will be described in the next section.

Before we move to the next section, however, we review in the following the main characteristics of our previous GA (Carvalho & Freitas 2000a), hereafter called GA-Small (standing for GA with Small training set), in order to make this paper self-contained. Hereafter the new GA introduced in this paper will be called GA-Large-SN (standing for GA with Large training set and with Sequential Niching), since it not only uses a larger training set but also uses a sequential niching method, as will be described later.

In GA-Small each individual represents the antecedent (IF part) of a small-disjunct rule. The consequent (THEN part) of the rule, which specifies the predicted class, is not represented in the genome. Rather, it is fixed for a given GA-Small run, so that all individuals have the same rule consequent during all that run.

Each run of GA-Small discovers a single rule (the best individual of the last generation) predicting a given class for examples belonging to a given small disjunct. Since it is necessary to discover several rules to cover examples of several classes in several different small disjuncts, GA-Small is run several times for a given dataset. More precisely, one needs to run GA-Small $d * c$ times, where d is the number of small disjuncts and c is the number of classes to be predicted. For a given small disjunct, the k-th run of GA-Small, $k = 1,...,c$, discovers a rule predicting the k-th class.

The genome of an individual consists of a conjunction of conditions composing a given rule antecedent. Each condition is an attribute-value pair, as shown in Figure 1. In this figure A_i denotes the i-th attribute and Op_i denotes a logical/relational operator comparing A_i with one or more values V_{ij} belonging to the domain of A_i, as follows. If attribute A_i is categorical (nominal), the operator Op_i is "*in*", which will produce rule conditions such as "A_i *in* $\{V_{i1},...,V_{ik}\}$", where $\{V_{i1},...,V_{ik}\}$ is a subset of the values of the domain of A_i. By contrast, if A_i is continuous (real-valued), the operator Op_i is either "≤" or ">", which will produce rule conditions such as "$A_i \leq V_{ij}$", where V_{ij} is a value belonging to the domain of A_i. Each condition in the genome is associated with a flag, called the active bit B_i, which takes on the value 1 or 0 to indicate whether or not, respectively, the i-th condition is present in the rule antecedent (phenotype). This allows GA-Small to use a fixed-length genome (for the sake of simplicity) to represent a variable-length rule antecedent (phenotype).

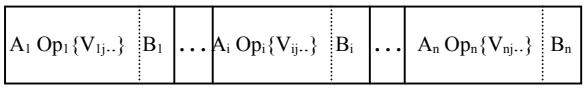


Figure 1: Structure of the genome of an individual.

For a given GA-Small run, the genome of an individual consists of n genes (conditions), where $n = m - k$, m is the total number of predictor attributes in the dataset and k is the number of ancestor nodes of the decision tree leaf node identifying the small disjunct in question. Hence, the genome of a GA-Small individual contains only the attributes that were *not* used to label any ancestor of the leaf node defining that small disjunct.

To evaluate the quality of an individual GA-Small uses the fitness function:

$$\text{Fitness} = (TP / (TP + FN)) * (TN / (FP + TN)) \quad (1)$$

where TP, FN, TN and FP – standing for the number of true positives, false negatives, true negatives and false positives – are well-known variables often used to evaluate the performance of classification rules – see e.g. (Hand, 1997). In formula (1) the term (TP / (TP + FN)) is

usually called sensitivity (*Se*) or true positive rate, whereas the term (TN / (FP + TN)) is usually called specificity (*Sp*) or true negative rate. These two terms are multiplied to foster the GA to discover rules having both high *Se* and high *Sp*.

GA-Small uses tournament selection, with tournament size of 2 (Michalewicz, 1996). It also uses standard one-point crossover with crossover probability of 80%, and mutation probability of 1%. Furthermore, it uses elitism with an elitist factor of 1 – i.e., the best individual of each generation is passed unaltered into the next generation.

GA-Small also includes an operator especially designed for simplifying candidate rules. The basic idea of this rule-pruning operator is to remove several conditions from a rule to make it shorter. This operator is applied to every individual of the population, right after the individual is formed.

Unlike the usually simple operators of GA, GA-Small's rule-pruning operator is an elaborate procedure based on information theory (Cover & Thomas, 1992). Hence, it can be regarded as a way of incorporating a classification-related heuristic into a GA for rule discovery. The heuristics in question is to favor the removal of rule conditions with low information gain, while keeping the rule conditions with high information gain. In other words, the larger information gain of a rule condition has the smaller probability of removing that condition from the rule – see (Carvalho & Freitas, 2000a; 2000b) for details.

Once all the *d* * *c* runs of GA-Small are completed, examples in the test set are classified. For each test example, the system pushes the example down the decision tree until it reaches a leaf node. If that node is a large disjunct, the example is classified by the decision tree algorithm. Otherwise the system tries to classify the example by using one of the *c* rules discovered by the GA for the corresponding small disjunct. If there is no small-disjunct rule covering the test example it is classified by a default rule, which predicts the majority class among the examples belonging to the current small disjunct. If there are two or more rules discovered by the GA covering the test example, the conflict is solved ty using the rule with the largest fitness (on the training set) to classify that example.

3 THE EXTEND HYBRID DECISION-TREE / GENETIC-ALGORITHM METHOD FOR RULE DISCOVERY

In the previous section we have reviewed GA-Small, the GA algorithm for discovering small disjunct rules previously proposed as part of our hybrid decision-tree/GA method. The two main limitations of that GA are:

(a) Each run of GA-Small has access to a very small training set, consisting of just a few examples belonging to a single leaf node of a decision tree. Intuitively, this makes it difficult to induce reliable classification rules in some cases.

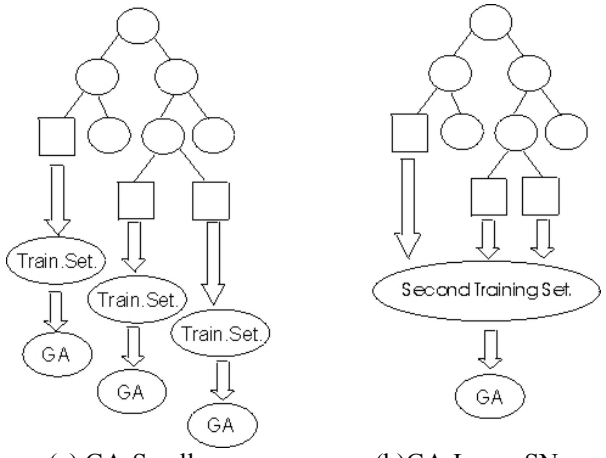
(b) Although each run of the GA is relatively fast (since it uses a small training set), the hybrid method as a whole has to run the GA many times (since the number of GA-Small runs is proportional to the number of small disjuncts and the number of classes). Hence, the hybrid C4.5/GA-Small method turns out to be considerably slower than the use of C4.5 alone.

These two limitations were our motivation to develop a new GA for discovering small disjunct rules. We stress that in this paper we propose just a new GA, without modifying the decision-tree algorithm of the above-mentioned hybrid method.

By comparison with GA-Small, the new GA proposed in this paper – denoted GA-Large-SN, as mentioned above – involves five major modifications. These modifications are described in the detail in the next subsections.

3.1 INCREASING THE CARDINALITY OF THE TRAINING SET

In our new GA-Large-SN, all the examples belonging to all the leaf nodes considered small disjuncts are grouped in a single training set, called the "second training set" (to distinguish it from the original training set used by C4.5 to build the decision tree). This second training set is provided as input data for the GA. This is the most important characteristic of GA-Large-SN, and it is the basis for the other characteristics discussed below.



(a) GA-Small (b)GA-Large-SN

Figure 2: Differences in the training sets of the GAs

This characteristic of GA-Large-SN is illustrated in Figure 2(b), where one can clearly see that all small disjuncts are grouped into a single, relatively large training set. This is in sharp contrast with the approach used by GA-Small (described in section 2), illustrated in Figure 2(a), where one can clearly see that each small disjunct is used as a small training set.

3.2 USING A NICHING METHOD TO FOSTER THE DISCOVERY OF MULTIPLE RULES

As a result of the above-discussed increase in the cardinality of the training set, one needs to discover several rules to cover the examples of each class. (Recall that this was not the case with the approach described in section 2, since in that approach it was assumed that a GA-Small run had to discover a single rule for each class.) Therefore, in our new GA-Large-SN it is essential to use some kind of niching method, in order to foster population diversity and avoid its convergence to a single rule. In this work we use a sequential niching method (Beasley et al., 1993). We chose this kind of method for two reasons. First, its simplicity. Second, and most important, it does not require the specification of additional parameters for its execution, unlike well-known niching methods such as fitness sharing (Goldberg & Richardson, 1987) and crowding (Mahfoud, 1995).

```
BEGIN
/* TrainingSet-2 contains all examples belonging to all small disjuncts */
  RuleSet = ∅;
  build TrainingSet-2;
  WHILE cardinality(TrainingSet-2) > 5
     run the GA;
     add the best rule found by the GA to RuleSet;
     remove from TrainingSet-2 the examples
        correctly covered by that best rule;
  END-WHILE
END-BEGIN
```

Figure 3: GA with sequential niching for discovering small disjunct rules

The pseudo-code of our GA with sequential niching is shown, at a high level of abstraction, in Figure 3. It starts by initializing the set of discovered rules (denoted RuleSet) with the empty set and building the second training set (denoted TrainingSet-2), as explained above. Then it iteratively performs the following loop. First, it runs the GA, using TrainingSet-2 as the training data for the GA. The best rule found by the GA (i.e., the best individual of the last generation) is added to RuleSet. Then the examples correctly covered by that rule are removed from TrainingSet-2, so that in the next iteration of the WHILE loop TrainingSet-2 will have a smaller cardinality. An example is "correctly covered" by a rule if the example's attribute values satisfy all the conditions in the rule antecedent and the example belongs to the same class as predicted by the rule. This process is iteratively performed while the number of examples in TrainingSet-2 is greater than 5. (It is assumed that when the cardinality of TrainingSet-2 is smaller than 5 there are too few examples to allow the discovery of a reliable classification rule.)

It should be noted that the sequential niching method used in this work is a variation of the one proposed by (Beasley et al., 1993). The latter actually requires the specification of a parameter, associated with a distance metric, for modifying the fitness landscape according to the location of solutions found in previous iteractions. In order to implement this parameter, the author uses an Euclidian distance.

By contrast, there is no need for this kind of parameter in our version of sequential niching. In order to avoid that the same search spaced be explored several times, the examples that are correctly covered by the discovered rules are removed from the training set. Hence, the nature of the fitness landscape is automatically updated as rules are discovered along different iterations of the sequential niching method.

3.3 MODIFICATION OF THE METHOD USED TO DETERMINE A RULE'S CONSEQUENT

Each run of GA-Large-SN still discovers a single rule, and a rule's consequent (the class predicted by the rule) is not encoded into the genome, like in the GA-Small described in section 2. However, unlike GA-Small, in GA-Large-SN the consequent of each rule is not fixed upfront for all rules (individuals) in the population. Rather, the consequent of each rule is dynamically chosen as a function of the rule's antecedent. More precisely, a rule's consequent is chosen as the most frequent class in the set of examples covered by that rule's antecedent.

3.4 A NEW HEURISTICS FOR RULE PRUNING

The GA-Large-SN proposed in this paper uses a new heuristic measure for rule pruning. This measure is based on the idea of using the decision tree built by C4.5 to compute a classification accuracy rate for each attribute, according to how accurate were the classifications performed by the decision tree paths in which that attribute occurs. That is, the more accurate were the classifications performed by the decision tree paths in which a given attribute occurs, the higher the accuracy rate associated with that attribute, and the smaller the probability of removing that a condition with attr ibute from a rule. The computation of an accuracy rate for each attribute is performed by the procedure shown in Figure 4.

The computation of the accuracy rate associated with each attribute is performed as follows. For each attribute A_i, the algorithm checks each path of the decision tree built by C4.5 in order to determine whether or not A_i occurs in that path. (The term path is used here to refer to each complete path from the root node to a leaf node of the tree.) For each path p in which A_i occurs, the algorithm computes two counts, namely the number of examples classified by the rule associated with path p, denoted $\#Classif(A_i,p)$, and the number of examples *correctly* classified by the rule associated with path p, denoted $\#CorrClassif(A_i,p)$.

```
BEGIN
  Count_of_Unused_Attr = 0;
  FOR each attribute A_i, i=1,...,m
    IF attribute A_i occurs in at least one path in the tree
      THEN compute the accuracy rate of A_i, denoted Acc(A_i) (see text);
      ELSE increment Count_of_Unused_Attr by 1;
    END-IF
  END-FOR
  Min_Acc = the smallest accuracy rate among all attributes
            that occur in at least one path in the tree;
  FOR each of the attributes A_i, i=1,...,m, such that A_i
      does not occur in any path in the tree
    Acc(A_i) = Min_Acc / Count_of_Unused_Attr;
  END-FOR
                m
  Total_Acc = Σ Acc(A_i) ;
               i=1
  FOR each attribute A_i, i=1,...,m
    Compute the normalized accuracy rate of A_i,
      denoted Norm_Acc(A_i), as:
      Norm_Acc(A_i) = Acc(A_i) / Total_Acc ;
  END-FOR
END-BEGIN
```

Figure 4: Computation of each attribute's accuracy rate, for rule pruning purposes

where Z_i is the number of decision tree paths where attribute A_i occurs. Note that formula (2) is used only for attributes that occur in at least one path of the tree. All the attributes that do not occur in any path of the tree are assigned the same value of $Acc(A_i)$, and this value is determined by the formula:

$$Acc(A_i) = Min\_Acc \,/\, Count\_of\_Unused\_Attr \,, \qquad (3)$$

where *Min_Acc* and *Count_of_Unused_Attr* are determined as shown in Figure 4.

Finally, the value of $Acc(A_i)$ for every attribute A_i, i=1,...,m, is normalized by dividing its current value by *Total_Acc*, which is determined as shown in Figure 3.

Once the normalized value of accuracy rate for each attribute A_i, *denoted* $Norm\_Acc(A_i)$, has been computed by the procedure of Figure 4, it is directly used as a heuristic measure for rule pruning. The basic idea here is the same as the basic idea of the rule pruning procedure mentioned in section 2. In that section, where the heuristic measure was the information gain, it was mentioned that the larger the information gain of a rule condition, the smaller the probability of removing that condition from the rule. In GA-Large-SN, we replace the information gain of a rule condition with $Norm\_Acc(A_i)$, the normalized value of the accuracy rate of the attribute included in the rule condition. Hence, the larger the value of $Norm\_Acc(A_i)$, the smaller the probability of removing the i-th condition from the rule. The remainder of the rule pruning procedure proposed in (Carvalho & Freitas 2000a) remains essentially unaltered.

Note that the accuracy rate-based heuristic measure for rule pruning proposed here effectively exploits information from the decision tree built by C4.5. Hence, it can be considered as a kind of hypothesis-driven measure, since it is based on a hypothesis (in our case, a decision tree) previously constructed by a data mining algorithm.

By contrast, the previously-mentioned information gain-based heuristic measure does not exploit such information. Rather, it is a measure whose value is computed directly from the training data, independent of any data mining algorithm. Hence, it can be considered as a kind of data-driven measure.

3.5 INCREASING THE GENOME LENGTH

Recall that in GA-Small (reviewed in section 2) the genome contained only the attributes which were *not* used to label any ancestor of the leaf node defining the small disjunct being processed by the GA. That approach made sense because GA-Small was using as the training set only the examples belonging to a single leaf node. Clearly, the attributes in the ancestor nodes of that leaf node were not useful to distinguish between classes of examples in the leaf node, since all those examples had the same values for those attributes.

However, the situation is different in the case of the new GA-Large-SN proposed in this paper. Now the training set of the GA consists of all the examples belonging to all the leaf nodes that are considered small disjuncts – i.e., all those examples are effectively mixed into a single training set. Hence, the above notion of "attributes in the ancestor nodes of a single leaf node" is not meaningful any more. Therefore, in GA-Large-SN the genome contains m genes, where m is the number of attributes of the data being mined. I.e., all attributes can occur in the rule represented by an individual, so that in theory a rule can contain at most m conditions in its antecedent. Of course, in practice the number of conditions in a rule will be much smaller than m, due to the use of the above-discussed rule pruning operator.

4 COMPUTATIONAL RESULTS

We have evaluated the performance of GA-Large-SN across eight public-domain data sets of the the well-known data repository of the UCI (University of California at Irvine), available at:

http://www.ics.uci.edu/~mlearn/MLRepository.html.

The examples that had some missing value were removed from these data sets. In the Adult data set we have used the predefined division of the data set into a training and a test set. In the Connect data set we have randomly partitioned the data into a training and a test set with 47290 and 20267 examples, respectively. In the other datasets we have run a well-known 10-fold cross-validation procedure, which essentially works as follows. The data set is randomly partitioned into 10 mutually-exclusive and exhaustive partitions. Then the classification algorithm is run 10 times. In the i-th run, i = 1,...,10, the i-th partition is used as the test set, and the remaining nine partitions are grouped and used as the

training set. After the 10 runs are over, the reported accuracy rate is the average accuracy rate over all those 10 runs.

In our experiments we have used a commonplace definition of small disjunct, based on a fixed threshold of the number of examples covered by the disjunct. The definition is: "A decision-tree leaf is considered a small disjunct if and only if the number of examples belonging to that leaf is smaller than or equal to a fixed size *S*."

In order to better evaluate the performance of GA-Large-SN, it is important to compare it against other classification method(s). In particular, we wanted to compare the hybrid system against another method that induces rules or trees (which can be straightforwardly converted to rules). In this case the kind of knowledge representation used by the systems being compared is the same, and the difference in the results will reflect mainly differences in search strategies. Hence, we can compare the evolutionary search strategy of GA-Large-SN against the local, greedy search strategy of a rule induction or decision tree algorithm.

Within this spirit we report the results of experiments comparing our hybrid C4.5/GA-Large-SN system with two other classification methods. The first is C4.5 alone, which is used to classify all examples – i.e., both large-disjunct examples and small-disjunct examples. The second is a "double run" of C4.5, hereafter called "double C4.5" for short. The later is a new way of using C4.5 to cope with small disjuncts, as follows.

The main idea of our "double C4.5" is to build a classifier running twice the algorithm C4.5. The first run considers all examples in the original training set, producing a first decision-tree. Once all the examples belonging to small disjuncts have been identified by this decision tree, the system groups all those examples into a single example subset, creating the "second training set", as described above for GA-Large-SN (see Figure 2(b)). Then C4.5 is run again on this second, reduced training set, producing a second decision tree. In other words, the second run of C4.5 uses as training set exactly the same "second training set" used by GA-Large-SN. This makes the comparison between GA-Large-SN and "double C4.5" very fair.

In order to classify a new example, the rules discovered by both runs of C4.5 are used as follows. First, the system checks whether the new example belongs to a large disjunct of the first decision tree. If so, the class predicted by the corresponding leaf node is assigned to the new example. Otherwise (i.e., the example belongs to one of the small disjuncts of the first decision tree), the new examples are classified by the second decision tree.

The motivation for this more elaborated use of C4.5 was an attempt to create a simple algorithm that was more effective in coping with small disjuncts.

Recall that the hybrid C4.5/GA-Large-SN method has an important parameter, namely the small-disjunct size threshold (*S*). In order to evaluate how robust the method is with respect to this parameter, we have done experiments with four different values of *S*, namely 3, 5, 10 and 15. For each of these four *S* values, we have done ten different experiments, varying the random seed used to generate the initial population of individuals. The results reported below, for each value of *S*, are an arithmetic average of the results over these ten different experiments. Therefore, the total number of experiments is 40 (4 values of *S* * 10 different random seeds). In addition, recall that each of these 40 experiments actually consists of a 10-fold cross-validation run for most data sets (with the exception of the Adult and Connect data sets, where a single division of the data into training and test sets was used).

Each run of GA-Large-SN is relatively fast, so that each of these 40 experiments took a processing time on the order of six minutes for the biggest data set, Connect, and for the largest value of *S* (15), on a Pentium III with 192Mb of RAM.

We now report results comparing the classification accuracy rate (on the test set) of the proposed hybrid C4.5/GA-Large-SN with C4.5 alone (Quinlan, 1993) and with the above-described "double C4.5". We have used C4.5's default parameters. In each GA-Large-SN run the population has 200 individuals, and the GA is run for 50 generations.

Table 1: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for *S* = 3

Data set	C4.5	C4.5(2)	C4.5/GA		
Connect	72.60 (0.3)	**78.06 (0.3)**	77.86 (0.1)	+	-
Adult	78.62 (0.3)	81.19 (0.3)	**85.45 (0.1)**	+	+
Crx	91.79 (2.1)	92.57 (1.2)	**93.69 (1.2)**		
Hepatitis	80.78(13.3)	78.95 (6.9)	**89.25 (9.5)**		
House-votes	93.62 (3.2)	**97.32 (2.4)**	97.18 (2.5)		
Segmentation	**96.86 (1.1)**	76.62 (2.8)	81.46 (1.1)	-	+
Wave	75.78 (1.9)	68.18 (3.7)	**83.86 (2.0)**	+	+
Splice	65.68 (1.3)	55.65 (6.0)	**70.62 (8.6)**		+

The results are shown in Tables 1, 2, 3 and 4 referring to *S* values of 3, 5, 10 and 15, respectively. In these tables the first column indicates the data sets. The second column shows the accuracy rate on the test set achieved by C4.5 alone, classifying both large-disjunct and small-disjunct examples. The third column reports the accuracy rate for C4.5(2). The fourth column reports the accuracy rate achieved by our hybrid C4.5/GA-Large-SN system, using C4.5 to classify large-disjunct examples and our GA classify small-disjunct examples. The values between brackets are standard deviations. For each data set, the highest accuracy rate among the three classifiers is shown in bold.

In addition, in the fourth column we indicate, for each data set, whether or not the accuracy rate of C4.5/GA-Large-SN is significantly different from the accuracy rates of the other two methods. More precisely, the cases where the accuracy rate of C4.5/GA-Large-SN is significantly better (worse) than the accuracy rate of each of the other two methods is indicated by the "+" ("-") symbol. A difference between two methods is deemed significant when the corresponding accuracy rate intervals (taking into account the standard deviations) do not overlap.

Let us now analyze the results of Tables 1, 2, 3 and 4 starting with Table 1 (where S = 3). In this table C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 5 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. In addition, C4.5/GA-Large-SN is significantly better than C4.5(2) in 4 data sets, and the reverse is true in only 1 data set.

Table 2: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for S = 5

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	77.09 (0.3)	**77.85 (0.2)** + +
Adult	78.62 (0.3)	79.27 (0.3)	**85.50 (0.2)** + +
Crx	91.79 (2.1)	92.03 (1.0)	**93.06 (1.6)**
Hepatitis	80.78(13.3)	75.67 (17.1)	**89.48 (9.7)**
House-votes	93.62 (3.2)	93.54 (3.9)	**97.44 (2.9)**
Segmentation	**96.86 (1.1)**	74.49 (3.4)	80.41 (1.0) - +
Wave	75.78 (1.9)	65.59 (4.4)	**85.31 (2.4)** + +
Splice	65.68 (1.3)	57.45 (8.7)	**70.44 (7.8)**

In Table 2 (where S = 5) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 7 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 4 data sets, and the reverse is not true in any data set.

Table 3: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for S = 10

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	76.19 (0.3)	**76.95 (0.1)** + +
Adult	78.62 (0.3)	76.06 (0.3)	**80.04 (0.1)** + +
Crx	**91.79 (2.1)**	90.78 (1.2)	91.66 (1.8)
Hepatitis	80.78(13.3)	82.36 (18.7)	**95.05 (7.2)**
House-votes	93.62 (3.2)	89.16 (8.0)	**97.65 (2.0)**
Segmentation	**96.86 (1.1)**	72.93 (5.5)	78.68 (1.1) -
Wave	75.78 (1.9)	64.93 (3.9)	**83.95 (3.0)** + +
Splice	65.68 (1.3)	61.51 (6.6)	**70.70 (6.3)**

In Table 3 (where S = 10) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 6 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 3 data sets, and the reverse is not true in any data set.

In Table 4 (where S = 15) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 6 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 3 data sets and the reverse is not true in any data set.

Table 4: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for S = 15

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	74.95 (0.3)	**76.01 (0.3)** + +
Adult	78.62 (0.3)	74.29 (0.3)	**79.32 (0.2)** + +
Crx	**91.79 (2.1)**	90.02 (0.8)	90.40 (2.4)
Hepatitis	80.78(13.3)	66.16 (19.1)	**82.52 (7.0)**
House-votes	93.62 (3.2)	88.53 (8.4)	**95.91 (2.3)**
Segmentation	**96.86 (1.1)**	73.82 (5.8)	77.11 (1.9) –
Wave	75.78 (1.9)	65.53 (4.0)	**82.65 (3.7)** + +
Splice	65.68 (1.3)	64.35 (4.7)	**70.62 (5.5)**

We can summarize the results of the above four tables as follows.

- C4.5/GA-Large-SN outperformed C4.5 alone in 87.50% of the data sets for S = 3 and S =5, and in 75% for S = 10 and S = 15.
- C4.5/GA-Large-SN outperformed C4.5(2) in 75% of the data sets for S = 3, and in 100% for S = 5, S = 10 and S = 15.
- Considering the results of the three methods for every data set and every value of S, the best accuracy rate was obtained by C4.5/GA in 78.2% of the cases, by C4.5 alone in 15.6% of the cases, and by C4.5(2) in only 6,2% of the cases.

Finally, a brief comment on computational time is appropriate here. We have mentioned, at the beginning of section 3, that one of our motivations for designing GA-Large-SN was to reduce processing time, by comparison with GA-Small. We have done an experiment comparing the processing time of both GA-Large-SN and GA-Small, on the same machine, in the same data set, namely the Connect data set – which is the largest data used in the above-reported experiments with GA-Large-SN. We observed that GA-Large-SN takes about only 12% of the processing time of GA-Small in the Connect data set.

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper we have described a new hybrid decision-tree/GA (C4.5/GA-Large-SN) method. The GA component of this method, called GA-Large-SN, consists of major modifications of the original GA (here called GA-Small) proposed by (Carvalho & Freitas 2000a), as discussed throughout section 3.

We have compared the new hybrid C4.5/GA-Large-SN system with 2 algorithms based on the use of C4.5 alone, namely: (a) the default version of C4.5; (b) a "double run of C4.5", which uses the same training set as GA-Large-SN. This comparison was performed across four different values for a parameter defining the size of a small disjunct.

Overall, the hybrid C4.5/GA-Large-SN obtained considerably better accuracy rates than both above-mentioned versions of C4.5 alone, in all the four definitions of small-disjunct size used in this paper.

In this paper we have focused on comparing the performance of the hybrid C4.5/GA-Large-SN with the performance of C4.5 alone, since C4.5 is a very well-known algorithm that is often used in comparison with other algorithms in the literature. In future research we also intend to compare the predictive accuracy of the rules discovered by C4.5/GA-Large-SN with the predictive accuracy of the rules discovered by the GA alone.

References

D.Beasley, D.R.Bull, R.R.A.Martin, (1993). Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation 1(2),* 101-125. MIT Press

D.R.Carvalho, A.A.Freitas (2000a). A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in Data Mining. *Proc 2000 Genetic and Evolutionary Computation Conf. (Gecco-2000), 1061-1068*. Las Vegas, NV, USA. July.

D.R.Carvalho, A.A.Freitas (2000b). A genetic algorithm-based solution for the problem of small disjuncts. Principles of Data Mining and Knowledge Discovery (*Proc. 4th European Conf., PKDD-2000*. Lyon, France). Lecture Notes in Artificial Intelligence 1910, 345-352. Springer-Verlag.

T.M.Cover, J.A.Thomas (1991) *Elements of Information Theory*. John Wiley&Sons

V.Dhar, D.Chou, F.Provost (2000) Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid With Entropy Reduction. *Data Mining and Knowledge Discovery 4(4)*, 251-280. Oct. 2000.

A.P.Danyluk, F.J.Provost (1993). Small Disjuncts in Action: Learning to Diagnose Erros in the Local Loop of the Telephone Network, *Proc. 10th International Conference Machine Learning*, 81-88.

A.A.Freitas (2001) Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3)*, Nov. 2001, pp. 177-199.

A.A.Freitas (2002) Evolutionary Algorithms. Chapter of forthcoming *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2002.

D.E.Goldberg, J.Richardson (1987) Genetic Algorithms with Sharing for Multimodal Function Optimization. *ICGA-87*. 41-49.

D.J.Hand (1997). *Construction and Assessment of Classification Rules*, John Wiley & Sons.

R.C.Holte, L.E.Acker, B.W.Porter (1989). Concept Learning and the Problem of Small Disjuncts, *Proc. IJCAI – 89*, 813-818.

S.W.Mahfoud (1995). *Niching Methods for Genetic Algorithms*, Illigal Report N. 95001, Thesis Submitted for degree of Doctor of Philosophy in Computer Science. University of Illinois at Urbana-Champaign.

Z.Michalewicz (1996) *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd Ed. Springer-Verlag.

J.R.Quinlan (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publisher.

G.M.Weiss, H.A.Hirsh (2000). Quantitative Study of Small Disjuncts, *Proc. of Seventeenth National Conference on Artificial Intelligence.* Austin, Texas, 665-670.

Symbolic Regression in Design of Experiments: A Case Study with Linearizing Transformations

Flor A. Castillo **Ken A. Marshall** **Jim L. Green** **Arthur K. Kordon**

The Dow Chemical Company
Freeport, TX 77541
USA

Abstract

The paper presents the potential of genetic programming (GP)-generated symbolic regression for linearizing the response in statistical design of experiments when significant Lack of Fit is detected and no additional experimental runs are economically or technically feasible because of extreme experimental conditions. An application of this approach is presented with a case study in an industrial setting at The Dow Chemical Company.

1 INTRODUCTION

The complexity of some industrial chemical processes requires that first-principle or mechanistic model be considered in connection with empirical models. At the basis of empirical models is that underlying any system there is a fundamental relationship between the inputs and the outputs that can be locally approximated over a limited range of experimental conditions by a polynomial or a linear regression model.

Suitable statistical techniques such as design of experiments (DOE) are available to assist in this process (Box *et al*, 1978). The capability of the linear model to represent the data can be assessed through a formal Lack of Fit (LOF) test when experimental replicates are available (Montgomery, 1999). Significant LOF in the model indicates a regression function that is not linear; i.e. the polynomial initially considered is not appropriate. A more adequate model may be found by fitting a polynomial of higher order by augmenting the original design with additional experimental runs. Specialized designs such as the Central Composite Design are available for this purpose (Box *et al.*, 1978).

However, there are many practical cases where runs are very expensive or technically unfeasible because of extreme experimental conditions, thus making the fit of a higher order polynomial impractical. This problem can be handled if appropriate input transformations are used, provided that the basic assumption of least-square estimation regarding the probability distributions of errors is not affected. These assumptions require that errors be uncorrelated and normally distributed with mean zero and constant variance.

Some useful transformations are discussed in Box and Draper.(1987). Unfortunately, transformations that linearize the response without affecting the error structure are not always obvious and are often developed based on experience or theoretical insight. Genetic programming (GP)- generated symbolic regression provides a unique opportunity to rapidly develop and test these transformations. Symbolic regression includes the finding of a functional mathematical expression that fits a given set of data (Koza, 1992).

GP-generated symbolic regression is an evolution-based algorithm for automatically generating nonlinear input-output models. Several possible models of the response as a function of the input variables are obtained by combining basic functions, inputs, and numerical constants. This multiplicity of solutions offers a rich set of possible transformations of the inputs. At the same time, the most significant challenge of GP-generated transforms is that most models are not parsimonious and include chunks of inactive code or terms that do not contribute to the overall fitness (Banzhaf *et al*, 1998) and that may prove inefficient in producing a linearizing transformation. This problem can be managed to some degree at the expense of extra-computation time by appropriate algorithms that quickly test the ability of transforms to linearize the response without altering error structure.

The application of GP in DOE and the potential of combining them offer a unique set of opportunities that is beginning to grab the attention of researchers and industry. Experimental design techniques have already been used to evaluate the effects of GP parameters (Spoonger, 2000). An excellent discussion of algorithm-driven regression based on genetic programming for solving supersaturated designs is presented in Cela *et.al* (2001).

In this paper, a novel approach of integrating GP with DOE is presented. This approach has the potential to improve the effectiveness of empirical

model building by saving time and resources in situations where experimental runs are quite expensive or technically unfeasible because of extreme experimental conditions. GP is applied to the development of variable transforms that linearize the response in statistically designed experiments for a chemical process in The Dow Chemical Company.

2 METHODOLOGY

A series of experimental runs were performed in a lab scale reactor in four variables. The response variable was the selectivity of one of the products. These experiments were statistically analyzed and the effect of the variables as well as a prediction of the response within the area of experimentation was well understood. LOF was induced by removing one experimental run to simulate a common situation in which LOF is significant and additional experimental runs are impractical due to the extreme cost of experimentation or because it is technically unfeasible due to extreme experimental conditions. In this system the potential of GP-generated transforms was studied allowing the comparison of results with a well-known system.

The appropriateness of GP-generated transforms to linearize the response without affecting error structure was assessed by performing the transformations presented in the functional form of the GP model. Then a linear regression model was fit in the transformed inputs. This model, referred to as the *transformed linear model*, was examined for Lack of Fit and appropriate error structure. Both models, the transformed linear model and the GP model, were tested considering 9 additional experiments in the region of the design. The validity of the results was determined by comparing model predictions with the previously analyzed experiments and with a fundamental kinetic model (FKM) that was earlier developed. The results indicate that GP-generated transformations have the potential of linearizing the response in those cases where additional experimental runs are not possible.

3 THE EXPERIMENTAL DESIGN

The experiments conducted in lab-scale thermal chlorination reactor system consisted of a complete 2^4 factorial design in the factors x_1, x_2, x_3, x_4, with three center points. A total of 19 experiments were performed. The response variable, S_k, was the yield or selectivity of one of the products. The factors were coded to a value of –1 at the low level, +1 at the high level, and 0 at the center point. The complete design in the coded variables is shown in Table 1

To develop a base case and test for variable transformations, LOF was induced by removing run number 1 of the experimental design. The response S_k, was fit to the following first-order linear regression equation

Table 1: 2^4 factorial design with three center points

RUNS	x_1	x_2	x_3	x_4	S_k
1	1	-1	1	1	1.598
2	0	0	0	0	1.419
3	0	0	0	0	1.433
4	-1	1	1	1	1.281
5	-1	1	-1	1	1.147
6	1	1	-1	1	1.607
7	-1	1	1	-1	1.195
8	1	1	1	-1	2.027
9	-1	-1	-1	1	1.111
10	-1	1	-1	-1	1.159
11	-1	-1	-1	-1	1.186
12	1	-1	-1	1	1.453
13	1	1	-1	-1	1.772
14	-1	-1	1	-1	1.047
15	-1	-1	1	1	1.175
15	1	1	1	1	1.923
17	1	-1	-1	-1	1.595
18	1	-1	1	-1	1.811
19	0	0	0	0	1.412

considering only terms that are significant at the 95% confidence level.

$$S_k = \beta_o + \sum_{i=1}^{k} \beta_i x_i + \sum\sum_{i<j} \beta_{ij} x_i x_j \qquad (1)$$

Table 2 shows the corresponding Analysis of Variance showing evidence of Lack of Fit ($p = 0.0476$). Therefore, the hypothesis that a first-order model can adequately describe this system is rejected.

Table 2 - Analysis of variance for the linear model

Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Model	8	1.5091186	0.188640	107.6350
Error	9	0.0157733	0.001753	**Prob > F**
C. Total	17	1.5248919		<.0001
Lack Of Fit				
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Lack Of Fit	7	0.01555519	0.002222	20.3775
Pure Error	2	0.00021810	0.000109	**Prob > F**
Total Error	9	0.01577329		0.0476
				Max RSq
				0.99

The corresponding residual plot, presented in Figure 1, suggested non-constant variance, which is one of the necessary conditions of the error structure for least-square estimation.

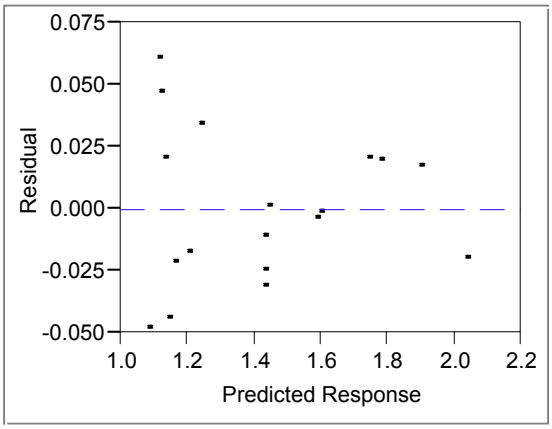


Figure 1 - Residual plot for first-order linear model suggesting non-constant variance

Under these circumstances, a variance-stabilizing power transformation of the response (y) was performed (Box and Cox, 1964). The response was transformed to y^{λ} where the parameter λ varies from –2 to 2 and the choice of λ that results in the minimum residual sum of squares of the transformed model is the maximum likelihood estimation of λ and the best transformation of the response. In the present case, however, the power transformation resulted in a λ value of 1 indicating that no transformation of the response was helpful. Cases like this are quite common in industrial processes. The next alternative to be investigated is the transformation of the input variables by means of GP-generated symbolic regression.

3.1 THE GP-GENERATED TRANSFORMATIONS

The GP approach will be used to search for potential transforms of the input variables. The GP algorithm was applied to the original data set, considering the response variable as the output and the four variables, x_1, x_2, x_3, x_4, in uncoded form as inputs. This resulted in a series of non-linear equations that satisfied the data. The functional form of these equations produced a rich set of possible transforms that were tested for the ability to linearize the response without altering error structure. An advantage of this approach is that experience or physical interpretation may be used to identify promising transforms, which were previously unavailable to the experimenter. An additional advantage is that GP generates a sensitivity analysis ranking all the input variables in order of importance to the fitness of the equations (Kordon and Smits, 2001) allowing to verify significant factors in the linearized models.

The GP algorithm is implemented as a toolbox in MATLAB. The initial functions for GP included: addition, subtraction, multiplication, division, square, change sign, square root, natural logarithm, exponential, and power. Function generation takes 20 runs with 500 population size, 100 number of generations, 4 reproductions per generation, 0.6 probability for function as next node, 0.01 parsimony pressure, and correlation coefficient as optimization criteria. A snapshot of the input/output sensitivity is shown in Figure 2, which shows x_1 as the most important input.

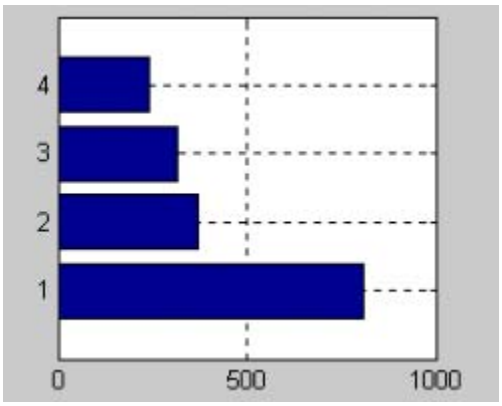


Figure 2 GP-based Input/output sensitivity of the four input variables

The selection of the best candidates is based on a trade-off between the fitness of the function and the ability to linearize the response while producing an acceptable error distribution. From the set of potential non-linear equations the best fit between model prediction and empirical response was found for the following analytical function:

$$S_k = \frac{3.13868\times10^{-17} e^{\sqrt{2x_1}} \ln\left[(x_3)^2\right] x_2}{x_4} + 1.00545 \quad (2)$$

Where x_1, x_2, x_3, x_4 are the input variables and S_k is the output.

The correlation coefficient between the analytical function and the empirical data was 0.95. This nonlinear equation indicates an exponential relationship with x_1, a logarithmic relationship with x_3, a linear relationship with x_2, and an inverse relationship with x_4, as shown in Table 3. To test the capability of these transforms to linearize the response, the following transformations were applied to the input variables as supplied by the GP function (2).

Table 3 - Variable transformations suggested by GP model

Original Variable	Transformed Variable
x_1	$Z_1 = \exp\left(\sqrt{2x_1}\right)$
x_2	$Z_2 = x_2$
x_3	$Z_3 = \ln[(x_3)^2]$
x_4	$Z_4 = x_4^{-1}$

Then a first-order linear regression model (i.e., the transformed linear model) was fit to the transformed variables. Table 4 shows the corresponding parameter estimates. The analysis of variance, presented in Table 5,shows no evidence of LOF indicating that the GP-generated transformations were succesful in linearizing the response.

Table 4 - Parameter estimates for transformed linear model

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	–0.29	0.0464	–6.26	0.0033
Z_1	0	1.3e-15	75.83	<.0001
Z_3	0.165	0.0079	20.80	<.0001
Z_4	0.147	0.0372	3.98	0.0164
Z_2	0.092	0.0070	13.17	0.0002
$(Z_1-7.e12)*(Z_3-3.2)$	0	2.7e-15	18.02	<.0001
$(Z_1-7.e12)*(Z_4-0.8)$	0	1.3e-14	7.58	0.0016
$(Z_3-3.199)*(Z_4-0.8)$	–0.701	0.08058	–8.70	0.0010
$(Z_1-7.e12)*(Z_2-3.9)$	0	2.4e-15	5.38	0.0058
$(Z_3-3.199)*(Z_2-3.9)$	0.050	0.01481	3.40	0.0274
$(Z_4-0.835)*(Z_2-3.9)$	0.180	0.06895	2.62	0.0590
$(Z_1-7.e12)*(Z_3-3.2)*(Z_2-3.9)$	0	5.1e-15	–5.04	0.0073
$(Z_1-7.e12)*(Z_4-0.8*(Z_2-3.9)$	0	2.4e-14	3.43	0.0264
$(Z_3-3.199)*(Z_4-0.8)*(Z_2-3.9)$	0.603	0.14952	4.04	0.0156

Table 5 - Analysis of variance for transformed linear model

Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Model	13	1.5241819	0.117245	660.5351
Error	4	0.0007100	0.000177	**Prob > F**
C. Total	17	1.5248919		<.0001
Lack Of Fit				
Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Lack Of Fit	2	0.00049190	0.000246	2.2554
Pure Error	2	0.00021810	0.000109	**Prob > F**
Total Error	4	0.00071000		0.3072
				Max RSq
				0.9999

The transformed linear model itself is less parsimonious than the nonlinear GP model including even third order iterations. However, the model is very significant.

The corresponding residual plot for the transformed linear model is presented in Figure 3. This plot indicates no violation regarding basic assumptions for the probability distribution of errors required by least squares, indicating that the GP-generated transformations linearized the response without altering the error structure of the model produced. One observation is that the residual of one center point is larger than the residuals of the other two center points. However, in the original analysis, this data point had also been excluded due to problems with experimental conditions during the run.

The transformed linear model and the nonlinear GP model were used to predict the selectivity to the output at the conditions of experiment 1 (the experiment removed from the original data set in order to induce Lack of Fit).

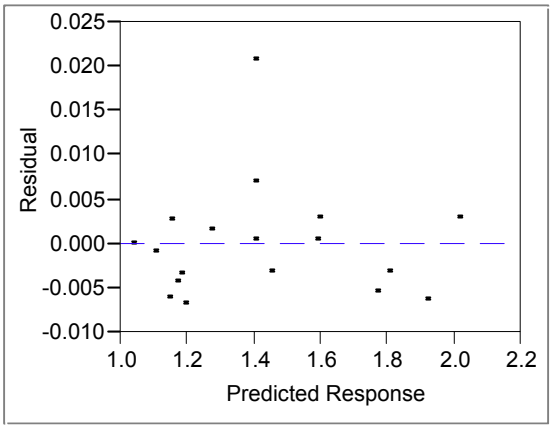


Figure 3 - Residual plot for the transformed linear model

Figure 4 shows the plot of predicted versus actual values for the two models.

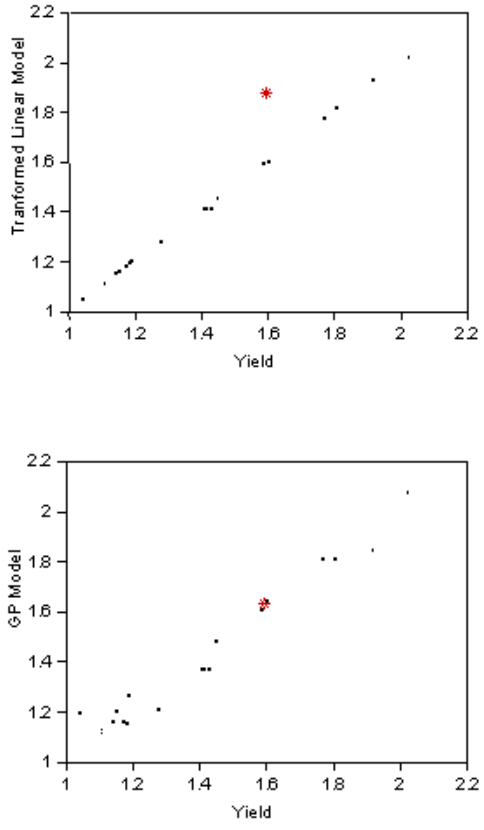


Figure 4 - Predicted versus actual values for the transformed linear and the nonlinear GP model

The point corresponding to experiment 1 is indicated in the figure by an asterisk. The performance of both models was very good. The correlation coefficient was 0.99 for the transformed linear model, and 0.978 for the GP model.

The nonlinear GP model gives a more accurate prediction for the value of the removed point. But both models predict an increase response by operating at conditions of high x_1, x_2, x_3, and low x_4. These results were consistent with the results obtained previously by analyzing the full design and by a fundamental kinetic model.

3.2 THE TESTING DATA SET

The prediction capabilities of the transformed linear model and the GP model were tested with nine additional experimental points within the range of experimentation. This is a relative small data set because of the cost and difficulty of experimentation. Plots of the predicted response for the transformed linear and the GP model versus the actual values, presented in Figure 5, indicate good performance of both models indicating that the models are comparable in terms of prediction with additional data inside the region of the design. The correlation coefficient was 0.99 for the transformed linear model, and 0.98 for the GP model. The selection of one of these models over the other would be driven by the requirements of a particular application. For example, in the case of process control, the more parsimonious model would generally be preferred.

4 CONCLUSIONS

In the course of conducting designed experiments, Lack of Fit is often encountered, indicating that the proposed linear regression model fails to adequately describe the data. One traditional approach to address this problem is to introduce higher order terms to the linear model. This is accomplished by adding experiments to the original design, which can be time-consuming, costly, or may be technically unfeasible because of extreme experimental conditions.

A second approach is to use transformations to avoid additional experimentation. One technique is transformation of the responses, but this is not always effective. In those cases, transformation of the input variables may be the only alternative to remove Lack of Fit and provide an appropriate model. Unfortunately these transformations are not always obvious and are often developed based on experience or theoretical insight. GP provides a way to rapidly develop and test these transformations so that those appropriate linear models are developed.

The genetic programming (GP) algorithm was successfully applied to the results of DOE in a chemical process in Dow Chemical Company. Experimentation in this system is difficult and time-consuming due to the severe conditions of the experiments. Data for the interested output were manipulated to induce Lack of Fit. Genetic programming was used to generate a nonlinear model for the output as a function of four experimental variables. The form of the nonlinear model was used to suggest input variable transformations for a linear model. The resulting transformed linear model showed no evidence of Lack of Fit. No additional experimental data had to be used in the analysis to achieve this result. The success of this industrial application illustrates the great potential of using GP to address Lack of Fit in linear regression problems. This approach can improve the effectiveness of empirical model building by saving time

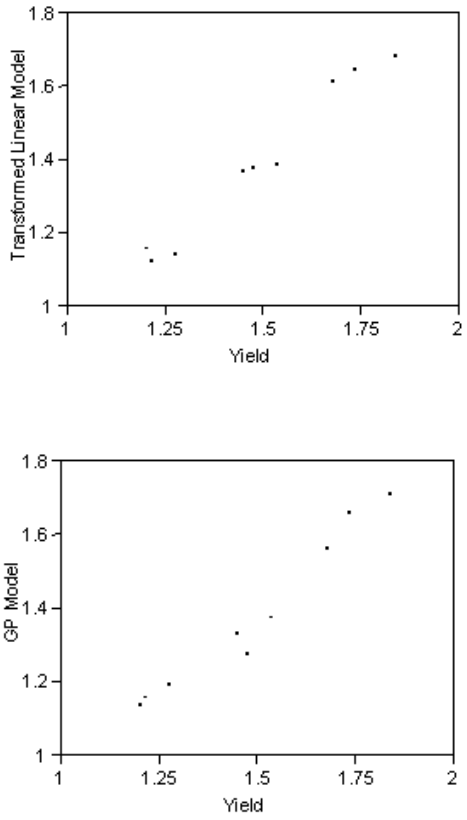


Figure 5 - Predicted versus actual values for additional data

and resources when experiments are expensive or difficult. However, more systematic research in the area of defining a methodology for robust nonlinear response surface generated by GP is recommended..

References

Banzhaf W., P. Nordin, R. Keller, and F. Francone, *Genetic Programming: An Introduction*, Morgan Kaufmann, San Francisco, 1998.

Box, G.E.P, and Draper, N. R., *Empirical Model Building and Response Surfaces*, John Wiley and Sons, New York, 1987.

Box, G.E.P., and Cox, D.R., "An Analysis of Transformations", *J. Roy. Stat. Soc.*, Series B, V. 26, p. 211, 1964.

Box, G.E.P., Hunter, W.G., and Hunter, J.S., *Statistics for Experiments: An Introduction to Design, Data Analysis, and Model Building*, John Wiley and Sons, New York, 1978.

Cela, R., Martinez, E., and Carro, A. M, Supersaturated experimental Design: New Approaches to building and Using it Part II Solving Supersaturated Designs by Genetic Programming Algorithms*, Chemometrics and Intelligent Laboratory Systems*, **57**, pp. 75-92, 2001

Kordon, A. K. and G. F. Smits, Soft Sensor Development Using Genetic Programming, Proceedings of the GECCO'2001, San Francisco, pp. 1346-1351.

Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

Montgomery, D. C., *Design and Analysis of Experiments*, John Wiley and Sons, New York, 1999.

Spoonger. *Using Factorial Experiments to Evaluate the Effects of Genetic Programming parameters*. In Proceedings of EuroGP'2000, Edinburgh, LNCS Ul802, pp. 2782, 2000.

Using Genetic Algorithms To Solve The Yard Allocation Problem

Ping Chen
Department of Computer Science
National University of Singapore
Singapore 117543
chenp@comp.nus.ed.sg

Zhaohui Fu
Department of Computer Science
National University of Singapore
Singapore 117543
fuzh@comp.nus.ed.sg

Andrew Lim
Department of Computer Science
National University of Singapore
Singapore 117543
alim@comp.nus.ed.sg

Abstract

The Yard Allocation Problem (YAP) is a real-life resource allocation problem faced by the Port of Singapore Authority (PSA). We first show that YAP is NP-Hard. As the problem is NP-Hard, we propose a Genetic Algorithm approach. For benchmarking purposes, Tabu Search and Simulated Annealing are applied to this problem as well. Extensive experiments show very favorable results for the Genetic Algorithm approach.

1 INTRODUCTION

Singapore has one of the world's busiest ports in terms of shipping tonnage with more than one hundred thousand ship arrivals every year. One of the major logistical problems encountered is to use the minimum container yard necessary to accommodate all different requests. Each request consists of a single time interval and a series of yard space requirements during the interval. An interesting constraint applying to every request is that the length of the required space can either increase or remain unchanged as time progresses, and once yard space is allocated to a certain request, that portion of the yard space cannot be freed until the completion of the request. The current allocation is made manually, hence it requires a considerable amount of manpower.

This paper is organized in the following way. Section 2 gives a formal problem definition. This geometrical problem is then transformed into a graph problem in Section 3. For benchmarking purpose, we briefly discuss two heuristics, namely Tabu Search and Simulated Annealing, applied on YAP in Section 4 and 5 respectively. Section 6 illustrates our application of Genetic Algorithms on YAP in details, and various genetic operators are presented in this section. Section 7 compares the different experimental results obtained by those three heuristics. In Section 8, we present our conclusion.

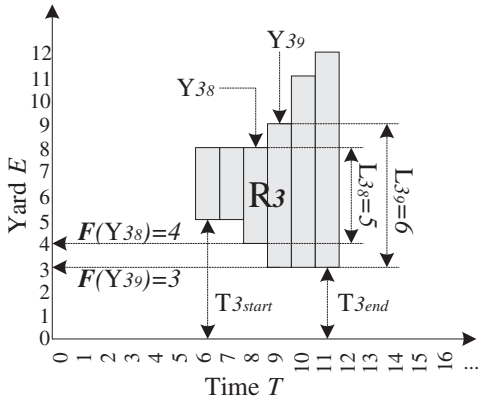


Figure 1: A *valid* request R_3.

2 PROBLEM DEFINITION

The main objective of the Yard Allocation Problem (YAP) is to minimize the container yard used while satisfying all the space requirements. The formal definition of the problem can be described as follows:

Instance: A set R of n yard space requests and an infinite container yard E. $\forall R_i \in R, R_i$ has series of (continuous) space requirements Y_{i_j} with length $L_{i_j}, j \in [T_{i_{start}}, T_{i_{end}}]$.
Output: A mapping function F, such that $F(Y_{i_j}) = e_k$, where $e_k \in E$ is some position on E.
Constraint: $\forall p, q \in [T_{i_{start}}, T_{i_{end}}]$ such that $p = q - 1$, $F(Y_{i_p}) \geq F(Y_{i_q})$ and $F(Y_{i_p}) + L_{i_p} \leq F(Y_{i_q}) + L_{i_q}$.
Objective: To minimize:

$$\max_{\forall R_i \in R, \forall Y_{i_j} \in R_i} (F(Y_{i_j}) + L_{i_j})$$

In other words, the objective is to accommodate all requests

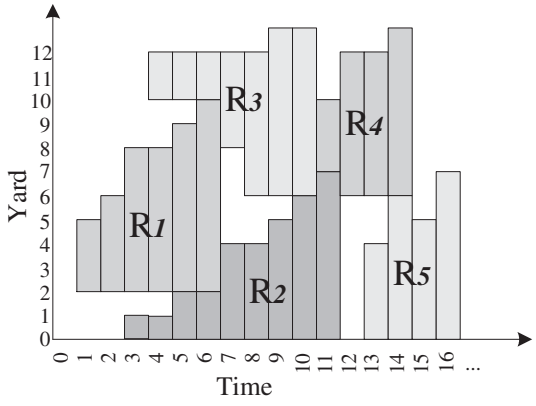


Figure 2: Five *valid* requests on yard

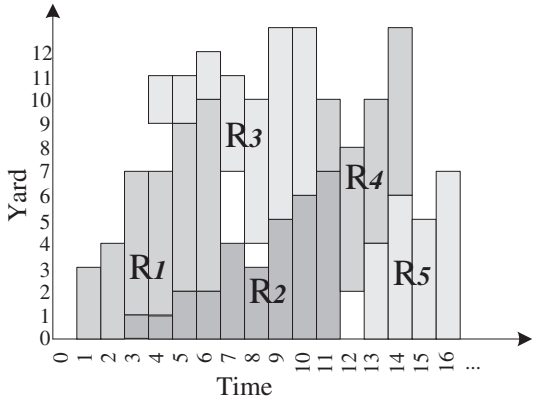


Figure 3: Five *invalid* requests on yard

with the minimum amount of yard space used.

We use an example to illustrate the definition. Figure 1 shows a layout with only one *valid* requests R_3. The yard E is treated as an infinite straight line. Time T becomes a discrete variable with a minimum unit of 1. R_3 has six space requirements within interval $[6, 11]$ ($T_{3_{start}} = 6, T_{3_{end}} = 11$). The final position for Y_{3_8} and Y_{3_9} are $F(Y_{3_8}) = 4$ and $F(Y_{3_9}) = 3$ respectively. The corresponding output for R_3 will then be $(5, 5, 4, 3, 3, 3)$. Note all our pre-defined constraints hold as $F(Y_{3_8}) \geq F(Y_{3_9})$ and $F(Y_{3_8}) + L_{3_8} \leq F(Y_{3_9}) + L_{3_9}$. The max comes from $Y_{3_{11}}$ with the value of $F(Y_{3_{11}}) + L_{3_{11}} = 12$.

We simply call each request a Stair Like Shapes (SLS) throughout this paper. Figure 2 shows five *valid* requests with the minimum yard required of 13. Though the packing in Figure 3 looks more compact, in fact, all allocations are *invalid* as the containment constraint is violated.

Theorem 1 *The Yard Allocation Problem (YAP) is NP-Hard.*

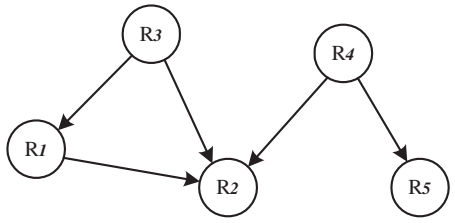


Figure 4: Graph Transformation of Figure 2

The Ship Berthing Problem (SBP) was first introduced in [Lim98]. The SBP has a similar configuration except all the requests are of rectangular shape instead of SLS. [Lim99] has provided an NP-Hard proof for SBP by reducing the Set Partitioning Problem to SBP. As SBP is special case of YAP and YAP is in the class NP, YAP is NP-Hard.

3 GRAPH TRANSFORMATION

Figure 2 illustrates the problem geometrically. However, the direct model may not be efficiently manipulated. We first transform the geometrical layout into a graph. Figure 4 is the corresponding graph transformation of the configuration in Figure 2. Each request R_i is represented by a vertex and there exists an edge E_{ij} connecting R_i and R_j iff R_i and R_j have an overlap at some time. The direction of the edge determines the relative position of the two requests in the physical yard. Take Figure 2 again as an example, both R_1 and R_2 require some space at time 3,4,5 and 6, therefore in Figure 4 there is an edge between R_1 and R_2. Since R_1 is located above R_2, the direction of edge is from R_1 to R_2. We name this edge E_{12}. Clearly, the transformed graph is a Direct Acyclic Graph (DAG). In a DAG, each vertex R_i can be assigned an Acyclic Label (AL) L_i and the edge E_{ij} implies $AL(R_i) < AL(R_j)$. Note that each $AL(R_i)(1 \leq i \leq n)$ is unique.

Lemma 1 *For each feasible layout of the yard, there exists at least one corresponding AL assignment of the vertices in the graph representation.*

A simple constructive proof can be obtained by the well-known Topological Sort algorithm. An AL assignment can also be interpreted as a permutation of $1, 2, \ldots, n$.

A "free" SLS is the one with no other SLS above it, i.e. there is no obstacle blocking it from being popped out from the top of the layout. Again, use Figure 2 as example. At the first iteration of the loop, R_3 and R_4 are the only two "free" SLSs. If we assign $AL(R_3) = 0$, in the second

iteration, R_1 will become a new "free" SLS. The process continues until no more SLS is left in L.

The AL assignment only has the partial order property. Each physical layout may correspond to more than one AL assignments due to the lack of total order property. $[R_1 : 2, R_2 : 3, R_3 : 0, R_4 : 1, R_5 : 4]$ and $[R_1 : 2, R_2 : 4, R_3 : 1, R_4 : 0, R_5 : 3]$ are two possible AL assignments.

This one-to-many relationship between physical layout and AL assignments in the graph representation will incur a huge amount of confusion in heuristic searches, including Genetic Algorithms, etc. Heuristic methods tend to identify certain *good* patterns which may potentially lead to a better solution while exploring the search space. Two very different looking solutions, which may actually correspond to the same physical layout, will make it very difficult for the heuristic to indentify the correct patterns.

We can avoid such confusion by normalizing the AL assignment. When there are more than one SLSs to be popped out, we break the tie by selecting the SLS with the smallest label. Each un-normalized AL assignment is used to construct the corresponding DAG. Then a Topological Sort with above-mentioned tie-breaker will give the *unique* AL assignment. From this point onwards, all our solutions are represented by their normalized unique AL assignments.

Each physical layout now has a unique AL assignment. Naturally, the optimal layout has an optimal AL assignment. Our goal is to find out such an optimal AL assignment. One of the major operations, the evaluation of a given AL assignment, turns out to be non-trivial. In SBP [Lim98] [FL00], a longest path algorithm on a DAG was used to find the minimum berth length needed. However, YAP deals with SLS, whose relative position and distance cannot be calculated in a straight-forward way, unlike rectangles. We have to use a recursive procedure to find the minimum yard needed for a given AL assignment A.

Evaluate-Solution (A)
1 **while** exists unallocated SLS
2 pick SLS S with largest AL
3 Drop($S, S_{end}, 0$)
4 **foreach** time T_i
5 **if** $T_i > L$
6 $L := T_i$
7 **return** L

Drop (S, t, l)
1 L :=lowest position to drop all stairs (time t')
2 **if** $L < l$
3 $L = l$
4 **forall** stair s after $t' - 1$
5 drop s to position L
6 Drop($S, t' - 1, L$)

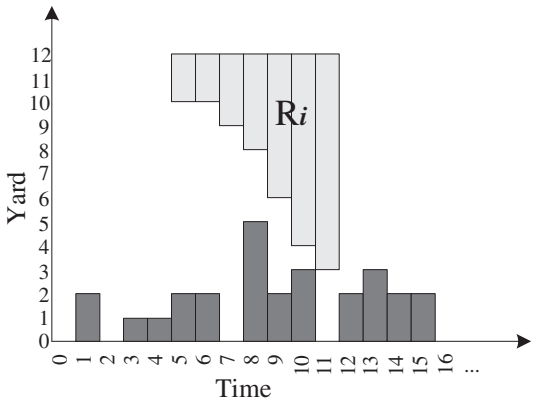


Figure 5: Before dropping: R_i is ceiling aligned

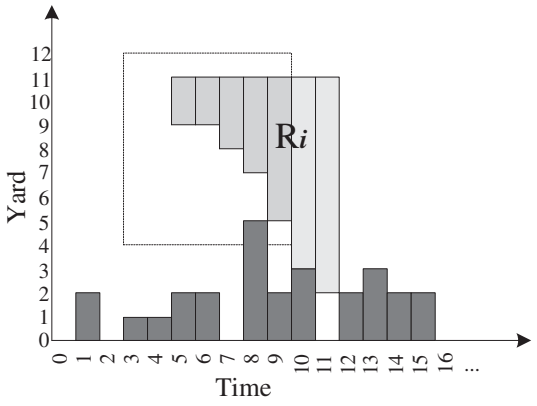


Figure 6: Each stair of R_i drop by 1. Stairs at time 10 and 11 are in their final positions. Those stairs which can drop further are in dark color surrounded by a rectangle

The recursive function $Drop$ uses a greedy approach to drop a given SLS to a position as low as possible. We illustrate the details through Figures 5, 6 and 7: R_i has seven space requirements starting from time 5 till 11. R_i is first aligned to the ceiling before the process starts (Figure 5). Then from time 5 to 11, we find the maximum distance that each "stair" can drop, without exceeding a lower bound of 0. The minimum amongst all the maximum possible drop is used. In this case, the minimum distance of 1 is given at time 10 and hence every stair is shifted down by 1 (Figure 6). Because of the initial ceiling alignment, no further shifting down is needed for all stairs at time 10 (inclusive) onwards. Note that the surface was touched at time 10.

Stairs from time 5 to 9, which are surrounded by a rectangle in Figure 6, can still be dropped further but this time with a lower bound of 3, which is the height of the pre-

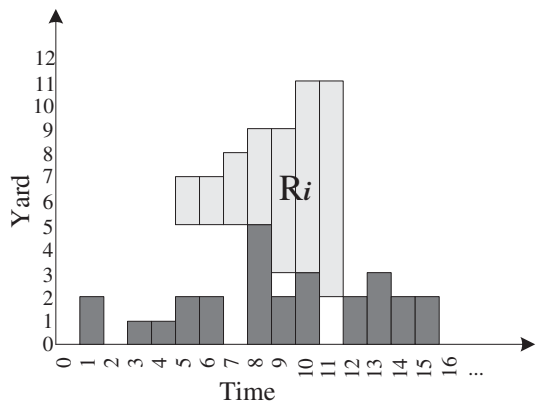


Figure 7: Final layout

vious touching surface at time 10. The dropping process completes after a few more recursions at time 8,7,6 and 5. The final layout is shown by Figure 7. Note the worst case time complexity for $Drop$ is $n \times T$, where n is the number of requests and T is the average time span for all requests.

Lemma 2 *For a given AL assignment, the greedy dropping approach always returns the layout with minimum yard used.*

Proof. The proof of the correctness of a greedy algorithm consists of two parts: First, the greedy choice always leads to an optimal solution, or any optimal solution can be transformed into a solution obtained by the greedy choice. Second, the problem has an optimal sub-structure, i.e. the global optimal implies local optimal. The optimal sub-structure property is obvious for YAP. To show the greedy choice property, we compare the solution G obtained by greedy dropping approach with any arbitrary optimal solution O. Consider the following algorithm:

Compact (A, G, O)
1 **let** $L :=$ set of SLSs;
2 **while** L is not empty
3 pick SLS S with largest AL
4 **for** $(i = S_{begin}; i \leq S_{end}; i++)$
5 **let** $G_{s_i} :=$ position of S_i in G
6 **let** $O_{s_i} :=$ position of S_i in O
7 **if** $O_{s_i} > G_{s_i}$
8 $O_{s_i} := G_{s_i}$

The algorithm $Compact$ will transform any optimal solution into a corresponding solution that can be obtained by the greedy approach without increase the amount of the yard used. Note line 7 is based on the fact that no optimal solution can allocate S_i in a lower position than greedy approach.

Up to now, we have built a one-to-one relationship between physical layout and the AL assignment $(0, 1, \ldots, n-1)$. The problem is to find the optimal AL assignment.

4 TABU SEARCH

Tabu Search [GL97] [Ham93] is a local search meta-heuristic. According to the different usage of memory, conventionally, Tabu Search has been classified into two categories: Tabu Search with Short Term Memory (TSSTM) and Tabu Search with Long Term Memory (TSLTM) [GL97] [SY99]. Tabu Search can also be hybridized with other heuristics, like Squeaky Wheel Optimization [CFL02].

4.1 TABU SEARCH WITH SHORT TERM MEMORY

Our TSSTM implementation consists of two major components: neighborhood search and the tabu list. The neighborhood solution can be obtained by swapping any two ALs in the AL assignment. For example:

$$(2\ 3\ 0\ 1\ 4) \rightarrow (1\ 3\ 0\ 2\ 4)$$

by interchanging the positions of 1 and 2. However, certain swaps, after normalization, may be identical to the original AL assignment. Such solutions are excluded from the neighborhood for efficiency.

Our solution is represented in an AL assignment, which is just a series of numbers. Due to this simplicity, our Tabu List is implemented to record the whole AL assignment for a certain number of solutions recently visited. To be more efficient, string matching algorithms are used to identify the *tabu active* solutions.

4.2 TABU SEARCH WITH LONG TERM MEMORY

We implemented TSLTM in two phases: Diversification and Intensification. We used two kinds of diversification techniques, one is a random re-start and the other is to randomly pick a sub-sequence and insert it into a random position. For example:

$$(0 \mid 1\ 2 \mid 3\ 4) \rightarrow (0\ 3\ 4 \mid 2\ 1 \mid)$$

if $(\mid 1\ 2 \mid)$ is chosen as the sub-sequence and its inverse (or original, if random) is inserted at the back. Intensification is similar to TSSTM. TSLTM uses a *frequency* based memory by recording both *residence* frequency and *transition* frequency of the visited solutions. In our implementation, residence frequency is taken as the number of times that the $AL(R_i) < AL(R_j), 1 \leq i, j \leq n$ in the selected solution

in each iteration. The transition frequency is taken as the summation of the improvements when $AL(R_i)$ is swapped with $AL(R_j)$. The sum can be either positive or negative.

Diversification and Intensification are interleaved and during either phase, the residence frequency and transition frequency are updated according to the current selected solution. The objective function has three contributors. Besides the length of the yard space required, both residence frequency and transition frequency are used to evaluate the solution.

5 SIMULATED ANNEALING

Simulated annealing [Haj88], [KGV83], [OG89] is a very general optimization method which stochastically simulates the slow cooling of a physical system.

We used the following Simulated Annealing algorithm on our problem:

Step 1. Choose some initial temperature T_0 and a random initial starting configuration θ_0. Set $T = T_0$. Define the Objective function (Energy function) to be $En()$ and the cooling schedule σ.

Step 2. Propose a new configuration, θ' of the parameter space, within a neighborhood of the current state θ, by setting $\theta' = \theta + \phi$ for some random vector ϕ.

Step 3. Let $\delta = En(\theta') - En(\theta)$. Accept the move to θ' with probability

$$\alpha(\theta, \theta') = \begin{cases} 1 & \text{if } \delta < 0 \\ exp(-\frac{\delta}{T}) & \text{otherwise} \end{cases}$$

Step 4. Repeat Step 2 and 3 for K of iterations, until it is deemed to have reached the equilibrium.

Step 5. Lower the temperature by $T = T \times \sigma$ and repeat Steps 2-4 until certain stopping criterion, for our case $T < \epsilon$, is met.

Due to the logarithmic decrement of T, we set $T_0 = 1000$. The Energy function is simply defined as the length of the yard required. The probability $exp(-\frac{\delta}{T})$ is known as the Boltzmann factor. The number of iterations K is proportional to the input size n. The neighborhood is defined similarly as the one in Tabu Search, which are swapping of any two AL and re-positioning of a random AL subsequence.

6 GENETIC ALGORITHM

Genetic Algorithms [Hol75] are search procedures that use the mechanics of natural selection and natural genetics.

It is clear that the classical binary representation is not a suitable in YAP, in which a list of Acyclic Labels $(0, 1, \ldots, n-1)$ is used as the solution representation. The solution space is a permutation of $(0, 1, \ldots, n-1)$. The binary codes of these AL do not provide any advantage. Sometimes the situation is even worse: the change of a single bit may not result in a valid solution. We adopt a vector representation, i.e. use the AL assignment directly as the chromosome in the genetic process. We will illustrate the two major genetic operators used in our approach, crossover and mutation.

6.1 CROSSOVER OPERATOR

Using AL assignment as chromosome, we have implemented three crossover operators:

- Classical crossover with repair.
- Partially-mapped crossover.
- Cycle crossover.

All these operators are be tailored to suit our problem domain. A tiny change in the crossover operator may act in totally different manners.

6.1.1 Classical Crossover with Repair

The Classical Crossover operator is the simplest among the three methods mentioned above. It builds the offspring by appending the head from one parent with the tail from the other parent, where the head and tail come from random cut of the parents' chromosomes. A repair procedure may be necessary after the crossover [Mic96]. For example, the two parents (with random cut point marked by '|'):

$$\begin{aligned} p_1 &= (0\,1\,2\,3\,4\,5 \mid 6\,7\,8\,9) \text{ and} \\ p_2 &= (3\,1\,2\,5\,7\,4 \mid 0\,9\,6\,8). \end{aligned}$$

will produce the following two offsprings:

$$\begin{aligned} o_1 &= (0\,1\,2\,3\,4\,5 \mid 0\,9\,6\,8) \text{ and} \\ o_2 &= (3\,1\,2\,5\,7\,4 \mid 6\,7\,8\,9). \end{aligned}$$

However, the two offsprings are not valid AL assignments after the crossover. A repair routine replaces the repeated ALs with the missing ones randomly. The repaired offsprings will be:

$$\begin{aligned} o_1 &= (7\,1\,2\,3\,4\,5 \mid 0\,9\,6\,8) \text{ and} \\ o_2 &= (3\,1\,2\,5\,7\,4 \mid 6\,0\,8\,9). \end{aligned}$$

The classical crossover operator tries to maintain the absolute AL positions in the parents.

6.1.2 Partially Mapped Crossover

Partially Mapped Crossover (PMX) was first used in [GL85] to solve the Traveling Salesman Problem (TSP). We have made several adjustments to accommodate our chromosome (AL assignment) representation. The modified PMX builds an offspring by choosing a subsequence of an AL assignment from one parent and preserving the order and position of as many ALs as possible from the other parent. The subsequence is determined by choosing two random cut points. For example, the two parents:

$$\begin{aligned} p_1 &= (0\ 1\ 2 \mid 3\ 4\ 5\ 6 \mid 7\ 8\ 9) \text{ and} \\ p_2 &= (3\ 1\ 2 \mid 5\ 7\ 4\ 0 \mid 9\ 6\ 8). \end{aligned}$$

would produce offspring as follows. First, two segments between cutting points are swapped (symbol 'u' represents 'unknown' for this moment):

$$\begin{aligned} o_1 &= (u\ u\ u \mid 5\ 7\ 4\ 0 \mid u\ u\ u) \text{ and} \\ o_2 &= (u\ u\ u \mid 3\ 4\ 5\ 6 \mid u\ u\ u). \end{aligned}$$

The swap defines a series of mappings implicitly at the same time:

$$3 \leftrightarrow 5, 4 \leftrightarrow 7, 5 \leftrightarrow 4 \text{ and } 6 \leftrightarrow 0.$$

The 'unknown's are then filled in with AL from original parents, for which there is no conflict:

$$\begin{aligned} o_1 &= (u\ 1\ 2 \mid 5\ 7\ 4\ 0 \mid u\ 8\ 9) \text{ and} \\ o_2 &= (u\ 1\ 2 \mid 3\ 4\ 5\ 6 \mid 9\ u\ 8). \end{aligned}$$

Finally, the first u in o_1 (which should be 0, who will cause a conflict) is replaced by 6 because of the mapping $0 \leftrightarrow 6$. Note such replacement is transitive, for example, the second u in o_1 should follow the mapping $7 \leftrightarrow 4, 4 \leftrightarrow 5, 5 \leftrightarrow 3$ and is hence replaced by 3. The final offspring are:

$$\begin{aligned} o_1 &= (6\ 1\ 2 \mid 5\ 7\ 4\ 0 \mid 3\ 8\ 9) \text{ and} \\ o_2 &= (7\ 1\ 2 \mid 3\ 4\ 5\ 6 \mid 9\ 0\ 8). \end{aligned}$$

The PMX crossover exploits important similarities in the value and ordering simultaneously when used with an appropriate reproductive plan [GL85].

6.1.3 Cycle Crossover

Original Cycle Crossover (CX) was proposed in [OSH87], again for the TSP problem. Our CX builds offspring in such a way that each AL (and its position) comes from one of the parents. We explain the mechanism of the CX with following example. Two parents:

$$\begin{aligned} p_1 &= (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9) \text{ and} \\ p_2 &= (3\ 1\ 2\ 5\ 0\ 4\ 7\ 9\ 6\ 8). \end{aligned}$$

will produce the first offspring by taking the first AL from the first parent:

$$o_1 = (0\ u\ u\ u\ u\ u\ u\ u\ u\ u).$$

Since every AL in the offspring should come from one of its parents (for the same position), the only choice we have at this moment is to pick AL 3, as the AL from parent p_2 just "below" the selected 0. In p_1, it is in position 3, hence:

$$o_1 = (0\ u\ u\ 3\ u\ u\ u\ u\ u\ u).$$

which, in turn, implies AL 5, as the AL from p_2 "below" the selected 3:

$$o_1 = (0\ u\ u\ 3\ u\ 5\ u\ u\ u\ u).$$

Following the rule, the next AL to be inserted is 4. However, selection of 4 requires the selection of 0, which is already in the list. Hence the cycle is formed as expected.

$$o_1 = (0\ u\ u\ 3\ 4\ 5\ u\ u\ u\ u).$$

The remaining 'u's are filled from p_2:

$$o_1 = (0\ 1\ 2\ 3\ 4\ 5\ 7\ 9\ 6\ 8).$$

Similarly,

$$o_2 = (3\ 1\ 2\ 5\ 0\ 4\ 6\ 7\ 8\ 9).$$

The CX preserves the absolute position of the elements in the parent sequence [Mic96].

Our experiments shows Classical crossover and CX have a stable but slow improvement rate according to time while PMX demonstrates an oscillating but fast convergence trend. In our later experiments, majority of the crossover is done by PMX. Classical crossover and CX are applied at a much lower probabilities.

6.2 MUTATION OPERATOR

Mutation is another classical genetic operator, which alters one or more genes (portion of the chromosome) with a probability equal to the mutation rate. There are several known mutation algorithms which work well on different problems:

- Inversion: invert a subsequence.
- Insertion: select an AL and insert it back in a random position.
- Displacement: select a subsequence and insert it back in a random position.
- Reciprocal Exchange: swap two ALs.

In fact the Inversion, Displacement and Reciprocal Exchange are quite similar to our neighborhood solution and diversification techniques used in Tabu Search and Simulated Annealing in previous sessions. We adopt a relatively low mutation rate at 1%.

The population size $P = 1000$ is set for most cases. The evolution process starts with a random population. The population is sorted according to the objective function, the best the quality, the higher the probability it will be selected for reproduction. At each iteration, a new generation with population size $2P$ is produced and the better half, which is of size P, survive for the next iteration. The evolution process continues until certain stop criterion are met.

7 EXPERIMENTAL RESULTS

Table 1: Experimental results (Entries in the table shows the minimum length of the yard required. Name of Data Set shows the number of SLSs in the file; LB:Lower Bound)

Data Set	LB	TSSTM	TSLTM	SA	GA
R126	21	28	26	25	24
R117	34	39	37	34	34
R145	39	50	45	42	39
R178	50	69	69	66	55
R188	74	105	98	102	79
R173	77	98	91	98	79
R250	83	141	119	117	89
R236	97	139	130	114	101
R213	164	245	246	245	187

We conducted extensive experiments on randomly generated data[1] The graph for each test case contains one connected components, in other words, the test cases cannot be partitioned into more than one independent sub-case. Due to the difficulties of finding any optimal solution in the experiments, a trivial *lower bound* is taken to be the sum of the space requirements at each time slot and used for benchmarking purpose.

Table 1 illustrates the results. It is not surprising to see that GA outperforms all other heuristics in all test cases by a considerable margin. TSSTM has the simplest implementation with the worst results. TSLTM has an obvious improvement from TSSTM, though the improvement is not very stable. We believe one of the major difficulties with Long Term Memory is the assignment of relative weights to yard length, residence frequency and transition frequency in the objective function. SA is relatively easy to implement with comparable results to TSLTM. The most successful approach, using Genetic Algorithm, gives the best results, which are within 8% of the trivial lower bound at most of the time.

[1]All test data are available on the WWW with URL: http://www.comp.nus.edu.sg/~fuzh/YAP.

Table 2: Experiment running time (seconds) for Table 1.

Data Set	TSSTM	TSLTM	SA	GA
R126	594	3423	2023	302
R117	753	2521	1873	442
R145	1215	3693	2233	784
R178	2568	5362	2576	1023
R188	2523	7822	3529	1321
R173	3432	6743	3431	1675
R250	4578	10239	5031	3632
R236	5027	11053	5892	2453
R213	4891	10476	6342	4322

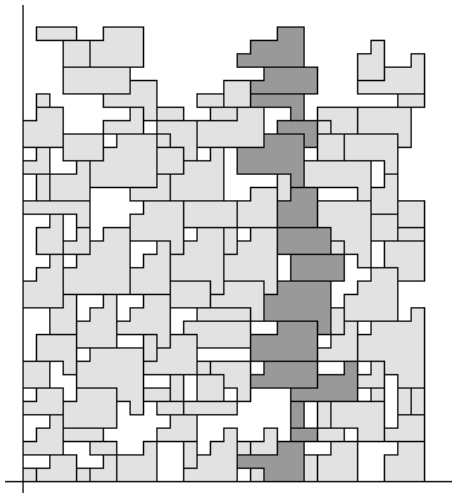


Figure 8: Physical layout of 117 SLSs (requests). Data Set: R117

Table 2 shows the running time for each of the test performed in Table 1 on a Dual-CPU (Pentium III 800MHz each) Linux machine. It is clear that GA is also the most cost-effective approach.

Another interesting discovery from the experiments is that the normalization routine does not improve the results of GA as much as our expectation. Besides the slowing down factor, it sometimes even degenerates the results. We believe that normalization should make the search process more stable and focus by removing the *confusing* factors. But its side effects, for example reducing the solution space, sometimes overwhelm its merits.

Figure 8 and Figure 9 provide the graphic results obtained by GA for input file R117 and R145. The heavily-shaded SLSs contain the region that is the densest (lower bound) in both figures. Due to the stair-like shapes, the packing layout looks sub-optimal. A closer look reveals further improvements to be very unlikely.

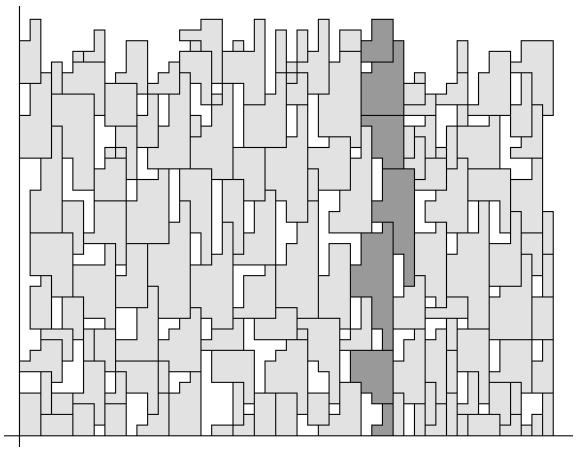


Figure 9: Physical layout of 145 SLSs (requests). Data Set: R145

8 CONCLUSION

In this paper, we have shown the Yard Allocation Problem (YAP) is NP-Hard by reducing the Ship Berthing Problem (SBP) to it. The geometrical representation of YAP is then transformed into a Direct Acyclic Graph (DAG) for efficient manipulation. A normalization procedure is proposed to guarantee a one-to-one relationship between geometric layout and Acyclic Label Assignment of the DAG. Finding the optimal layout is transformed to the search for the optimal Acyclic Label Assignment. Two heuristic methods, Tabu Search and Simulated Annealing are first applied. The results obtained are not very attractive.

A Genetic Algorithm approach is applied after TS and SA's failing to achieve good results against the lower bound. Various genetic operators are proposed and applied. Extensive experiments showed our GA approach, outperformed both the original Tabu Search and Simulate Annealing by a margin of more than 10%.

References

[CFL02] Ping Chen, Zhaohui Fu, and Andrew Lim. The yard allocation problem. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Edmonton, Alberta, Canada*, 2002.

[FL00] Zhaohui Fu and Andrew Lim. A hybrid method for the ship berthing problem. In *Proceedings of the Sixth Artificial Intelligence and Soft Computing*, 2000.

[GL85] D. Goldberg and R. Lingle. Alleles, loci, and the traveling salesman problem, 1985.

[GL97] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[Haj88] Bruce Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13:311–329, 1988.

[Ham93] Peter L. Hammer. *Tabu Search*. Basel, Switzerland: J.C. Baltzer, 1993.

[Hol75] John H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[KGV83] S. Kirpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[Lim98] Andrew Lim. On the ship berthing problem. *Operations Research Letters*, 22(2-3):105–110, 1998.

[Lim99] Andrew Lim. An effective ship berthing algorithm. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 594–599, 1999.

[Mic96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag Berlin Heidelberg New York, 1996.

[OG89] Ralph H. J. M. Otten and Lukas P. P. P. van Ginneken. *The Annealing Algorithm*. Kluwer Academic Publishers, Boston, U.S.A., 1989.

[OSH87] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the tsp, 1987.

[SY99] Sadiq Sait and Habib Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE, 1999.

Gaphyl: An Evolutionary Algorithms Approach for the Study of Natural Evolution

Clare Bates Congdon
Computer Science Department
Colby College
5846 Mayflower Hill Drive
Waterville, ME 04901
ccongdon@colby.edu

Abstract

Gaphyl is an application of genetic algorithms (GA's) to phylogenetics, an approach used by biologists to investigate the evolutionary relationships among organisms. Typical phylogenetic software packages use heuristic search methods to navigate through a space of possible trees in an attempt to find the most plausible evolutionary hypotheses, as exhaustive search is not practical in this domain. Gaphyl substitutes an evolutionary search mechanism, with the result that on a complex problem from the literature (the major clades of the angiosperms), Gaphyl is able to find a more complete solution (more equally plausible hypotheses) in less time than the standard approach. Contributions of GA operators are investigated, as are some possibilities for hybrid systems.

1 INTRODUCTION

The human genome project and similar projects in biology have led to a wealth of data and the rapid growth of the emerging field of bioinformatics, a hybrid discipline between biology and computer science that uses the tools and techniques of computer science to help manage, visualize, and find patterns in this data. The work reported here is an application to biology, and indicates gains from using genetic algorithms (GA's) as the search mechanism for the task.

Phylogenetics [6] is a method widely used by biologists to reconstruct hypothesized evolutionary pathways followed by species currently or previously inhabiting the Earth. Given a dataset that contains a number of different species, each with a number of attribute-values, phylogenetics software constructs phylogenies, which

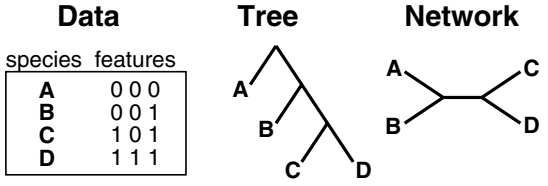


Figure 1: A toy example data set, sample phylogeny, and sample network. In this example, there are four species and three features. The tree formed shows the hypothesis that species B is related to species A, gaining the third feature. Similarly, C and D are more closely related to B than to A, also acquiring new features.

are representations of the possible evolutionary relationships among the given species. A typical phylogeny is a tree structure: The species nearest the root of a tree can be viewed as the common ancestor, the leaves of a tree are the species, and subtrees are subsets of species that share a common ancestor. Each branching of a parent node into offspring represents a divergence in one or more attribute-values of the species within the two subtrees. In an alternate approach, sometimes called "unrooted trees" or "networks", the root of the tree is not assumed to be an ancestral species, although these hypotheses are often drawn as trees as a convenience. Unrooted trees represent hypothetical relationships between species, but do not attempt to model ancestral relationships.

An example phylogeny for a toy data set is shown in Figure 1. In this example, species A is the common ancestor in the tree, and B is the common ancestor of the subtree below A (assuming the tree is rooted). The relationships between species is also shown in the network representation, to better understand the unrooted tree.

Phylogenies are evaluated using metrics such as parsimony: A tree with fewer evolutionary changes is considered better than one with more evolutionary

changes. The work reported here used Wagner parsimony. Wagner parsimony is straightforward to compute (requiring only a single pass through the tree) and incorporates few constraints on the evolutionary changes that will be considered. For example, some parsimony approaches require the assumption that species will only grow more complex via evolution — that features will be gained, but not lost in the process.

The typical phylogenetics approach uses a deterministic hillclimbing methodology to find a phylogeny for a given dataset, saving one or more "most parsimonious" trees as the result of the process. The most parsimonious trees are the ones with a minimum number of evolutionary changes connecting the species in the tree. Multiple "bests" correspond to equally plausible evolutionary hypotheses, and finding more of these competing hypotheses is an important part of the task. The tree-building approach adds each species into the tree in sequence, searching for the best place to add the new species. The search process is deterministic, but multiple trees may be found in the process of the search, and different trees may be found by running the algorithm with different random "jumbles" of the order of the species in the dataset.

This research is an investigation into the utility of using evolutionary algorithms on the problem of finding parsimonious phylogenies.

2 DESIGN DECISIONS

To hasten the development of our system, we used parts of two existing software packages. Phylip [5] is a phylogenetics system widely used by biologists. In particular, this system contains code for evaluating the parsimony of the phylogenies (as well as some helpful utilities for working with the trees). Using the Phylip source code rather than writing our own tree-evaluation modules also helps to ensure that our trees are properly comparable to the Phylip trees. Genesis [7] is a genetic algorithms (GA) package intended to aid the development and experimentation with variations on the GA. In particular, the basic mechanisms for managing populations of solutions and the modular design of the code facilitate implementing a GA for a specific problem. We named our new system Gaphyl, a reflection of the combination of GA and Phylip source code.

The research described here was conducted using published datasets available over the internet [4]. The first dataset used is the families of the superorder of Lamiiflorae dataset [1], consisting of 23 species and 29 attributes. This dataset was chosen as being large enough to be interesting, but small enough to be manageable. A second dataset, the major clades of the angiosperms [3], consisting of 49 species and 61 attributes, was used for further experimentation. These datasets were selected because the attributes are binary, which simplified the development of the system. As a preliminary step in evaluating the GA as a search mechanism for phylogenetics, "unknown" values for the attributes were replaced with 1's to make the data fully binary. This minor alteration to the data does impact the meaningfulness of the resulting phylogenies as evolutionary hypotheses, but does not affect the comparison of Gaphyl and Phylip as search mechanisms.

The typical GA approach to doing "crossover" with two parent solutions with a tree representation is to pick a subtree (an interior or root node) in both parents at random and then swap the subtrees to form the offspring solution. The typical mutation operator would select a point in the tree and mutate it to any one of the possible legal values (here, any one of the species). However, these approaches do not work with the phylogenies because each species must be represented in the tree exactly once.

Operators designed specifically for this task are described in the following sections and in more detail in [2].

2.1 CROSSOVER OPERATOR

The needs for our crossover operator bear some similarity to traveling salesperson problems (TSP's), where each city is to be visited exactly once on a tour. There are several approaches in the literature for working on this type of problem with a GA, however, the TSP naturally calls for a string representation, not a tree. In designing our own operator, we studied TSP approaches for inspiration, but ultimately devised our own. We wanted our operator to attempt to preserve some of the species relationships from the parents. In other words, a given tree contains species in a particular relationship to each other, and we would like to retain a large degree of this structure via the crossover process.

Our crossover operator proceeds as follows:

1. Choose a species at random from one of the parent trees. Select a subtree at random that includes this node, excluding the subtree that is only the leaf node and the subtree that is the entire tree. (The exclusions prevent crossovers where no information is gained from the operation.)
2. In the second parent tree, find the smallest subtree containing all the species from the first parent's subtree.

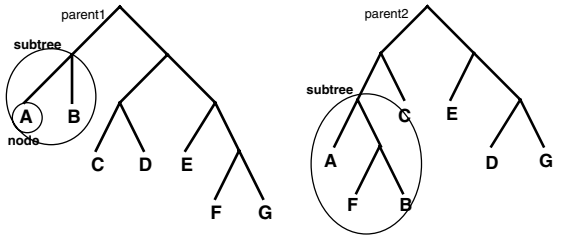


Figure 2: Two example parent trees for a phylogenetics problem with seven species. A subtree for crossover has been identified for each tree.

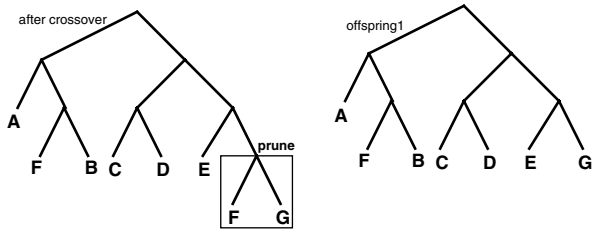


Figure 3: At the left, the offspring initially formed by replacing the subtree from parent1 with the subtree from parent2; on the right, the offspring tree has been pruned to remove the duplicate species F.

3. To form an offspring tree, replace the subtree from the first parent with the subtree from the second parent. The offspring must then be pruned (from the "older" branches) to remove any duplicate species.
4. Repeat the process using the other parent as the starting point, so that this process results in two offspring trees from two parent trees.

This process results in offspring trees that retain some of the species relationships from the two parents, and combine them in new ways.

An example crossover is illustrated in Figures 2 and 3. (Note that in the phylogenies, swapping the left and right children does not affect the meaning of the phylogeny.)

2.2 CANONICAL FORM

Trees are put into a canonical form when saving the best trees found in each generation, to ensure that no equivalent trees are saved among the best ones. Canonical form is illustrated in Figure 4.

2.3 MUTATION OPERATORS

One of our mutation operators selects two leaf nodes (species) at random, and swaps their positions in the tree. This operator allows the GA to investigate slight variations on a parent tree.

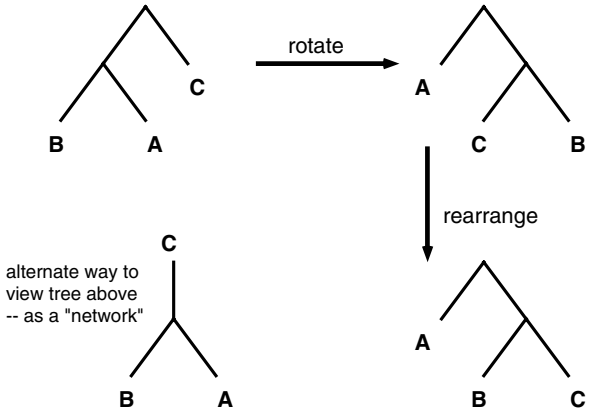


Figure 4: An illustration of putting a tree into canonical form. The tree starts as in the top left; an alternate representation of the tree as a "network" is shown at the bottom left. First, the tree is rotated, so that the first species in the dataset is an offspring of the root. Second, subtrees are rearranged so that smaller trees are on the left and alphabetically lower species are on the left.

A second mutation operator picks a random subtree and a random species within the subtree. The subtree is rotated to have the species as the left child of the root and reconnected to the parent. The idea behind this operator is that within a subtree, the species might be connected to each other in a promising manner, but not well connected to the rest of the tree.

2.4 IMMIGRATION

The population is subdivided into a specified number of subpopulations which, in most generations, are distinct from each other (crossovers happen only within a given subpopulation). After a number of generations have passed, each population migrates a number of its individuals into other populations; each emigrant determines at random which population it will move to and which tree within that population it will uproot. The uprooted tree replaces the emigrant in the emigrant's original population. The number of populations, the number of generations to pass between migrations, and the number of individuals from each population to migrate at each migration event are determined by parameters to the system. Immigration was added due to problems with premature convergence identified in early stages of development.

3 EXPERIMENTAL RESULTS

Recall that both Gaphyl and Phylip have a stochastic component, which means that evaluating each system requires doing a number of runs. In Phylip, each distinct run first "jumbles" the species list into a different random order. In Gaphyl, there are many different ef-

fects of random number generation: the construction of the initial population, parent selection, and the selection of crossover and mutation points. For both systems, a number of different runs must be done to evaluate the approach.

3.1 COMPARISON OF GAPHYL AND PHYLIP

1. With the Lamiiflorae data set, the performance of Gaphyl and Phylip is comparable. Phylip is more expedient in finding a single tree with the best parsimony (72), but both Gaphyl and Phylip find 45 most parsimonious phylogenies in about twenty minutes of run time.
2. With the angiosperm dataset, a similar pattern emerges: Phylip is able to find one tree with the best fitness (279) quite quickly, while Gaphyl needs more run time to first discover a tree of fitness 279. However, in a comparable amount of runtime, Gaphyl is able to find 250 different most parsimonious trees of length 279 (approximately 24 hours of runtime). Phylip runs for comparable periods of time have not found more than 75 distinct trees with a parsimony of 279, and runs of nearly 3 days have not turned up more than 95 distinct trees. Furthermore, the trees found by Phylip are a proper subset of the trees found by Gaphyl.

In other words, Gaphyl is more successful than Phylip in finding more trees (more equally plausible evolutionary hypotheses) in the same time period. This represents a more complete solution to the problem.

The Lamiiflorae task is considerably easier to solve than the angiosperm task. Example parameter settings are a single population of 500, 500 generations, 50% elitism (the 250 best trees are preserved into the next generation), 100% crossover, 10% first mutation, and 100% second mutation. Empirically, it appears that 72 is the best possible parsimony for this dataset, and that there are not more than 45 different trees of length 72.

The angiosperm task seems to benefit from immigration in order for Gaphyl to find the best known trees (fitness 279). Successful parameter settings are 5 populations, population size of 500 (in each subpopulation), 2000 generations, immigration of 5% (25 trees) after every 500 generations, 50% elitism (the 250 best trees are preserved into the next generation), 100% crossover, 10% first mutation, and 100% second mutation. (Immigration does not happen following the final generation.) We have not yet done enough runs with either Phylip or Gaphyl to estimate the maximum number of trees at this fitness, nor a more concise estimate of how long Phylip would have to run to find 250 distinct trees, nor whether 279 is even the best possible parsimony for this dataset.

3.2 BIG PICTURE: THE ROLE OF THE GA IN THIS TASK

In constructing Gaphyl, we used the code from Phylip's evaluation metric, but the search mechanisms are those of the GA described in this section. In other words, we are investigating the use of the GA as an alternate search method for this already established task. There is an immediate gain to our approach: Our search for trees increases in complexity with the number of species in the dataset. The number of attributes, however, does not affect the search. Conversely, the complexity of the search in Phylip increases relative to the number of attributes as well as the number of species in the dataset. Biologists frequently run phylogeny software for weeks at a time, so a savings in speed has a measurable impact.

Both systems are far from optimized, so strong conclusions cannot be drawn from runtime alone. However, the pattern that is emerging is that as the problems get more complex, Gaphyl is able to find a more complete set of trees with less work than what Phylip is able to find. The work done to date illustrates that Gaphyl is a promising approach for phylogenetics work, as Gaphyl finds a wider variety of trees on this problem than Phylip does. This further suggests that Gaphyl may be able to find solutions better than those Phylip is able to find on datasets with a larger number of species and attributes, because it appears to be searching more successful regions of the search space.

While it is true that one cannot compare software on runtime alone, recall that Gaphyl was constructed from existing systems, neither one of which was optimized for speed. In particular, Genesis was designed to simplify GA experimentation and modifications (much like the project here). It is possible to make some comparisons of operations done by the two systems in their search, but these are apples and oranges, since the work done to get from one tree to the next varies between the systems. In Phylip, each jumble corresponds to a hillclimbing search, which (with the Angiosperms dataset) investigates on the order of 10,000 trees for each random ordering of the species list, and 40,000 jumbles in the 24 hours, or on the order of 400 million trees. In Gaphyl, 10 experiments (using different seeds to the random number generator) with 2500 total trees and 2000 generations investigates on the order of 50 million trees in 24 hours, although the number

should be halved due to the 50% elitism.

3.3 CONTRIBUTION OF OPERATORS

To evaluate the contributions of the GA operators to the search, additional runs were done with the first data set (and a single population). Empirically, crossover and the second mutation operator had been found to be the largest contributors to successful search, so attention was focused on the contributions of these operators.

In the first set of experiments, the first mutation rate was set to be 0%. First, the crossover rate was varied from 0% to 100% at increments of 10% while the second mutation rate was held constant at 100%. Second, the second mutation rate was varied from 0% to 100% at increments of 10% while the crossover rate was held constant at 100%. 20 experiments were run at each parameter setting; 500 generations were run.

Figure 5 illustrates the effects of varying the crossover rate (solid line) and second mutation rate (dashed line) on the average number of generations taken to find at least one tree of the known best fitness (72). Experiments that did not discover a tree of fitness 72 are averaged in as taking 500 generations. For example, 0% crossover was unable to find any trees of the best fitness in all 20 experiments, and so its average is 500 generations. This first experiment illustrates that in general, higher crossover rates are better. There is not a clear preference, however, for high rates of the second form of mutation. To look at this operator more closely, the final populations of the 20 experiments were looked at to determine how many of the best trees were found in each run.

Figure 6 illustrates the effects of varying the crossover rate (solid line) and second mutation rate (dashed line) on the average number of best trees found. Experiments that did not discover a tree of fitness 72 are averaged in as finding 0 trees. For example, 0% crossover was unable to find any trees of the best fitness in all 20 experiments, and so its average is 0 of the best trees. As Figure 6 illustrates, runs with a higher second mutation rate tend to find more of the best trees than runs with a lower second mutation rate.

The impact of the first mutation operator had seemed to be low based on empirical evidence. So another set of experiments was done to assess the contribution of this operator. In both, the crossover rate was set at 100%; in one, the second mutation rate was set at 0% and in the other, the second mutation rate was set at 100%. The results of this experiment clearly indicate that higher rates of this form of mutation are not beneficial. Furthermore, this operator is not clearly contributing to the search. The results are illustrated in Figure 7.

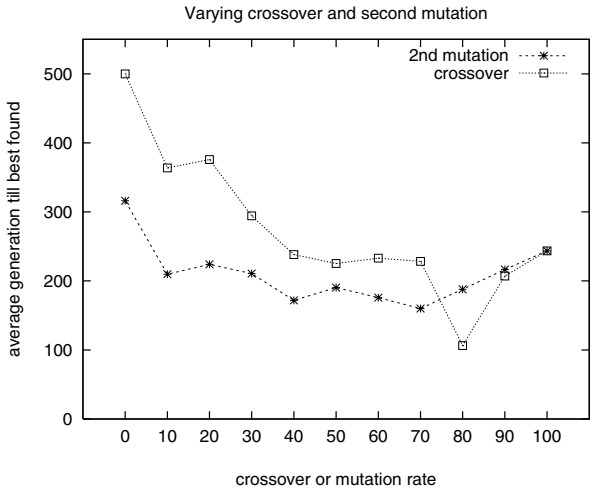


Figure 5: The effect of varying crossover rate while holding second mutation constant and of varying the second mutation rate while holding the crossover rate constant. The average generation at which the best fitness (72) was found is illustrated.

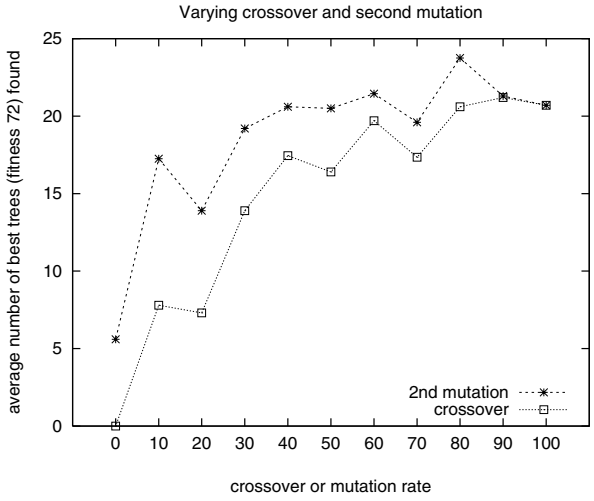


Figure 6: The effects of varying crossover rate while holding second mutation constant and of varying the second mutation rate while holding the crossover rate constant. The average number of best trees (45 max) found by each parameter setting is illustrated.

In the final set of experiments, the first experiments of varying crossover rate while holding second mutation rate constant and vice versa were repeated, but this time with a first mutation rate of 10%. The results are illustrated in Figure 8.

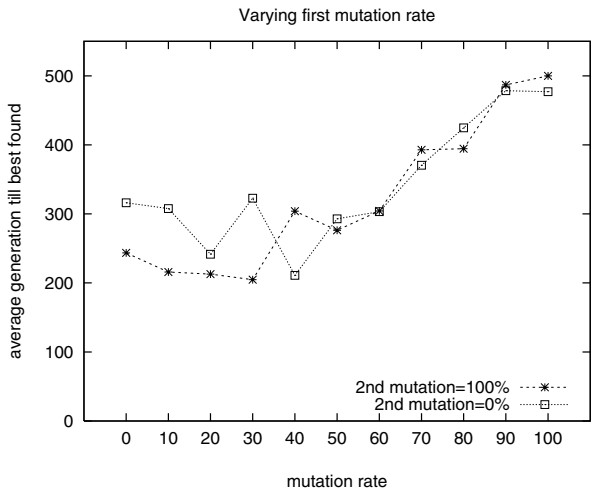


Figure 7: The effect of varying the first mutation rate while holding crossover and second mutation constant. The crossover rate is 100% for both graphs; second mutation rates of 100% and 0% are shown. The average generation at which the best fitness (72) was found is illustrated.

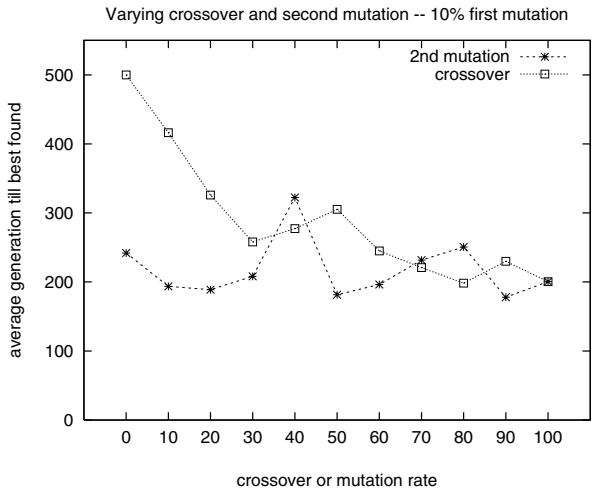


Figure 8: The effect of varying crossover rate while holding second mutation constant and of varying the second mutation rate while holding the crossover rate constant, this time with a first mutation rate of 10%. The average generation at which the best fitness (72) was found is illustrated.

3.4 CONTRIBUTION OF OTHER PARAMETERS

An additional set of experiments was designed to assess tradeoffs in terms of putting a fixed number of trees in one population or distributing them across a number of populations and tradeoffs between having larger population sizes or doing more generations, for a fixed number of evaluations in all cases. These experiments were done using the angiosperms dataset.

Population sizes for each experiment

Gens	Number of Populations				
	1	2	4	8	16
1600	1024	512	256	128	64
800	2048	1024	512	256	128
400	4096	2048	1024	512	256

Table 1: The population size for each experiment described in Section 3.4. When there are multiple populations, the number shown refers to the number of trees in each distinct population.

The base case may be thought of as 1 population of 1024 individuals, and 1600 generations. Then, along one dimension, the population is divided across 2, 4, 8, and 16 populations, a total of five variations. Along the other dimension the number of generations is halved as the population size is doubled, for a total of three variations. This creates an array of 15 parameter settings, illustrated in Table 1. The horizontal axis shows the number of populations, the vertical axis shows the number of generations, and each interior cell shows the population size.

Twenty experiments, with different seeds to the random number generator, were done for each setting. When multiple populations are used, five percent of the population immigrates after 25%, 50%, and 75% of the generations have completed.

The results of these experiments, illustrated in Table 2, show the best results with 2 populations of 1024 trees run for 800 generations, with a total of 7 out of the 20 runs finding trees of the best known fitness of 279. In general, it appears that two populations are better than one, but that there might not be great gains from more than two populations. Further, it appears that the system benefits from a balance between a large population size and a large number of generations.

3.5 EXPLORATION OF HYBRID POSSIBILITIES

We have noted that Phylip is relatively quick to find at least one of the best solutions, but that over a span of time, does not find as many of the best solutions as Gaphyl does. Therefore, it seems that investigating the possibility of a hybrid system would be beneficial. The hybrid variation explored here is to use Phylip runs to seed the initial population of the GA run.

In these experiments, the point of comparison is the

Number of runs that found a "best"

Gens	Number of Populations					
	1	2	4	8	16	sum
1600	1	2	1	2	1	7
800	5	7	2	5	2	21
400	3	3	4	0	0	10
sum	9	12	7	7	3	

Average fitness of final populations

Gens	Number of Populations				
	1	2	4	8	16
1600	281.70	281.55	281.10	280.85	280.95
800	280.60	279.95	280.25	279.95	280.15
400	280.45	280.15	280.45	281.15	281.95

Table 2: The number of runs that found the best solution and the average best solution found across 20 runs, varying the number of populations and number of generations, with a constant 1024 trees (split across the specified number of populations).

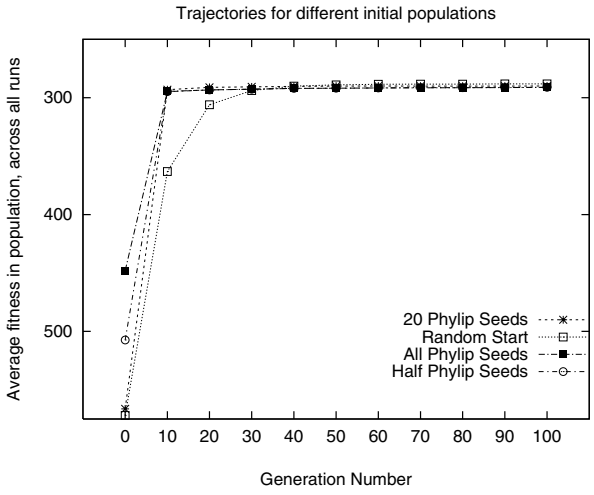
starting point for the system. Four variations were explored, using the angiosperms dataset:

1. Starting with an entirely random initial population.
2. Starting with an initial population comprised of a random selection of trees found by running one Phylip jumble.
3. Starting with an initial population comprised of half Phylip trees from one jumble and half random trees.
4. Starting with an initial population comprised of 20 Phylip trees, one of the best from each of 20 different jumbles, and the remainder random trees.

25 experiments were run for each variation. One population was used, so as not to confound the effects of multiple populations. The population size was 2000 trees, run for 1000 generations. Other parameters are as reported previously.

Of these runs, the 4th variation fared the best, finding at least one tree with the 279 fitness in 14 of the 25 runs. Secondly, the first variation found at least one tree with 279 fitness in 5 of the 25 runs. The second and third variations did not find any trees of 279 fitness in the 25 runs. Trajectories of average fitnesses across all runs are shown in Figure 9.

These experiments suggest that while seeding from Phylip runs may help the progress of the GA, the initial seeds must be sufficiently diverse for this "jump start" to be helpful. It appears that choosing the seed trees from a single Phylip jumble is comparable to starting the GA with a population that has already converged. (Note: This experiment was repeated with five distinct Phylip jumbles, always with similar results.)



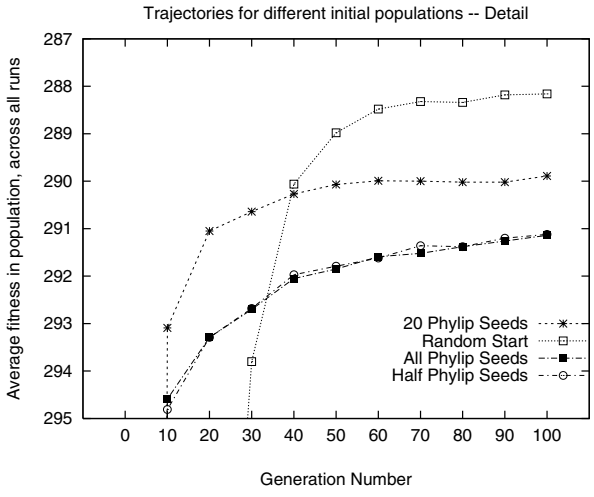


Figure 9: Trajectories for the four experiments with seeding the initial population. The second graph shows more detail of the lower fitness values.

4 CONCLUSIONS AND FUTURE WORK

The GA search process as implemented in Gaphyl represents a gain for phylogenetics in its ability to find more equally plausible trees than Phylip in the same runtime. Furthermore, as the datasets get larger in

the number of species and attributes, the effectiveness of Gaphyl over Phylip appears to increase. One possible facet of this success is that the Gaphyl search process is independent of the number of attributes (and attribute-values); the complexity of the search varies with the number of species (which determines the number of leaf nodes in the tree). Phylip uses attribute information in its search process.

The first mutation operator is perhaps the "obvious" form of mutation to implement for this problem, and yet, its use (at high levels) appears to detract from the success of the search. While multiple populations appear to help the system avoid premature convergence, too many populations are not helpful.

The creation of a hybrid system that uses Phylip's relatively fast but limited search strategy to seed the initial population is a promising approach, as long as care is taken that the seeds are diverse.

There is obviously a wealth of possible extensions to the work reported here. First, more extensive evaluations of the capabilities of the two systems must be done on the angiosperms data set, including an estimate of the maximum number of trees of fitness 279 (and, indeed, whether 279 is the most parsimonious tree possible). This would entail more extensive runs with both approaches.

Second, more work must be done with a wider range of datasets to evaluate whether Gaphyl is consistently able to find a broader variety of trees than Phylip, and perhaps able to find trees better than Phylip is able to find.

Third, Gaphyl should be extended to work with nonbinary attributes. This is particularly important in that phylogenetic trees are increasingly used by biologists primarily with the A, C, G, T markers of genetic data.

Finally, we need to compare the work reported here to other projects that use GA approaches with different forms of phylogenetics, including [8] and [9]. Both of these projects use maximum likelihood for constructing and evaluating the phylogenies. The maximum likelihood approach (which is known as a "distance-based method") is not directly comparable to the Wagner parsimony approach (which is known as a "maximum parsimony" approach).

Acknowledgments

I would like to thank Emily F. Greenfest, who worked with me in the initial design and implementation of this project. Thanks also to Judy L. Stone and Randall Downer for sharing their knowledge of phylogenetic theory and software. Thanks to Randolph M. Jones, Joshua R. Ladieu, and the anonymous reviewers for comments on previous versions of this paper. The Colby College Natural Science Grant Program provided travel and equipment support for the author.

References

[1] L. An-Ming. A preliminary cladistic study of the families of the superorder lamiiflorae. *Biol. J. Linn. Soc.*, 103:39–57, 1990.

[2] C. B. Congdon. Gaphyl: A genetic algorithms approach to cladistics. In L. De. Raedt and A. Siebes, editors, *Principles of Data Mining and Knowledge Dicovery (PKDD 2001)*, Lecture notes in artificial intelligence 2168, pages 67–78, New York, 2001. Springer.

[3] R. Dahlgren and K. Bremer. Major clades of the angiosperms. *Cladistics*, 1:349–368, 1985.

[4] M. J. Donaghue. Treebase: A database of phylogenetic knowledge. web-based data repository, 2000. `http://phylogeny.harvard.edu/treebase`.

[5] J. Felsenstein. Phylip source code and documentation, 1995. Available via the web at `http://evolution.genetics.washington.edu/phylip.html`.

[6] P. L. Forey, C. J. Humphries, I. L. Kitching, R. W. Scotland, D. J. Siebert, and D. M. Williams. *Cladistics: A Practical Course in Systematics.* Number 10 in The Systematics Association Series. Clarendon Press, Oxford, 1993.

[7] J. J. Grefenstette. A user's guide to GENESIS. Technical report, Navy Center for Applied Research in AI, Washington, DC, 1987. Source code updated 1990; available via the web at `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/genesis/`.

[8] P. O. Lewis. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Mol. Biol. Evol.*, 15(3):277–283, 1998.

[9] H. Matsuda. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. In L. Hunter and T. E. Klein, editors, *Pacific Symposium on Biocomputing '96*, pages 512–523. World Scientific, London, 1996.

Learning in *RoboCup* Keepaway using Evolutionary Algorithms

Anthony Di Pietro **Lyndon While** **Luigi Barone**

Department of Computer Science & Software Engineering
The University of Western Australia
Western Australia 6009
Email: {anthony, lyndon, luigi}@cs.uwa.edu.au
Telephone: +61 8 9380 2720

Abstract

Manually coordinating the efforts of many autonomous agents can be a formidable challenge, so the idea of using machine learning techniques (such as evolutionary algorithms) to produce such coordination is attractive. We present a study using evolutionary algorithms to train autonomous agents to play the game of *keepaway* — a sub-problem of the *RoboCup* robotic soccer league. Our results exceed those previously produced using other methods.

1 INTRODUCTION

Multi-agent systems is an active area of artificial intelligence research that focuses on the interaction of artificially intelligent agents. Manually coordinating the efforts of many autonomous agents can be a formidable challenge, so the idea of using machine learning to produce such coordination is attractive.

Team sports, such as soccer, involve autonomous humans collaborating to achieve a common goal. The *RoboCup* initiative challenges AI researchers to achieve this same collaboration among autonomous software agents in a complex, noisy, real-time environment.

Evolutionary algorithms are a robust technique for learning in such environments. A population of candidates is generated with an initial solution encoded into each candidate, and the population evolves to produce better solutions.

This paper describes a study where autonomous agents learn to play the game of *keepaway* (a sub-problem of the *RoboCup* robotic soccer league) using evolutionary algorithms, and compares our approach to others that have been used previously.

Section 2 describes the *RoboCup* set-up, introduces the keepaway sub-problem, and discusses previous work in this area. Section 3 describes the evolutionary algorithm that we use and gives details of its various components, especially the genetic representation (Section 3.2) and the fitness function (Section 3.3). Section 4 describes our experiments and results, and Sections 5 and 6 conclude the paper.

2 THE *ROBOCUP* SIMULATION LEAGUE

The *RoboCup* initiative (Kitano et al., 1997) aims to create a robot soccer team that can beat the human world champion team by the year 2050. It is hoped that this ambitious goal will promote research and collaboration in a variety of fields. There are several *RoboCup* leagues for physical robots of different sizes, and there is also a simulator league in which software agents connect via a network to the *RoboCup* soccer server and play in a virtual environment.

The soccer server provides a virtual soccer field, simulates the physics of the ball and players, and enforces the rules of the game. The field, physics, and rules used by the server are based on those of real soccer, but there are significant differences: for example, the goal width is doubled and the world is two dimensional — and the referee always makes perfect judgements! The soccer server communicates with the players via a client/server protocol, allowing them to receive information about the game state and send commands to perform actions. The game state available to the players is incomplete: a player receives only the information that would be available to a real player in their position. Furthermore, each client may control only one player, and players are allowed to communicate only with limited virtual speech via the soccer server, so coordination among agents is difficult. The soccer server manual (Chen et al., 2001) states,

> One of [the] purposes of `soccerserver` is [the] evaluation of multi-agent systems, in which efficiency of communication between agents is one of the criteria. Users must realize control of multiple clients by such restricted communication.

The soccer server also incorporates complex game parameters (such as stamina and inertia), and adds noise to the movement of objects and to the players' senses.

RoboCup tournaments are held regularly, allowing researchers to demonstrate the effectiveness of their techniques by competing against other teams in a complex, noisy, real-time environment.

2.1 THE KEEPAWAY SUB-PROBLEM

While the *RoboCup* competition itself presents a valid application for machine learning, its complexity makes many machine learning experiments prohibitively time-consuming; better/faster experimental results can be expected for simpler problems.

One such sub-problem in *RoboCup* is the game of *keepaway*. In keepaway, one team (the *keepers* or *forwards*) starts with possession of the ball, and the other team (the *takers* or *defenders*) must take the ball from them. The game ends when a taker gets possession of the ball, or when the ball leaves the area of play. The keepers' objective is to maximise the duration of the game, while the takers' objective is the opposite.

Figure 1 shows a keepaway game in progress.

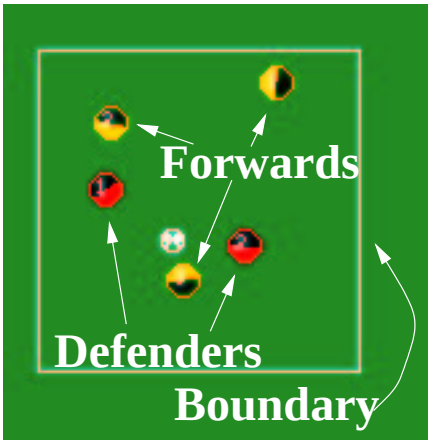


Figure 1: A 3 vs. 2 Keepaway Game In Progress (Diagram From Stone et al. (2001)).

2.2 MACHINE LEARNING IN KEEPAWAY

Thus far, the best *RoboCup* teams have been manually programmed agents (Reis and Lau, 2001; Stone and McAllester, 2001). These teams were created based on the assumption that because *RoboCup* is a simulation of real soccer, implementing strategies known to work in real soccer will result in a good *RoboCup* team.

However, this fails to account for the fundamental differences between *RoboCup* and real soccer, and the high-level differences that these create. Also, the *RoboCup* rules and soccer server are constantly changing, so maintaining a manually programmed team can be difficult. Hence the idea of applying machine learning techniques to *RoboCup* is gaining ground (e.g. (Luke et al., 1998; Andre and Teller, 1999))[1].

Stone et al. (2001) used reinforcement learning to train keepers for 3 vs. 2 keepaway (i.e. three keepers vs. two takers) on a 20m × 20m playing field. They used a software coach to set up games, enforce the rules, terminate the games, and provide feedback to the players. They defined the following high-level behaviour functions for keepers to use.

> **HoldBall():** Remain stationary while keeping possession of the ball and turning it such that it is as far away from the opponents as possible.
>
> **PassBall(*f*):** Kick the ball directly to teammate *f*.
>
> **GoToBall():** Intercept a moving ball or move directly to a stationary ball.
>
> **GetOpen():** Move to a position that is free from opponents and open for a pass from the ball's current position (using SPAR (Veloso et al., 1998)).

The keepers' decision-making process was simplified by using the policy space shown in Figure 2. The takers always called `GoToBall()` (or `HoldBall()` if they already had the ball). This meant that decisions were required only for the keeper in possession of the ball, and that the decision was a choice of three possible actions: `HoldBall()`, `PassBall(1)` and `PassBall(2)`.

Stone *et al.* also implemented three benchmark policies: making random decisions; always holding the ball; and a manually-coded policy that holds the ball unless a taker is within 10m *and* a safe pass is available. The reinforcement learning generated policies

[1] We describe here only applications of machine learning to keepaway.

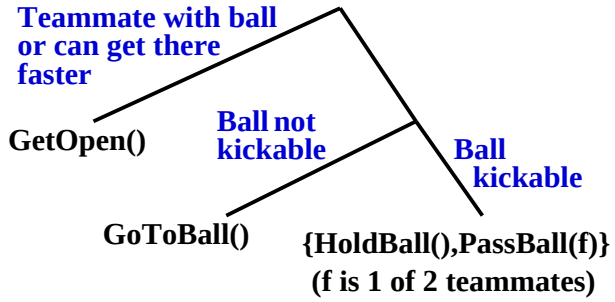


Figure 2: The Keepers' Policy Space (Diagram From Stone et al. (2001)).

that achieved game durations of 14 seconds, whereas all of the benchmark policies (including the manually-coded one) could manage durations of only 5 seconds.

Gustafson (2001) used genetic programming (Koza, 1992; Luke et al., 1998) to evolve keepers for 3 vs. 1 keepaway, using the *TeamBots* (Belch) simulator. Games were of a fixed duration and the keepers' objective was to minimise the number of turnovers per game. He also used somewhat different physics: the ball moved twice as fast as the takers, which moved twice as fast as the keepers.

Using his best layered learning genetic programming parameters, Gustafson produced a final population with a mean fitness of 70 turnovers, and a best fitness of 9 turnovers. These results are not easily comparable to Stone's because the problem domain and fitness evaluation are fundamentally different.

2.3 OUR KEEPAWAY SET-UP

Following Stone *et al.*, we generated keepers for 3 vs. 2 keepaway on a 20m × 20m playing field in the *RoboCup* soccer server. We reconfigured the soccer server to use 50ms cycles, i.e. to run at double speed, without otherwise changing the mechanics of the game. We based the takers on the freely-available *CMUnited-99* (Stone et al., 2000) source code that always called `GoToBall()` (or `HoldBall()` if it already had the ball).

We developed a software coach to set up games, enforce the rules, terminate games, and provide feedback to the players. A game ended when the ball went out of bounds, or when the ball had been within the kickable area of at least one taker for 5 or more consecutive server cycles. We introduced the additional constraint that if a game lasted for 600 server cycles (one minute of *RoboCup* game time), it would automatically be terminated. This constraint was added to impose a limit on degenerate luck in the noisy *RoboCup* environment.

3 EVOLUTIONARY ALGORITHMS

Evolutionary computation is a broad term that encompasses all methods of using the principles of biological evolution (Darwin, 1859) to solve problems on a computer.

We chose an evolutionary algorithm (EA) for this study because they are known to work well in noisy and unknown domains (Darwen, 2000). In an EA, the population is modelled as a set of *candidates*, and the parameters that vary between individuals are encoded into the candidates as genetic attributes. These attributes may be initialised with random or other values. The evolutionary process is then simulated to produce new candidates that represent better solutions.

To instantiate this technique for a particular problem, we have to specify the representation used for candidates' keepaway policies, the fitness function used to evaluate policies, and the operators and parameters used for reproduction, mutation and selection. We describe these aspects of the set-up in Sections 3.1–3.3.

3.1 EVOLUTIONARY OPERATORS

We used a $(1 + 1)$ evolutionary strategy. In each generation, each candidate produces one child, then the population members for the next generation are chosen from the combined parents and children populations.

Reproduction is simulated by creating a duplicate of the candidate with mutation. We implemented mutation using a Gaussian random distribution with zero mean difference. The standard deviation of the distribution determines how aggressively the system tries to evolve. We used a value of 0.1, resulting in about 18% of children surviving into the next generation.

The selection function determines which candidates will survive into the next generation , and which will die. We used a simple scheme where the n candidates with the highest fitness ratings survived (from the $2n$ parents+children available).

3.2 PROBLEM REPRESENTATION

We allowed our keepers the same high-level actions at each cycle as Stone's (see Section 2.2). However, because these high-level primitives (and the corresponding low-level skills) were independently implemented, our keepers could be expected to perform differently and to learn different strategies. Furthermore, whereas Stone *et al.* used SPAR (Veloso et al., 1998) for their `GetOpen()` function, we used a superior algorithm given in Stone and McAllester (2001).

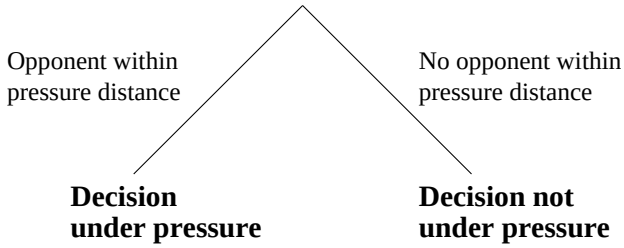


Figure 3: The Agents' Overall Decision-Making Framework.

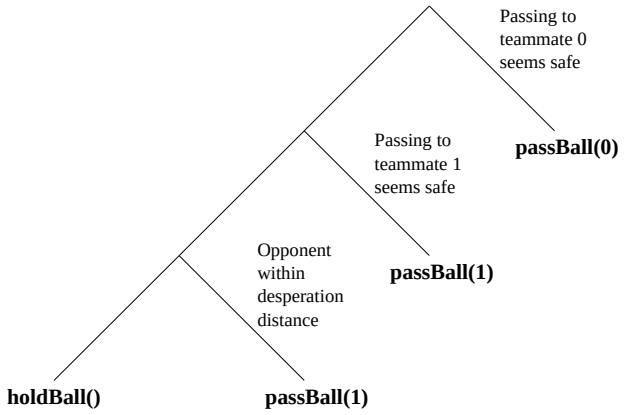


Figure 4: The Agents' Decision-Making Framework When Under Pressure.

Our keepers used the same policy space as Stone's (Figure 2), thus a decision was required only for the keeper in possession of the ball. We further decomposed this case using the decision-making framework shown in Figures 3, 4, 5. This is a simplistic initial framework that is likely to be extended in the future. Note that `teammate 0` is the teammate nearest to the ball and `teammate 1` is the other teammate.

If there is an opponent within the *pressure distance* (an evolved parameter), the agent acts "under pressure" (Figure 4); otherwise, it acts "not under pressure" (Figure 5).

When "under pressure", the agent wants to pass the ball. If passing to `teammate 0` seems "safe", it will do so; otherwise, it will consider passing to `teammate 1`. If neither pass seems "safe", the agent holds the ball, unless there is an opponent within the *desperation distance* (another evolved parameter), in which case it clears the ball by passing to `teammate 1`.

When "not under pressure", the agent will pass only if it can improve the strategic utility of the state by centralising the ball. It passes to the most central teammate that is "very safe" to pass to, if any.

A pass is deemed safe only if the following three values are all "large enough":

- the distance to the recipient;
- the minimum angle formed by the recipient, the ball, and each opponent; and
- the distance between the recipient and each opponent.

Each of these distances and angles is an evolved parameter. Thus the decision-making framework used a total of twelve evolved parameters:

- the pressure distance;
- the desperation distance;
- five parameters for assessing passes to `teammate 0`: two distances and an angle for acting "under pressure", plus one distance and an angle for acting "not under pressure"; and
- the same five parameters for `teammate 1`.

Each parameter is a real value in the range $[0, 1]$, representing a proportion of the range available. Distances were scaled by the maximum diagonal length of the field, and angles were scaled by 180° (the maximum absolute angle size). The only exception to this was that the desperation distance was scaled by the pressure distance. This forced the desperation distance to be less than the pressure distance, so that there was always some taker-proximity at which the agent would clear the ball.

3.3 FITNESS EVALUATION

The noisy environment of the *RoboCup* soccer server induces noisy fitness evaluations. Beyer (2000) states that coping with noisy fitness evaluations in an EA is still in its infancy. He describes three techniques, the simplest of which are increasing the population size, and resampling and averaging the fitness.

As the population size and the number of fitness samples/evaluation are increased, generations take longer to simulate. Thus one must choose a balance between population size, number of samples/evaluation, and number of generations simulated. Darwen (2000) claims that to obtain the best result from the available CPU time, one should use a "generously large" population, and that the number of fitness samples used should be just enough such that using more does not improve learning (clearly this is problem dependent).

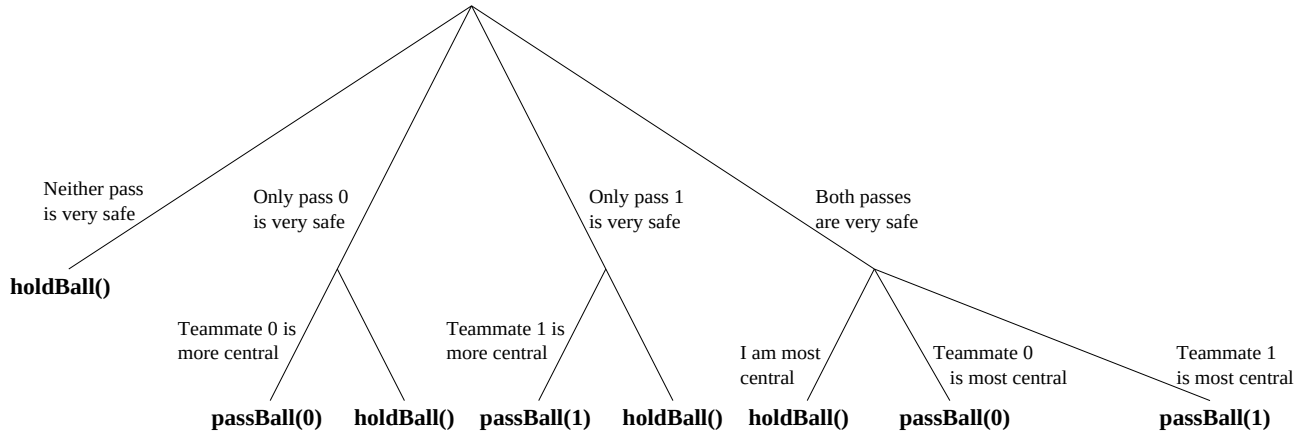


Figure 5: The Agents' Decision-Making Framework When Not Under Pressure.

The noisy fitness evaluations of keepaway forced us to reevaluate all candidates in each generation, otherwise a mediocre candidate might bias the population by receiving a fortuitously good fitness. However, this reevaluation is very expensive. If we use n samples/evaluation in a population of size p, and if each candidate is completely reevaluated in each generation, this requires a total of $2np$ samples/generation.

We used a different approach based on keeping a *moving average* of the candidates' samples. When a candidate is first generated (as a child), its fitness is sampled n times, and its *fitness array* is initialised with n copies of the average of these samples. In each subsequent generation, the fitness is sampled once only, and this new sample replaces the oldest sample remaining in its fitness array. The fitness estimate for a candidate in a given generation is the average of the samples in its fitness array at that time, ignoring outliers.

This technique reduces the number of samples/generation from $2np$ to $(n + 1)p$, allowing us to run nearly twice as many generations in a given run-time. Note that this technique is applicable only where candidates can persist in the population.

In addition to the noisy environment of the *RoboCup* soccer server, fitness evaluations are affected by the choice of team members. Fitness samples are assigned equally to all players on a team. A good player that is teamed with two inferior players is unlikely to achieve a good game duration; conversely, a bad player that is teamed with two superior players is likely to achieve a good game duration. To minimise this effect, before each round of reevaluation we randomised the order in which the candidates were scheduled for evaluation. Thus if a player's fitness was sampled n times in a generation, it would play in n randomly selected teams.

Each keepaway game starts with one keeper in each of three corners. Both takers are placed in the other corner, and the ball is dropped in a randomly-selected corner occupied by a keeper. This means that one of the keepers will always start between two teammates and opposite the takers. This creates the potential to evolve specialised roles depending on player position. To exploit this potential for specialisation, we used a different population for each of the three keepers. This meant using three small populations instead of one large population.

4 RESULTS AND COMPARISONS

Darwen (2000) suggests that two important parameters that effect EA performance in a noisy environment are population size and number of fitness samples/evaluation. Our first round of experiments focused on varying these parameters to determine the range of parameters which behaved the "best". We experimented with population sizes ranging from 3 to 200, with the number of fitness samples ranging from 1 to 200. We found that smaller population sizes did not maintain enough diversity to cope with the noisy fitness evaluations, while larger population sizes learnt more slowly with no apparent benefit. Similarly, using too few fitness samples resulted in a fitness approximation that was too noisy and produced erratic results, while using too many fitness samples resulted in slower learning with no apparent benefit. This round of experiments lead us to conclude that a population size of 10 to 40 should be used, with samples/evaluation between 5 and 10.

Our second round of experiments tested each of the population sizes 10, 20, 30, and 40, with 5, 7, and 9 fitness samples. We found that a population size

of 20 with 7 fitness samples learned fastest, but were concerned that the population size may be too small to maintain diversity (hence limiting further improvement), and to retain what had been learnt (exposing the population to the danger of a poor candidate being awarded an over-generous fitness due to limited sampling). We observed that in the experiments with a population size of 10, the populations almost always lost diversity after their performance levelled off, sometimes resulting in a degradation in fitness due a "lucky" candidate dominating the small population. Populations of size 20 occasionally exhibited this behaviour, but to a lesser extent, and with swifter recovery. Populations of size 30 never exhibited any significant problems with fitness retention. Populations of size 40 learnt more slowly with no apparent benefit.

Based on these experiments, we concluded that the fastest learning speed was obtained using a population size of 20 with 7 fitness samples, but that for a reliable balance between learning speed and fitness retention, a population size of 30 with 9 fitness samples should be used. We then ran a third round of experiments using population sizes of 20 and 30, with 7 and 9 fitness samples, over a large number of runs, to confirm that our results could be reliably reproduced.

Finally, in our fourth round of experiments, we took a typical run with a population size of 30 and 9 fitness samples, and retroactively evaluated each member of the initial and final populations over thousands of games to accurately estimate their actual fitnesses. Meanwhile, we ran experiments with population sizes of 40 with 9 fitness samples for a longer time to confirm that the runs with a population size of 30 with 9 fitness samples were finding good solutions.

Figure 6 shows a scatter-graph of the surviving candidates from each generation of one keeper population for a run using a population size of 30 and 9 fitness samples. We observed similar trends in the populations of the other two players. The lines represent the best (maximum), average, and worst (minimum) fitnesses of the surviving candidates of each generation. The time for the run was 53 hours, (67 generations).

The highest average fitness was approximately 300 cycles (30 seconds of *RoboCup* game time). This was attained within 25 generations, or 20 hours of running time, after which the average fitness levels off. The degradation in fitness from generations 30 to 55 is a result of noise in the fitness evaluations. The first generation in which the best fitness was greater than 300 cycles was generation 8, which corresponds to 6.4 hours of running time; however, because the fitness evaluation is noisy, it is likely that the "real" fitness of this candidate was lower.

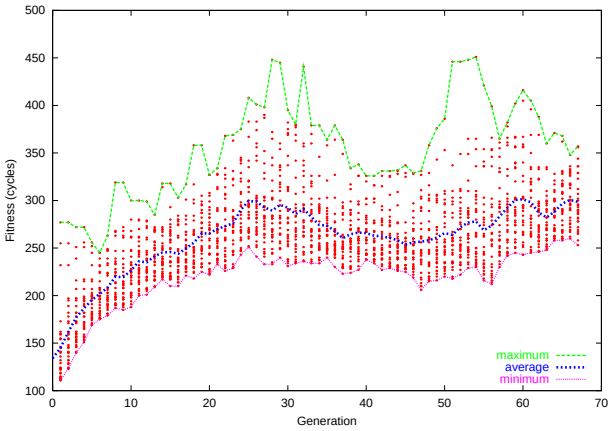


Figure 6: Fitness Versus Generations Using A Population Size Of 30 With 9 Fitness Samples.

The scatter-graph reveals that no candidates obtained an estimated fitness greater than 350 cycles in the first 15 generations. This is significant because it suggests that some learning is required before such high fitness estimates can be attained. Thus although there is an element of luck in playing keepaway, skill is an overriding factor in determining the game duration and hence a candidate's chance of survival (at least initially).

Figure 7 plots the distribution of the "accurately" estimated fitnesses for the initial and final populations from Figure 6. Note that the plotted accurately estimated fitness differs from the EA's coarse estimation (9 fitness samples); an accurate fitness estimate is determined retroactively by testing the candidate for thousands of fitness samples (to within ± 1 second).

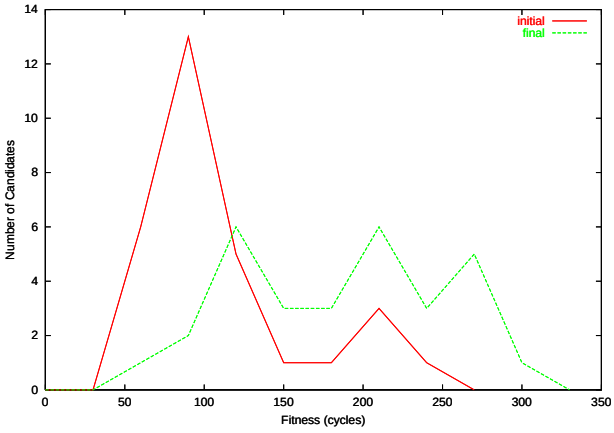


Figure 7: Fitness Distribution Of The Initial And Final Populations From Figure 6.

Retroactive testing of the final population showed that although the accurately estimated fitnesses of the surviving candidates were usually lower than the coarsely estimated fitnesses, there were several candidates with high accurately estimated fitnesses, and in some cases the accurately estimated fitness was higher than the coarsely estimated fitness. The candidate with the best accurately estimated fitness was rated the 6^{th} best member in the population by the EA. We call this candidate *Stuart*. We computed the 95% confidence interval for the mean fitness of this candidate as [319, 339] cycles. We call the second best candidate (using the accurately estimated fitness) in the final population *Bowser*, and the third best candidate *Vince*.

We also retroactively tested the initial (randomly generated) population to accurately estimate the average fitness before learning. The average fitness of the initial population was 128 cycles.

We were concerned about the high fitnesses of some of the members of the initial population, so we ran additional experiments (with a population size of 30 and 9 fitness samples) that began with a poor initial population. We confirmed that the same fitness values were learnt from the poor initial population, suggesting that our results are independent of the initial population.

Table 1 compares the results of this work with that of the reinforcement learning approach undertaken by Stone et al. We observe that Stone was able to improve game duration from approximately 5.5 seconds to 14.5 seconds (an improvement of 9.0 seconds) with approximately 20 hours of learning. Our results show improvement in the average game duration from 12.8 seconds to 24.8 seconds (an improvement of 12.0 seconds). Note this value is somewhat smaller than the estimated fitness determined by the EA – the accurate fitness estimate uses many more samples, with the EA including only those candidates that survive into the next generation (intuition suggests that surviving candidates will have been luckier than rejected candidates). The best member in the final population of the EA was found to have an accurately estimated game duration of 32.9 seconds. We expect that the differences in the static fixed strategies (always holding the ball and random) between the two works are due to differences between the underlying basic skills of the agents (`HoldBall()`, `GoToBall()`, and `GetOpen()`).

Figure 8 compares the strategies evolved by the players *Stuart*, *Bowser*, and *Vince*. It shows the frequency of hold durations between passes or turnovers – shorter hold durations mean that the player passes more often. We observe that *Bowser* and *Vince* are similar in that they both pass infrequently (approximately 4% of the time). However, investigation of these candidates shows that the EA has evolved a similar strategy using different representations. *Bowser* only passes when desperate; consequently, all his passes are clearances to his farther teammate. *Vince* however, can pass when not desperate, distributing his passes equally between his two teammates. *Stuart* however passes much more often, passing to both players. He uses strategic passes to centralise the ball when not under pressure.

Table 1: Comparison Of Our Results To Stone's.

Strategy	Stone et al.	This work
Always Hold	4.7	7.2
Random	4.9	13.5
Learning (Initial Average)	5.5	12.8
Learning (Final Average)	14.5	24.8

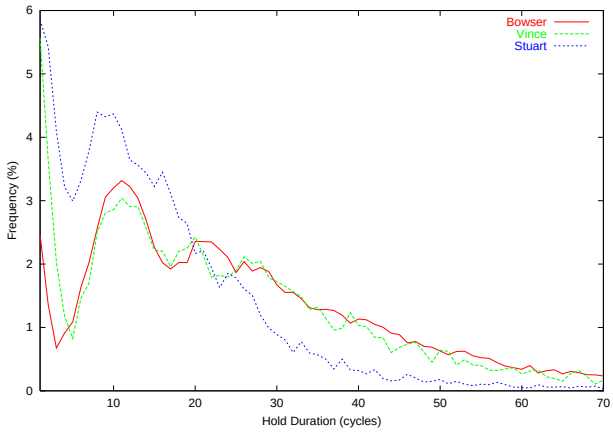


Figure 8: Frequency Of Hold Durations For *Stuart*, *Bowser*, and *Vince*.

5 CONCLUSIONS

We constructed an evolutionary algorithm to learn policies for playing the game of 3 vs. 2 keepaway, a sub-problem of the *RoboCup* soccer simulation league. We investigated the effects of the noisy fitness evaluation on the results from the system, and we used several techniques to alleviate this effect. In particular, we implemented a technique based on maintaining moving averages of fitness samples to enable us to use a good-sized population with several fitness samples/evaluation, yet still with a reasonable run-time.

Our results exceeded those from previous studies of this problem. Although the results are based on differ-

ent keeper implementations, the best previous learning system improved 9 seconds in its game duration, whereas the population in our system improved by 12 seconds. Although we ran our system for longer than in previous studies, in fact it achieved its best results well before the end of most experiments, so the learning is not as slow as it may appear at first sight.

The idea of using moving averages to improve the learning rate (in the real-time sense) in the context of noisy fitness evaluations may find uses in other applications. We expect that this technique can be used wherever individuals can persist in the population.

6 FUTURE WORK

We plan to extend this work in several directions, both within the *RoboCup* domain, and otherwise.

Within the domain of keepaway, we plan to improve the framework that agents use for representing decisions. We also aim to make the problem more complex (possibly by introducing a second objective), with the aim of encouraging specialisation in the separate player populations. We might also experiment with different population structures, e.g. using one population of players, or using a population of teams. We may also translate the work to other *RoboCup* subproblems, such as developing specific skills that exploit the physics of *RoboCup*.

We plan to further investigate the effects of noise, both in this domain and more generally. The use of moving averages has helped to improve the results, and we will apply the same technique in other domains to refine it further. Connected to this, we will also investigate other possible ways to make fitness evaluations based on multiple (noisy) samples, e.g. using the median of a group of samples, as opposed to the mean.

References

D. Andre and A. Teller. Evolving Team Darwin United. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer-Verlag, Berlin, 1999.

T. Belch. *TeamBots* software and documentation. URL `http://www.teambots.org/`.

H.-G. Beyer. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):239–267, 2000.

M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin. *RoboCup Soccer Server (Users Manual)*, June 2001.

P. J. Darwen. Computationally intensive and noisy tasks: Co-evolutionary learning and temporal difference learning on Backgammon. In *Proc. 2000 Congress on Evolutionary Computation*, pages 872–879, Piscataway, NJ, 2000. IEEE Service Center.

C. Darwin. *The Origin of Species*. Penguin Classics, London, 1859.

S. M. Gustafson and W. H. Hsu. Layered learning in genetic programming for a cooperative robot soccer problem. In *European Conference on Genetic Programming*, pages 291–301, 2001.

H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge. In *International Joint Conference on Artificial Intelligence (IJCAI97)*, 1997.

J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*. Springer-Verlag, Berlin, 1998.

L. P. Reis and N. Lau. FC Portugal team description: RoboCup 2000 simulation league champion. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer-Verlag, Berlin, 2001.

P. Stone and D. McAllester. An architecture for action selection in robotic soccer, 2001.

P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*. Springer-Verlag, Berlin, 2000.

P. Stone, R. S. Sutton, and S. Singh. Reinforcement learning for 3 vs. 2 Keepaway. In P. Stone, T. Balch, and G. Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 249–258. Springer-Verlag, Berlin, 2001.

M. Veloso, P. Stone, and M. Bowling. Anticipation: a key for collaboration in a team of agents: A case study in robotic soccer. *Proc. SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, 3839, September 1998.

Exploring Multiple Design Topologies Using Genetic Programming and Bond Graphs

Zhun Fan
Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824

Kisung Seo
Genetic Algorithms Research and Applications Group
Michigan State University
2857 W. Jolly Rd., Okemos, MI 48864

Ronald C. Rosenberg
Dept. of Mechanical Engineering
Michigan State University
East Lansing, MI 48824

Jianjun Hu
Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Erik D. Goodman
Genetic Algorithms Research and Applications Group
Michigan State University
2857 W. Jolly Rd., Okemos, MI 48864

Abstract

To realize design automation of dynamic systems, there are two major issues to be dealt with: open-topology generation of dynamic systems and simulation or analysis of those models. For the first issue, we exploit the strong topology exploration capability of genetic programming to create and evolve structures representing dynamic systems. With the help of ERCs (ephemeral random constants) in genetic programming, we can also evolve the sizing of dynamic system components along with the structures. The second issue, simulation and analysis of those system models, is made more complex when they represent mixed-energy-domain systems. We take advantage of bond graphs as a tool for multi- or mixed-domain modeling and simulation of dynamic systems. Because there are many considerations in dynamic system design that are not completely captured by a bond graph, we would like to generate multiple solutions, allowing the designer more latitude in choosing a model to implement. The approach in this paper is capable of providing a variety of design choices to the designer for further analysis, comparison and trade-off. The approach is shown to be efficient and effective in an example of open-ended real-world dynamic system design application, a printer re-design problem.

1 INTRODUCTION

In general, design of dynamic systems includes two steps: conceptual design and detailed design. In the conceptual design phase, the following questions should be answered (Tay *et al*. 1998): 1) What is the exact design problem to be solved? (This requires a complete and consistent listing of the requirements), and 2) what are the key problem areas in the solution? (This requires the identification of critical parts of the solution that will determine the performance). Then the process of detailed design can be undertaken, identifying those candidate solutions that meet the requirements and provide the level of performance needed. The research in this paper focuses on the detailed design of dynamic systems. The strategy is to develop an automated procedure capable of exploring the search space of candidate dynamical systems and providing design variants that meet desired design specifications or dynamical characteristics. The method must be able to explore the design space in a topologically open-ended manner, yet still find appropriate configurations efficiently enough to be useful.

Much research has been done on design automation of single domain systems using an evolutionary computation approach. For example, automated design of analog circuits has attracted much attention in recent years (Grimbleby, 1995) (Lohn, 1999) (Koza, 1999) (Zhun, 2000). It could be classified into two categories: GA-based and GP-based. Most GA-based approaches realize topology optimization via a GA and parameter optimization with numerical optimization methods (Grimbleby, 1995). Some GA approaches also evolve both topology and component parameters; however, they typically allow only a limited number of components to be evolved (Lohn, 1999). Although their work basically achieves good results in analog circuit design, it is not easily extendable to interdisciplinary systems like mechatronic systems.

Design of interdisciplinary (multi-domain) dynamic engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behavior (Coelingh *et al.*). However, most current modeling and simulation tools that provide for representation at a

schematic, or topological, level have been optimized for a single domain. The bond graph provides a unified model representation across inter-disciplinary system domains. Tay uses bond graphs and GA to generate and analyze dynamic system designs automatically (Tay *et al*. 1998). He uses nested GA to evolve both topology and parameters for dynamic systems. However, the efficiency of his approach is hampered by the weak ability of GA to search in both topology and parameter spaces simultaneously.

Genetic programming is an effective way to generate design candidates in an open-ended, but statistically structured, manner. There have been a number of research efforts aimed at exploring the combination of genetic programming with physical modeling to find good engineering designs. Perhaps most notable is the work of Koza *et al.*. He presents a single uniform approach using genetic programming for the automatic synthesis of both the topology and sizing of a suite of various prototypical analog circuits, including low-pass filters, operational amplifiers, and controllers. This approach appears to be very promising, having produced a number of patentable designs for useful artifacts. It is closely related to our approach, except that it searches in a single energy domain.

We investigate an approach combining genetic programming and bond graphs to automate the process of design of dynamic systems to a significant degree. To improve the topology search capability of GP and enable it to provide a diversity of choices to the designer, a special form of parallel GP, the Hierarchical Fair Competition GP (HFC-GP), is used in this paper (Hu, *et al*., 2002). The efficiency and effectiveness of the approach are illustrated in an interesting redesign example involving the drive mechanism for an electric printer. Several design alternatives for the printer drive are derived through exploring open-topologies in bond graph space. It turns out that some of them are obviously physically realizable and others are not.

2 DESIGN DOMAIN AND METHODOLOGY

2.1 MULTI-DOMAIN DYNAMIC SYSTEMS

Multi-domain system design differs from conventional design of electronic circuits, mechanical systems, and fluid power systems in part because of the need to integrate several types of energy behavior as part of the basic design. For example, in addition to appropriate "drivers" (sources), lumped-parameter dynamical mechanical systems models typically include at least masses, springs and dampers (Figure 1 a)) while "RLC" electric circuits include resistors, inductors and capacitors (Figure 1 b)). However, they could both be expressed in the same bond graph (Figure 1 c)).

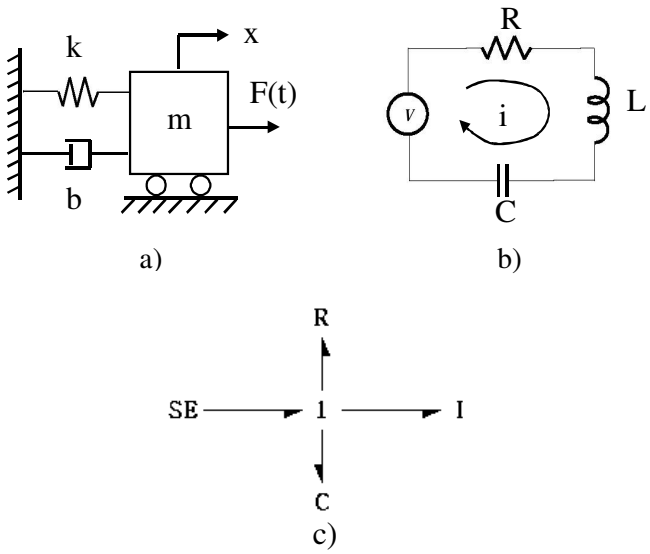


Figure 1. Dynamic systems and bond graph representation : a) mechanical, b) electrical , and c) bond graph that represents both

2.2 BOND GRAPHS

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic, etc. (Karnopp *et al.* 2000). It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design searching. For notation details and methods of system analysis related to the bond graph representation see Karnopp *et al.* and Rosenberg (Rosenberg *et al,* 1992). Much recent research has explored the bond graph as a tool for design (Sharpe and Bracewell 1995, Tay *et al*. 1998, Youcef-Toumi 1999, Redfield 1999).

In our research, the bond graph has additional desirable characteristics for selection as the tool for system representation and simulation. The evaluation efficiency of the bond graph model can be improved because analysis of causal relationships and power flow between elements and subsystems can be done quickly and easily, and reveals certain important system properties and inherent characteristics. This makes it possible to discard infeasible design candidates even before numerically evaluating them, thus reducing time of evaluation to a large degree. Because virtually all of the circuit topologies passing causal analysis can be simulated, our system does not need to check validity conditions of individual circuits to avoid singular situations that could interrupt the running of a program evaluating them.

Another characteristic of bond graphs is their ease of mapping to the engineering design process (Xia, *et al.* 1991). Because each component of the system can be represented correspondingly in a bond graph, junctions and elements can be added to or deleted from a model without causing dramatic changes. This emulates the

engineering process of modifying systems, refining simple designs discovered initially, adding size and complexity as needed to meet more complicated design demands step by step. As genetic programming usually shows a weak causality of structure evolution (Rosca, 1995), this potential strong causality of the bond graph modification process also makes bond graph representation an attractive technique to use in genetic programming to explore the open-ended dynamic system design space in an evolutionary process.

2.2 GENETIC PROGRAMMING AND BOND GRAPHS

The tree representation on GP chromosomes, as compared with the string representation typically used in GA, gives GP more flexibility to encode solution representations for many real-world design applications. The bond graph, which can contain cycles, is not represented directly on the GP tree—instead, the function set (nodes of the tree) encode a constructor for a bond graph.

We define the GP functions and terminals for bond graph construction as follows. There are four types of functions: first, *add* functions that can be applied only to a junction and which add a C, I, or R element; second, *insert* functions that can be applied to a bond and which insert a 0-junction or 1-junction into the bond; third, *replace* functions that can be applied to a node and which can change the type of element and corresponding parameter values for C, I, or R elements; and fourth, *arithmetic* functions that perform arithmetic operations and can be used to determine the numerical values associated with components (Table 1). Details of function definitions are illustrated in Seo *et al*. (2001).

Table 1 Function and terminal set for bond graph evolution

Name	Description
add_C	Add a C element to junctions
add_I	Add an I element to junctions
add_R	Add an R element to junctions
insert_J0	Insert a 0-junction in bond
insert_J1	Insert a 1-junction in bond
replace_C	Replace current element with C element
replace_ I	Replace current element with I element
replace_ R	Replace current element with R element
+	Add two ERCs
-	Subtract two ERCs
endn	End terminal for add element operation
endb	End terminal for insert junction operation
endr	End terminal for replace element operation
erc	Ephemeral random constant (ERC)

2.3 DESIGN PROCEDURE

The flow of the entire algorithm is shown in Figure 2. The user specifies the embryonic physical model for the target system (*i.e.,* its interface to the external world, in terms of which the desired performance is specified) After that, an initial population of GP trees is randomly generated. Each GP tree maps to a bond graph tree. Analysis is then performed on each bond graph tree. This analysis consists of two steps – causal analysis and state equation analysis. After the (vector) state equation is obtained, the important dynamic characteristics of the system are sent to the fitness evaluation module and the fitness of the tree is evaluated. For each evaluated and sorted population, genetic operations – selection, crossover, mutation and reproduction – are carried out to seek design candidates with improved quality. The loop of bond graph analysis and GP operation is iterated until a termination condition is satisfied or a specified number of iterations performed. The final step is to instantiate a physical design, replacing the bond graph with the physical components represented.

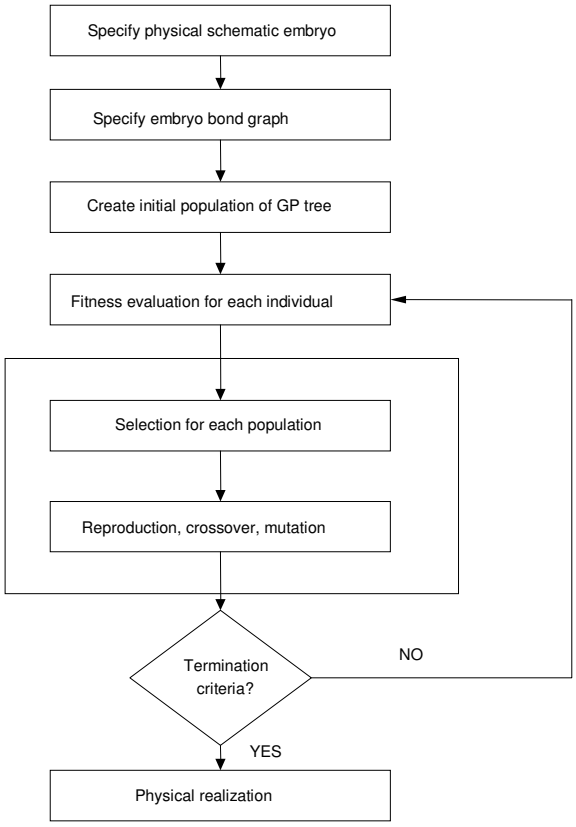


Figure 2. Flow chart of the design procedure

3 CASE STUDY

3.1 PROBLEM FORMULATION

The original problem was presented by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Figure 3 shows a closed-loop control system to position a rotational load

(inertia) denoted as J_L. The problem with the design is that the position output of the load J_L has intense vibrations (see Figure 4). The design specification is to reduce the vibration of the load to an acceptable level, given certain command conditions for rotational position.

We want the settling time to be less than 70ms when the input voltage is stepped from zero to one. Note that the settling time of the original system is about 2000ms. The time scale in Figure 4 is 4000 ms.

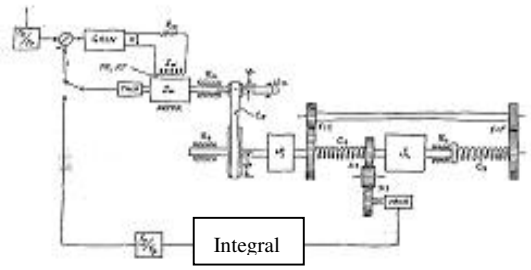


Figure 3. Schematic of the printer drive system

The system includes electric voltage source, motor and mechanical parts. As it is a multi-domain system, a bond graph is convenient to use for modeling.

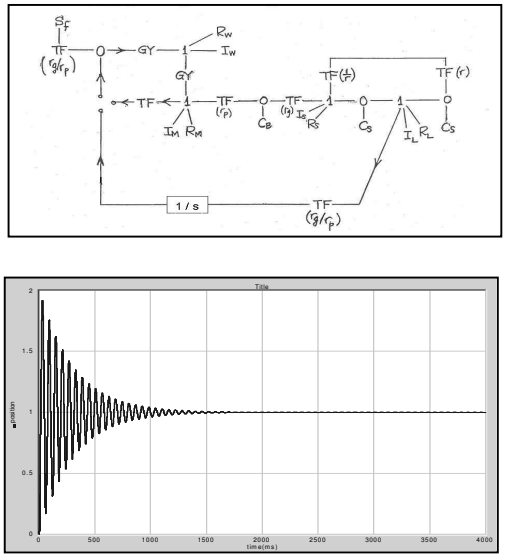


Figure 4. a) Bond graph model b) Positional vibration of the load

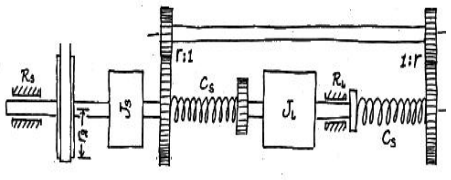


Figure 5. The embryo subsystem

By analyzing the model, we conclude that the critical part for the design is a subsystem that involves the drive shaft and the load (Figure 5). The input is the driving torque, T_d, generated through the belt coupling back to the motor (not shown).

This subsystem was deemed a logical place to begin the design problem. The questions left to the designer now are: 1) at which exact spots of the subsystem new components should be inserted, 2) which types of components and how many of them should be inserted, in which manner, and 3) what should be the values of the parameters for the components to be added? The approach reported in this paper is able to answer these three questions in one stroke in an automated manner, once the embryo system has been defined.

3.2 AN EMBRYO FOR EVOLUTION

To search for a new design using the BG/GP design tool, an embryo model is required. The embryo model is the fixed part of the system and the starting point for GP to generate candidates of system designs by adding new components in a developmental manner. The embryo used for this example, expressed in bond graph language, is shown in Figure 6, with the modifiable sites highlighted. The modifiable sites are places that new components can be added. The choice of modifiable sites is typically easy for the designer to decide. However, modifiable sites are only *possible* spots for insertion of new components – they are not necessarily inserted to any particular one of them. In this particular example, designers need have no idea whether assemblies of new components will be inserted at modifiable site (1), or at modifiable site (2) , at site(3), or at any combinations of them. Instead, the algorithm will answer these questions in an automatic way, without intervention by the human designer.

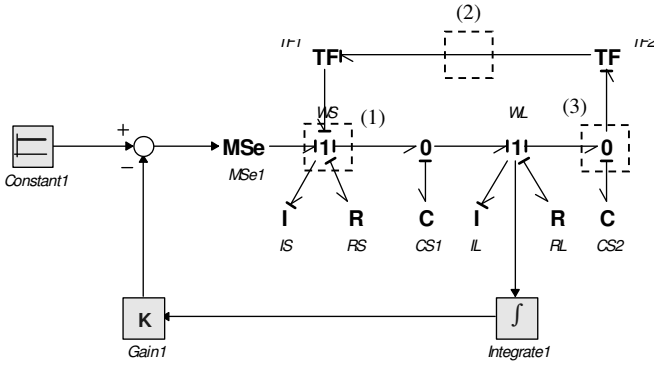


Figure 6. Bond graph model for the embryo system

The parameters for the embryo model are:

$I_s : 6.7 \times 10^{-6}\ kg \cdot m^2$

$R_s : 0.013 \times 10^{-3}\ N \cdot m \cdot \sec \,/\, rad$

$C_{s1} : 0.208\ \ N \cdot m \cdot \,/\, rad$

$C_{s2} : 0.208\ \ N \cdot m \cdot \,/\, rad$

R_L: $0.58 \times 10^{-3}\ N \cdot m \cdot \sec \,/\, rad$

I_L: $84.3 \times 10^{-6}\ kg \cdot m^2$

3.3 THE HFC MODEL OF PARALLEL GENETIC PROGRAMMING

A special form of parallel GP, HFC-GP is applied in this research. In the HFC (Hierarchical Fair Competing) model (Fig 7), multiple subpopulations are organized in a hierarchy, in which each subpopulation can only accommodate individuals within a specified range of fitnesses (Hu *et al*, 2002). New individuals are created continuously in the bottom layer. Use of the HFC model balances exploration and exploitation of GP effectively. Our experience shows that using the HFC model can also substantially increase the topological diversity of the whole population and help to provide the designer a diverse set of competing design candidates for further trade-offs.

3.4 DEFINITION OF FITNESS FUNCTION

The fitness function of individual design is defined

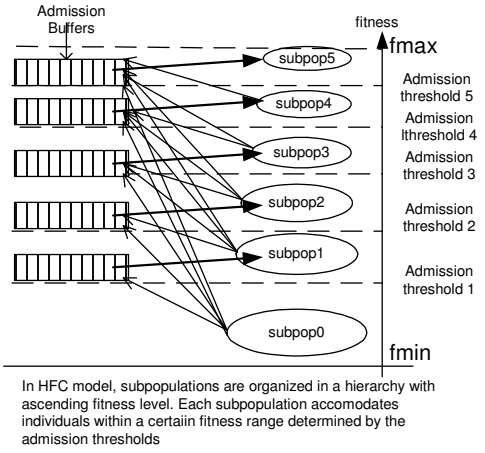


Figure 7. Hierarchical Fair Competition Model in GP

according to the position output response of the load J_L as follows.

Within the time range of interest (0~500ms in this example), uniformly sample 1000 points of the output response (yielding a time interval between two adjacent sampling points of 0.5ms). Compare the magnitudes of the position output of the load J_L at the sample points with target magnitudes (unity in this example), compute their difference and get a squared sum of differences as raw fitness, defined as $Fitness_{raw}$. Then calculate normalized fitness according to:

$$Fitness_{norm} = 0.5 + 1000 \,/\left(2000 + Fitness_{raw}\right)$$

It can be assumed approximately that the higher the normalized fitness, the better the design. Two reasons make the fitness definition an approximate one: 1) it does not reflect directly the strict definition of settling time, and 2) it does not include other considerations in design of the system except output response. A modified fitness function could be defined later if required. However, in this research, the definition is enough to manifest the feasibility and efficiency of the approach reported. The design results achieved (Figures 9-16) show performances satisfying the design specification presented in this research.

3.5 EXPERIMENTAL SETUP

We used a strongly-typed version [Luke, 1997] of lilgp [Zongker and Punch, 1996] to generate bond graph models. The major GP parameters were as shown below:

Number of generations: 500
Population size: 500
Initial population: half_and_half
Initial depth: 4-6
Max depth: 16
Max nodes: 1000
Selection: tournament (size=7)
Crossover: 0.8
Mutation: 0.2

Three major code modules were created in our work. The algorithm kernel of HFC-GP was a modified version of an open software package developed in our research group -- lilgp. A bond graph class was implemented in C++. The fitness evaluation package is C++ code converted from Matlab code, with hand-coded functions used to interface with the other modules of the project. The commercial bond graph software package 20Sim was used to verify the dynamic characteristics of the evolved design.

The GP program obtains satisfactory results on a Pentium-IV 1GHz in 5~15 minutes, which shows the efficiency of our approach in finding good design candidates.

3.6 EXPERIMENTAL OBSERVATIONS

The fitness improvement curve of GP algorithm is plotted versus generation number in Figure 8.

Three competing design candidates with different topologies, as well as their performances and possible physical realizations, are provided in Figures 9-16. We can see from the output rotational position responses that they all satisfy the design specification of settling time less than 70ms. Note that the time scale of the plots is 100 ms.

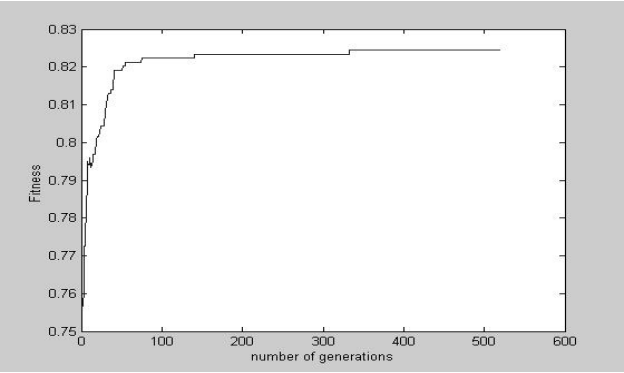


Figure 8. Fitness Improvement Curve

Design variant 1 is represented in Figure 9. Two new components (R, C) are added with a 0-junction at modifiable site (1). The parameters of the components added are also given. Dashed circles highlight the newly evolved components in the bond graph figures. Figure 10 displays rotational position output for variant 1.

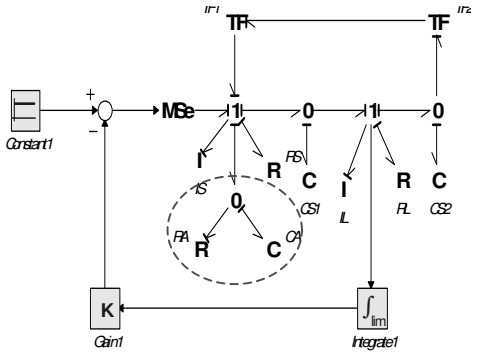


Figure 9. Bond graph model of design variant 1

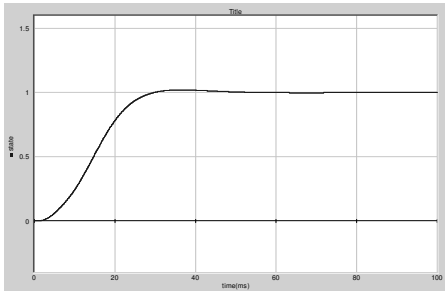


Figure 10. Position output of design variant 1

Figure 11 gives one possible physical realization of design variant 1. One damper and one spring are connected to the embryo model.

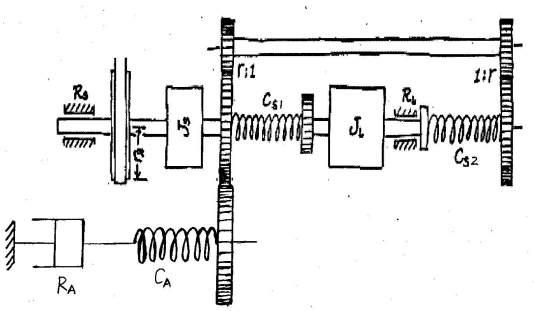


Figure 11. Physical realization of design variant 1

Variant 2 adds new components to modifiable site (2) and modifiable site (3) as shown in Figure 12. Figure 13 displays rotational position output for variant 2.

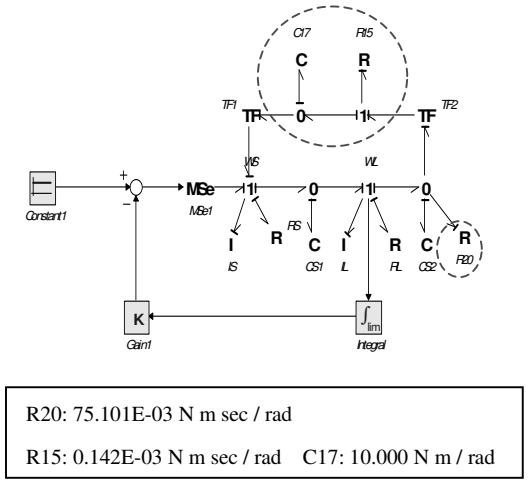


Figure 12. Bond graph model of design variant 2

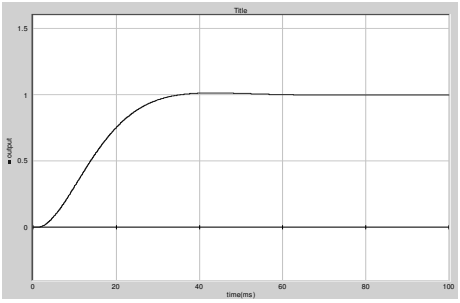


Figure 13. Position output of design variant 2

Figure 14 gives one possible physical realization of design variant 2.

Design variant 3 is represented in Figure 15. Variant 3 adds new components to modifiable site (1) and modifiable site (2). Figure 16 displays rotational position output for variant 3. The bond graph model of variant 3 is not obviously physically realizable, because component I is attached to a 0-junction.

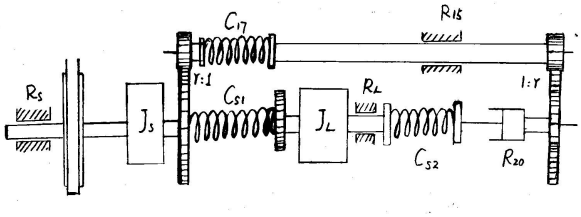


Figure 14. Physical realization of design variant 2

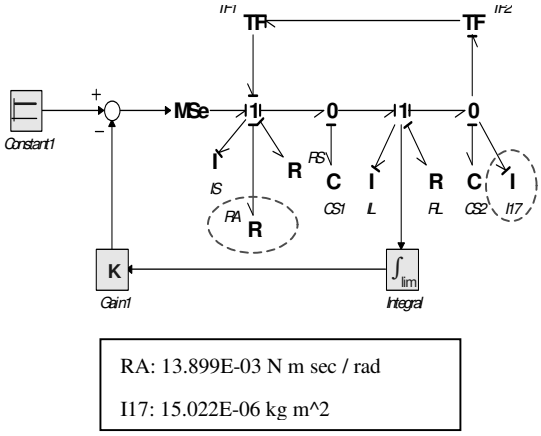


Figure 15. Bond graph model of design variant 3

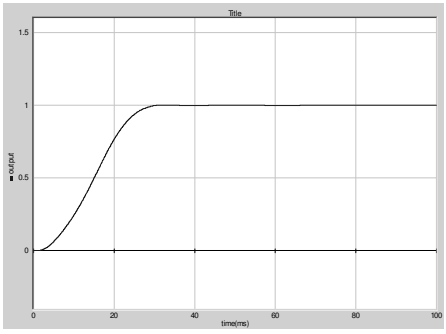


Figure 16. Position output of design variant 3

It is clear that the approach reported in this research is both efficient and effective, capable of providing designers with a variety of design alternatives. This gives designers considerable flexibility to generate and to compare a large variety of design schemes.

4 CONCLUSIONS

This research has explored a new automated approach for synthesizing designs for dynamic systems. By taking advantage of genetic programming as a search method for competent designs and the bond graph as a representation for dynamic systems, we have created a design environment in which open-ended topological search can be accomplished in an automated and efficient manner and the design process thereby facilitated.

The paper illustrates the process of using this approach in detail through a printer redesign problem. Bond graphs have proven to be an effective tool for both modeling and design in this problem. A special form of parallel GP, the Hierarchical Fair Competition-GP, has been shown to be capable of providing a diversity of competing designs with great efficiency.

We plan to continue our research by identifying improved methods for minimizing the occurrence of unrealizable bond graph designs. One such approach is to use multi-objective evolutionary computation to shape the results. A second approach is to use a set of revised GP operators to build bond graphs that avoid unrealizable models.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation through grant DMI 0084934.

References

H. J. Coelingh, T. de Vries, and J. Amerongen (1998) Automated Performance Assessment of Mechatronic Motion Systems during the Conceptual Design Stage. *Proc. 3rd Int'l Conf. on Adv. Mechatronics*, Okayama, Japan.

Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, and B. Zhang (2001). Bond Graph Representation and GP for Automated Analog Filter Design, *Proceedings of the Genetic and Evolutionary Computation Conference*: 81-86.

J. B. Grimbleby (2000). Automatic analogue circuit synthesis using genetic algoriths. *IEE Proc. – Circuits Devices Syst.* : 319-323.

J. Hu, E. D. Goodman (2002). The Hierarchical Fair Competition (HFC) Model for Parallel Evolutionary Algorithms. *Congress on Evolutionary Computation* .

D. C. Karnopp, D. L. Margolis and R. C. Rosenberg (2000). *System Dynamics: Modelling and Simulation of Mechatronic Systems. Third Edition*. New York: John Wiley & Sons, Inc.

J. R. Koza (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press.

J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane (1999b). The design of analogue circuits by means of genetic programming. In P. J. Bentley (ed.), *Evolutionary Design by Computers*, 365-385. London: John Wiley & Sons Ltd.

J. D. Lohn, S. P. Colombano (1999). A circuit representation techniques for automated circuit design. *IEEE Transactions on Evolutionary Computation*: 205-219.

S. Luke, 1997, *Strongly-Typed, Multithreaded C Genetic Programming Kernel,* http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.

H. M. Paynter (1991). An epistemic prehistory of bond graphs. In P. C. Breedveld and G. Dauphin-Tanguy (ed.), *Bond Graphs for Engineers,* 3-17. Amsterdam, The Netherlands: Elsevier Science Publishers.

R.C. Redfield (1999). Bond Graphs in Dynamic Systems Designs: Concepts for a Continuously Variable Transmission. *International Conference on Bond Graph Modeling and Simulation*: 225-230.

J. P. Rosca , D. H. Ballard (1995), Causality in genetic programming. In L. Eshelman (ed.), Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), 256-263. San Francisco, CA: Morgan Kaufmann.

R.C. Rosenberg, J. Whitesell, and J. Reid (1992). Extendable Simulation Software for Dynamic Systems. *Simulation* 58: 175-183.

K. Seo, E. D. Goodman and R. C. Rosenberg (2001). First steps toward automated design of mechatronic systems using bond graphs and genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference* : 189.

J.E. Sharpe , and R.H. Bracewell (1995). The Use of Bond Graph Reasoning for the Design of Interdisciplinary Schemes. *International Conference on Bond Graph Modeling and Simulation:* 116-121.

E. H. Tay, Woodie Flowers and John Barrus (1998). Automated Genration and Analysis of Dynamic System Designs. *Research in Engineering Design* 10: 15-29.

S. Xia, D. A. Linkens and S. Bennett (1991). Integration of qualitative reasoning and bond graphs: an engineering approach. In P. C. Breedveld and G. Dauphin-Tanguy(ed.), *Bond Graphs for Engineers,* 323-332. Amsterdam, The Netherlands: Elsevier Science Publishers.

K. Youcef-Toumi, Y. Ye., A. Glaviano, and P. Anderson (1999). Automated Zero Dynamics: Derivation from Bond Graph Models. *International Conference on Bond Graph Modeling and Simulation:* 39-44.

D. Zongker and W.F. Punch, III, 1998, *lil-gp 1.1 User's Manual*, GARAGe, College of Engineering, Michigan State University.

An Immunogenetic Technique to Detect Anomalies in Network Traffic

Fabio A. González* and Dipankar Dasgupta
Intelligent Security Systems Research Lab, Computer Science Division
The University of Memphis
Memphis, TN 38152
*and Universidad Nacional de Colombia
Bogotá, Colombia
{fgonzalz, ddasgupt}@memphis.edu

Abstract

The paper describes an immunogenetic approach which can detect a wide variety of intrusive activities on networked computers. In particular, this technique is inspired by the negative selection mechanism of the immune system that can detect foreign patterns in the complement (non-self) space. The novel pattern detectors (in the complement space) are evolved using a genetic search, which could differentiate varying degrees of abnormality in network traffic. The paper demonstrates the usefulness of such a technique in intrusion/anomaly detection. A number of experiments are performed using intrusion detection data sets (DARPA IDS evaluation program) and tested for validation. Some results are reported along with their analysis and concluding remarks.

1 INTRODUCTION

There are many approaches to anomaly detection, and most of them work on building a model or profile of the system that reflects its normal behavior. A simple approach is to define thresholds (upper and lower) for each monitored parameter of the system, and if a parameter exceeds this range, it is considered an abnormality. The most common approach uses a statistical model (Denning, 1986; Eskin, 2000) to calculate the probability of occurrence of a given value, lesser the probability, higher is the possibility of an anomaly. In general, statistical approaches model individually the different variables that represent the state of the system, which could make it difficult to detect complex multivariable temporal patterns.

Other approaches also build models to predict the future behavior of systems or processes based on the present and past states (Crosbie and Spafford, 1995; Dagupta and Gonzalez, 2001). Accordingly, if the actual state of the system differs considerably from the predicted state, an anomaly alarm is raised. These approaches are more successful in capturing temporal and multiple variable correlations. However, more time is needed for training the model, and in some cases its application can be infeasible because of the size of data sets involved (generally, this is the case with network security).

Therefore, the challenge is to build an anomaly detection system that can capture multi-variable correlations, and is capable of dealing with large amounts of data generated in a computer network environment. Data mining techniques have been applied with some success to this problem (Lane, 200; Lee and Stolfo, 1998). This approach has the advantages of dealing with large data sets and being able to get useful knowledge (generally expressed in terms of rules). For these techniques, it is important that the data have some degree of structure. In several works, the network traffic data (packet level) is processed to get connection information (type, duration, number of bytes transmitted, etc). In some cases, it is necessary to have the information regarding whether a connection is normal or it is an attack.

This paper proposes an approach that does not rely on structured representation of the data and uses only normal data to build a profile of the system. Although, it is applied to perform anomaly detection for network security, it is a general approach that can be applied to different anomaly detection problems.

The technique is inspired by artificial immune systems ideas, and it attempts to extend Forrest's (self/non-self) (Forrest et al., 1994) two-class approach to multiple classes. Specifically, the non-self space will be further classified in multiple sub-classes to determine

the level of abnormality. The core of the technique is a genetic algorithm that evolves rules to cover the abnormal space (non-self). Experiments are performed and the results are compared with the ones produced by a positive characterization technique.

2 BACKGROUND AND PREVIOUS WORKS

The idea of using immunological principles in computer security (Dasgupta, 1999; Hofmeyr and Forrest, 2000; Kephart, 1994) started since 1994. Stephanie Forrest and her group at the University of New Mexico have been working on a research project with a long-term goal to build an artificial immune system for computers. In their approach, the problem of protecting computer systems from harmful viruses was viewed as an instance of the more general problem of distinguishing *self* (legitimate users, uncorrupted data, etc.) from the dangerous *other* (unauthorized users, viruses, and other malicious agents). This method (called the *negative-selection* algorithm (Forrest et al., 1994)) was used to detect changes in the protected data and program files. In another work, they applied the algorithm to monitor UNIX processes where the purpose is to detect harmful intrusions in a computer system. Kephart (Kephart, 1994) suggested another immunologically inspired approach (decoy program) for virus detection. In this approach, known viruses are detected by their computer-code sequences (signatures) and unknown viruses by their unusual behavior within the computer system.

In (Kim and Bentley, 2001), the negative-selection algorithm is evaluated as a mechanism to perform network intrusion detection. The conclusion is that the algorithm presents "severe scaling problems for handling real traffic data", and it is suggested to use it only as a filter for invalid detectors. In the present work, we show that it is possible to overcome these issues and generate effective detectors by changing the representation scheme.

3 ANOMALY DETECTION PROBLEM DEFINITION

The purpose of anomaly detection is to identify which states of a system are normal and which are abnormal. The states of a system can be represented by a set of features. Accordingly,

Definition 1. System states space. A state of the system is represented by a vector of features, $x^i = (x_1^i, ..., x_n^i) \in [0.0, 1.0]^n$. The space of states is represented by the set $S \subseteq [0.0, 1.0]^n$. It includes the feature vectors corresponding to all possible states of the system.

The features can represent current and past values of system variables. The actual values of the variables could be scaled or normalized to fit a defined range $[0.0, 1.0]$.

Definition 2. Normal subspace (crisp characterization). A set of feature vectors, $Self \subseteq S$ represents the normal states of the system. Its complement is called Non_Self and is defined as $Non\_Self = S - Self$. In many cases, we will define the $Self$ (or Non_Self) set using its characteristic function $\chi_{self} : [0.0, 1.0]^n \rightarrow \{0, 1\}$

$$\chi_{self}(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in Self \\ 0 & \text{if } \vec{x} \in Non\_Self \end{cases}$$

The terms *self* and *non-self* are motivated by the natural immune system. In general, there is no sharp distinction between the normal and abnormal states, instead there is a degree of normalcy (or conversely, abnormality). The following definition tries to reflect this:

Definition 3. Normal subspace (non-crisp characterization). The characteristic function of the normal (or abnormal) subspace is extended to take any value within the interval $[0.0, 1.0] : \mu_{self} : [0.0, 1.0]^n \rightarrow [0.0, 1.0]$. In this case, the value represent the degree of normalcy: 1 indicates normal, 0 indicates abnormal, and intermediate values represent elements with some degree of abnormality[1].

The non-crisp characterization allows a more flexible distinction between normalcy and abnormality. However, in a real system it is necessary to decide when to raise an alarm. In this case, the problem becomes again a binary decision problem. It is easy to go from the non-crisp characterization to the crisp one by establishing a threshold:

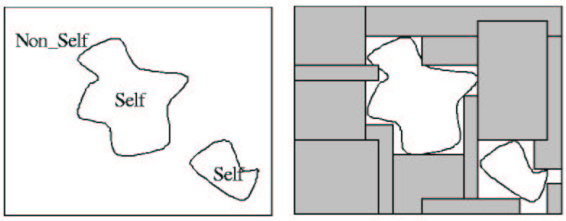
$$\mu_{self,t}(\vec{x}) = \begin{cases} 1 & \text{if } \mu_{self}(\vec{x}) > t \\ 0 & \text{if } \mu_{self}(\vec{x}) \leq t \end{cases}$$

[1] This definition is basically a fuzzy set specification. In fact, the function μ_{self} is a membership function. However, we chose not to refer it directly, because of the different emphasis of this work.

Definition 4. Anomaly detection problem. Given a set of normal samples $Self' \subseteq Self$, build a good estimate of the normal space characteristic function χ_{self} (or μ_{self} in the non-crisp case). This function should be able to decide whether the observe state of the system is anomalous or not.

4 PROPOSED APPROACH

The original negative selection algorithm proposed by Forrest (Forrest et al., 1994) used binary encoding for representing self/non-self strings. In a previous work (Dasgupta and Gonzalez, 2002), a real-valued representation to characterize the descriptor space (self/non-self space) was proposed. A genetic algorithm with sequential niching scheme was used to evolve detectors on the complement (non-self) space (as shown on Figure 1). The present work proposes another niching technique (deterministic crowding (Mahfoud, 1992)) to evolve the non-self-covering detectors, which appears to perform better in covering the complement space.



(a) (b)

Figure 1: (a) Self and non-self division of the descriptor space (b) Approximation of the non-self space by interval rules.

The detectors correspond to hyper-rectangles in a multidimensional space. These detectors are represented by rules with the following structure:

$$\begin{array}{lll} R^1\text{: If} & Cond_1 & \text{then } \mathbf{non\_self} \\ \vdots & \vdots & \vdots \\ R^m\text{: If} & Cond_m & \text{then } \mathbf{non\_self} \end{array}$$

where,

- $Cond_i = x_1 \in [low_1^i, high_1^i]$ and $\ldots$ and $x_n \in [low_n^i, high_n^i]$
- $(x_1, ..., x_n)$ is a feature vector
- $[low_i^j, high_i^j]$ specifies the lower and upper values for the feature x_i in the condition part of the rule R^j.

The condition part of each rule defines a hypercube in the descriptor space ($[0.0, 1.0]^n$). Then, a set of these rules tries to cover the non-self space with hypercubes. For the case $n = 2$, the condition part of a rule represents a rectangle. Figure 1.b illustrates an example of this kind of coverage for $n = 2$.

The non-self characteristic function (crisp version) generated by a set of rules $R = \{R^1, ..., R^m\}$ is defined as follows:

$$\chi_{non\_self,R}(\vec{x}) = \begin{cases} 1 & \text{if } \exists R^j \in R \text{ such that } \vec{x} \in R^j \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{x} \in R^j$ means that $\vec{x}$ satisfies the condition part of the rule R^j.

A rule is considered good if it does not include positive samples and covers a large area. This criteria guides the evolution process performed by the genetic algorithm.

As was discussed previously, a good characterization of the abnormal (non-self) space should be non-crisp. Then, the non-self space is further divided into different levels of deviation. We can think of these levels as concentric regions around the self zones. The farther a level is from the self, the more abnormal it is.

In order to characterize the different levels of abnormality, we considered a variability parameter (called v) to the set of normal descriptors samples, where v represents the level of variability that we allow in the normal (self) space. A higher value of v means more variability (a larger self space); a lower value of v represents less variability (a smaller *self* space). Figure 2 shows two sets of rules that characterize self spaces with a large and small value of v. Figure 2.a shows a covering using a small variability parameter v. Figure 2.b shows a covering using a larger value of v. The variability parameter can be assumed as the radius of a hyper-sphere around the self samples. Figure 2.c shows the levels of deviation defined by the two coverings.

In the non-self space, we use a genetic algorithm with different values of v to generate a set of rules that can provide complete coverage. In general, a set of rules has the following structure:

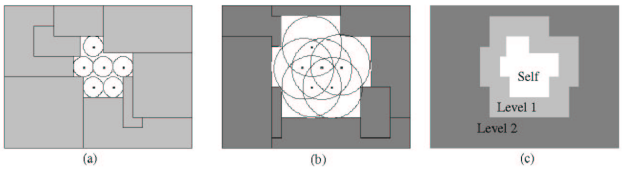


Figure 2: A set of normal samples is represented as points in 2-D space. The circle around each sample point represents the allowable deviation. (a) Rectangular rules cover the non-self (abnormal) space using a small value of v. (b) Rectangular rules cover the non-self space using a large value of v. (c) Level of deviation defined by each v, where level 1 corresponds to non-self cover in (a) and level 2 corresponds to non-self cover in (b)

$$
\begin{array}{rlll}
R^1\text{: If} & Cond_1 & \text{then } \textbf{Level 1} \\
\vdots & \vdots & \vdots \\
R^i\text{: If} & Cond_i & \text{then } \textbf{Level 1} \\
R^{i+1}\text{: If} & Cond_{i+1} & \text{then } \textbf{Level 2} \\
\vdots & \vdots & \vdots \\
R^j\text{: If} & Cond_j & \text{then } \textbf{Level 2} \\
\vdots & \vdots & \vdots
\end{array}
$$

The different levels of deviation are organized in a hierarchical way such that, the level 1 contains the level 2, the level 2 contains the level 3, and so forth. This means that a descriptor can be matched by more than one rule, but the highest level reported will be assigned. This set of rules generates a non-crisp characteristic function for the non-self space:

$$\mu_{non\_self}(\vec{x}) = \max\{l \mid \exists R^j \in R\ ,\ \vec{x} \in R^j \text{and } l = level(R^j)\} \cup \{0\},$$

where $level(R^j)$ represent the deviation level reported by the rule R^j.

4.1 GENETIC ALGORITHM FOR DETECTOR GENERATION

The purpose of the genetic algorithm is to evolve 'good' rules to cover the non-self space. In general, one rule is not enough, instead, a set of rules that solve the problem cooperatively is necessary. In our previous work, we used a genetic algorithm combined with a sequential niching technique (Beasley et al., 1993). That approach was useful in evolving good detector rules. The main drawback of that approach is that the genetic algorithm must be run multiple times to generate multiple rules. The approach proposed by this paper, uses a niching technique, deterministic crowding (Mahfoud, 1992), that allows the generation of multiple rules in a single run.

The input to the GA is a set of n-dimensional feature vectors $S = \{x^1, ..., x^m\}$, which represents samples of the normal behavior of the parameter, the number of different levels of deviation (*numLevels*), and the allowed variability for each level $\{v_1, ..., v_{numLevels}\}$. The algorithm is shown as follows:

---

for $i = 1$ **to** *numLevels*
 initialize population with random individuals
 for $j = 1$ to *numGenerations*
  for $k = 1$ to *population_size*/2
   select two individuals with uniform probability
    and without replacement
   apply crossover to generate a child
   mutate the child
   if *dist(child,parent1)* $<$ *dist(child,parent2)*
     and *fitness(child)* $>$ *fitness(parent1)*
    substitute parent1 with child
   elseif *dist(child,parent1)* $\geq$ *dist(child,parent2)*
     and *fitness(child)* $>$ *fitness(parent2)*
    substitute parent2 with child
   endif
  endfor
 endfor
 extract the best individuals from the population
 and add them to the final solution
endfor

Each individual (chromosome) in the genetic algorithm represents the condition part of a rule, since the consequent part is the same for all the rules (the descriptor belongs to non-self). However, the levels of deviation in non-self space are considered by the variability factor (v_i) that is used by the fitness function.

The condition part of the rule is determined by the low and high limits for each dimension. The chromosome that represents these values consists of an array of floats. The crossover operator used is a uniform crossover and the mutation operator is Gaussian mutation.

4.1.1 Fitness Evaluation

Given a rule R with condition part ($x_1 \in [low_1, high_1]$ **and** ... **and** $x_n \in [low_n, high_n]$), we say that a feature vector $x^j = (x_1^j, ..., x_n^j)$ satisfies the rule (represented for $x^j \in R$) if the hyper-sphere with center x^j and radius v_i intercepts the hyper-rectangle defined by the points $(low_i, ..., low_n)$ and $(high_1, ..., high_n)$.

The fitness of a rule is calculated taking into account the following two factors:

- The number of elements in the training set S, that belongs to the subspace represented by the rule:

$$num\_elements(R) = \{x^i \in S \mid x^i \in R\}$$

- The volume of the subspace represented by the rule:

$$volume(R) = \prod_{i=1}^{n} (high_i - low_i)$$

The fitness is defined as:

$$fitness_R = volume(R) - C \cdot num\_elements(R)$$

where, C is the coefficient of sensitivity. It specifies the amount of penalization that a rule suffers if it covers normal samples. The bigger the coefficient, the higher is the penalty value. The fitness can also take negative values.

4.1.2 Individual's Distance Calculation

A good measure of distance between individuals is important for deterministic crowding niching, since it allows the algorithm to replace individuals with closer individuals. This allows the algorithm to preserve the forming niches.

The distance measure used in this work is the following:

$$dist(c, p) = \frac{volume(p) - volume(p \cap c)}{volume(p)},$$

where,

- c : child individual
- p : parent individual

Note that the distance measure is not symmetric. The idea is that we give more importance to the area of the parent that is not covered. The justification is as follows: if the child covers a high proportion of the parent that means that the child is a good generalization of it, but if the child covers only a small portion, then it is not so.

5 EXPERIMENTATION AND RESULTS

5.1 TESTING DATA

We performed experiments with intrusion data obtained from the MIT-Lincoln Lab (MIT-Lincoln-Labs, 1999). These data represents both normal and abnormal information collected in a test network, where some simulated attacks were performed. The purpose of the data is to test the performance of intrusion detection systems. The data sets (corresponding to the year 1999) contain complete weeks with normal data (not mixed with attacks). This allows us to get enough samples to build the self profile and generate detectors.

The test data set is composed of network traffic data (tcpdump, inside and outside network traffic), audit data (bsm) and file systems data. For our experiments, we used only the outside tcpdump network data for a specific computer (e.g., hostname: marx), then we applied the tool *tcpstat* to get traffic statistics. We used the first week's data for training (attack free), and the second week's data for testing, which include some attacks. Some of these were network attacks the others were inside attacks. Only the network attacks are considered for our testing. The attack time line is shown in Figure 3.

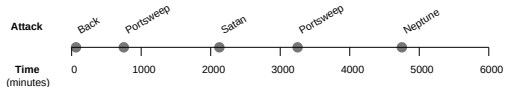


Figure 3: Network attacks on the second weekend

Three parameters were selected to detect some specific type of attacks. These parameters were sampled each minute (using *tcpstat*), and normalized. Table 1 lists six time series S_i and T_i for training and testing, respectively.

The set S of normal descriptors is generated from a time series $R = \{r_1, r_2, ..., r_n\}$ in an overlapping sliding window fashion:

$$S = \{(r_1, ..., r_w), (r_2, ..., r_{w+1}), ..., (r_{n-w+1}, ..., r_n)\},$$

where w is the window size. In general, from a time series with n points, a set of $n-w+1$ of w-dimensional

Table 1: Data sets and parameters used

Name	Description	Week	Type
S1	# of bytes / sec	1	Training
S2	# of packets / sec	1	Training
S3	# of ICMP packets / sec	1	Training
T1	# of bytes / sec	2	Testing
T2	# of packets / sec	2	Testing
T3	# of ICMP packets / sec	2	Testing

descriptors can be generated. In some cases, we used more than one time series to generate the feature vectors. In those cases, the descriptors are put side-by-side in order to produce the final feature vector.

5.2 EXPERIMENTAL SETTINGS

We used as the training set the time series $S1$, $S2$ and $S3$, and as the testing set the time series $T1$, $T2$ and $T3$, with a window size of 3. This means that the size of the feature vectors was 9.

The parameters for the genetic algorithm were: population size 200, number of generations 2000, mutation rate 0.1, and coefficient of sensitivity: 1.0 (high sensitivity).

The genetic algorithm was run with radius equal to 0.05, 0.1, 0.15 and 0.2 respectively. Then the elements in the testing set are classified using rules generated for each level (radius). This process is repeated 10 times and the results reported correspond to the average of these runs.

In order to evaluate the ability of the proposed approach to produce a good estimation of the level of deviation, we implemented a simple (but inefficient) anomaly detection mechanism. It uses the actual distance of an element to the nearest neighbor in the *Self* set as an estimation of the degree of abnormality. Formally, the characteristic function of the *non-self* set is defined as:

$$\begin{aligned} \mu_{non\_self}(\vec{x}) &= D(\vec{x}, Self) \\ &= \min\{d(\vec{x}, \vec{s}) : \vec{s} \in Self\} \end{aligned}$$

Here, $d(x, s)$ is a Euclidean distance metric (or any Minkowski metric[2]). $D(\vec{x}, Self)$ is the nearest neighbor distance, that is, the distance from x to the closest point in *Self*.Then, the closer an element $\vec{x}$ is to the self set, the closer the value of $\mu_{non\_self}(\vec{x})$ is to 0.

In the section results and discussion, we refer to this technique as *positive characterization* (PC) approach, to differentiate it from the proposed approach that we call *negative characterization* (NC) approach.

[2]In our experiments, we also used the D_∞ metric defined by: $D_\infty(\vec{x}, \vec{y}) = \max(|x_1 - y_1|, ..., |x_n - y_n|)$

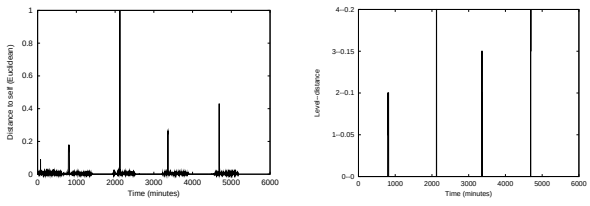
5.3 RESULTS AND DISCUSSION

Table 2: Number of generated rules for each deviation level

Level	Radius	Avg. Num. Rules	
		Seq. Niching	Det. Crowding
1	0.05	19.5	7.75
2	0.1	20.7	8.25
3	0.15	26	10
4	0.2	28	10

Table 2 shows the number of rules generated by the genetic algorithm for the previous technique (NC with sequential niching) and the new one (NC with deterministic crowding). The new technique produces less rules. This suggests the possibility that the new technique is discarding some good rules and therefore ignoring some niches. However, the performance of the two set of rules is the same. The conclusion is that the new technique is able to find a set of rules that is more compact but without compromising on performance. This can be explained by the fact that sequential niching is more sensitive to the definition of the distance between individuals than deterministic crowding.

Another notable point is the efficiency of the new technique. The new technique only needed four runs (one per level) to generate this set of rules. For the previous technique, it is necessary to run the GA as many times as the number of rules we want to generate. This is a clear improvement on computational time.



(a) PC output (b) NC output

Figure 4: Detection function $\mu_{non\_self,t}(\vec{x})$ generated by PC and PC applied to testing set : (a) deviation in testing set reported by PC (b) deviations in the testing set detected by evolved rule set.

Figure 4(a) shows the output of the PC algorithm when applied to the testing data. The five peaks correspond to the 5 attacks. Figure 4(b) shows a typical

attack profile produced by the application of the genetic generated rules to the testing set. Four of five attacks are detected.

As was shown in our previous work (Dasgupta and Gonzalez, 2002), the negative characterization (NC) is clearly more efficient (in time and space) compared to the positive characterization (PC). There seems to be a trade-off between compactness of the rule set representation and accuracy. Validity of these arguments are observed on our results. Figure 5 shows how the true positives rate changes according to the value of the threshold t. The PC technique has better performance than the NC, but only by a small margin. In general, the NC technique shows detection rates similar to the more accurate (but more expensive) PC technique. Table 3 summarizes the best true positive rates (with a maximum false alarm of 1%) accomplished by the two techniques.

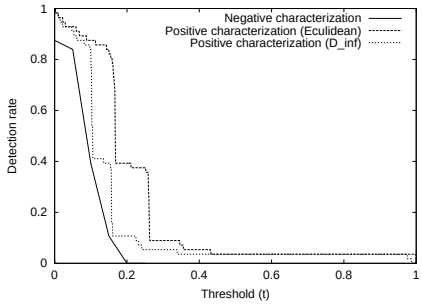


Figure 5: Comparison of the true positives rate of the detection function $\mu_{non\_self,t}(\overrightarrow{x})$ generated by positive characterization (PC) and negative characterization (NC) for different values of t.

Table 3: Best true positive rates for the different techniques with a maximum false alarm rate of 1%.

Detection Technique	**True Positive Rate**
Positive Charact. (Euclidean)	96.4%
Positive Charact. (D_∞)	92.8%
Negative Characterization	87.5%

Our previous work (Dasgupta and Gonzalez, 2002) shows that the NC technique produces a good estimate of the level of deviation when this is calculated using D_∞ distance. Table 4 shows the confusion matrix for the NC technique using sequential niching and deterministic crowding. For each element in the testing set, the function $\mu_{non\_self}(\overrightarrow{x})$ generated by the NC is applied to determine the level of deviation. This level of deviation is compared with the distance range reported by the PC algorithm (using D_∞ distance). Each row (and column) corresponds to a range or level of deviation. The ranges are specified on square brackets. A perfect output from the NC algorithm will generate only values in the diagonal.

Table 4: Confusion matrix for PC and NC reported deviations. The values of the matrix elements correspond to the number of testing samples in each class. The diagonal values represent correct classification.

PC output level	**NC output level**				
	seq. niching				
	0	1	2	3	4
1: [0.0,0.05]	5132	0	0	0	0
2: [0.05,0.1]	3	7.8	0.2	0	0
3: [0.1,0.15]	0	18.1	3.9	0	0
4: [0.15,0.2]	0	0	6.9	9.5	0.6
5: [0.2,..]	0	0	0	1	9
	det. crowding				
	0	1	2	3	0
1: [0.0,0.05]	5132	0	0	0	0
2: [0.05,0.1]	3	4	4	0	0
3: [0.1,0.15]	0	0	22	0	0
4: [0.15,0.2]	0	0	0	17	0
5: [0.2,..]	0	0	0	0	10

In the two cases, the values are concentrated around the diagonal. The two techniques produced a good estimate of the distance to the self set. However, the NC approach with deterministic crowding appears to be more precise. One possible explanation of this performance difference seems to be the fact that the sequential niching requires derating the fitness function for each evolved rule. This arbitrary modification in the fitness landscape can prevent evolving better rules.

6 CONCLUSIONS

In this paper, we investigated a technique to characterize and identify different intrusive activities by analyzing network traffic. The technique is based on artificial immune systems ideas and uses a genetic algorithm to generate good anomaly detectors rules. We used a real world data set (MIT-Lincoln lab) that has been used by other researchers for testing. The following are some preliminary observations:

- Our approach measures the anomaly of a system as the distance of a descriptor vector to a normal profile represented by descriptors collected during the normal operation. This approach appears to be very useful, since it was able to detect attacks in real test data set.

- The immunogenetic algorithm was able to produce good detectors that give a good estimation of the amount of deviation from the normal. This shows that it is possible to apply the negative selection algorithm to detect anomalies on real network traffic data. The real representation of the detectors was very useful in this work.
- The proposed algorithm is efficient; it was able to detect four of the five attacks detected by the positive characterization (with a detection rate of 87.5% and a maximum false alarms rate of 1%), while only using a fraction of the space (when compared to positive characterization).
- The use of deterministic crowding as niching technique improved the results obtained using sequential niching. While keeping the performance in terms of a high detection rate, the new algorithm generated a smaller set of rules that estimated in a more precise way the amount of deviation. The new technique is also more efficient in terms of computational power since it is able to evolve multiple rules for each individual run of the GA.

As part of our ongoing research we are exploring different covering strategies of the non-self space (for instance, using hyper-spheres), developing new algorithms to generate non-self covering rules and experimenting with other intrusion detection data sets.

Acknowledgments

This work was funded by the Defense Advanced Research Projects Agency (contract no. F30602-00-2-0514) and National Science Foundation (grant no. IIS-0104251).

References

Beasley, D., Bull, D. R., and Martin, R. R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125.

Crosbie, M. and Spafford, E. (1995). Applying genetic programming to intrusion detection. In Siegel, E. V. and Koza, J. R., editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8, MIT, Cambridge, MA, USA. AAAI.

Dagupta, D. and Gonzalez, F. (2001). *Information Assuarance in Computer Networks*, chapter An intelligent decision support system for intrusion detection and response, pages 1–14. Lecture Notes in Computer Science. Springer-Verlag.

Dasgupta, D. (1999). *Artificial Immune Systems and Their Applications*. Springer-Verlag, New York.

Dasgupta, D. and Gonzalez, F. (2002). An immunity-based technique to characterize intrusions in computer networks. *To appear in IEEE Transactions on Evolutionary Computation*, 6(2).

Denning, D. (1986). An intrusion-detection model. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 118–31.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conf. on Machine Learning*, pages 255–262. Morgan Kaufmann, San Francisco, CA.

Forrest, S., Perelson, A., Allen, L., and Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Proc. IEEE Symp. on Research in Security and Privacy*.

Hofmeyr, S. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473.

Kephart, J. (1994). A biologically inspired immune system for computers. In *Proceedings of Artificial Life*, pages 130–139, Cambridge, MA.

Kim, J. and Bentley, P. J. (2001). An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1330–1337, San Francisco, California, USA. Morgan Kaufmann.

Lane, T. (200). *Machine Learning Techniques For The Computer Security*. PhD thesis, Purdue University.

Lee, W. and Stolfo, S. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX.

Mahfoud, S. W. (1992). Crowding and preselection revisited. In Männer, R. and Manderick, B., editors, *Parallel problem solving from nature 2*, pages 27–36, Amsterdam. North-Holland.

MIT-Lincoln-Labs (1999). Darpa intrusion detection evaluation. http://www.ll.mit.edu/IST/ideval/index.html.

DESIGN OPTIMIZATION OF N-SHAPED ROOF TRUSSES

Karim Hamza
Ph.D. Pre-Candidate
University of Michigan
Ann Arbor, MI 48109-2102
khamza@engin.umich.edu

Haitham Mahmoud
Ph.D. Pre-Candidate
University of Michigan
Ann Arbor, MI 48109-2102
ham@engin.umich.edu

Kazuhiro Saitou
Assistant Professor
University of Michigan
Ann Arbor, MI 48109-2102
kazu@engin.umich.edu

Abstract

Design optimization of a class of plane trusses called the N-Shaped Truss (NST) is addressed. The parametric model of NST presented is intended for real-world application, avoiding simplifications of the design details that compromise the applicability. The model, which includes twenty-seven discrete variables concerning topology, configuration and sizing of the truss, presents a challenging optimization problem. Aspects of such challenge include large search space dimensionality, absence of a closed-form objective function and constraints, multi-modal objective function and costly CPU time per objective function evaluation. Three implementations of general-purpose genetic algorithms (GA) are tested for this problem, along with a version of taboo search called reactive taboo search (RTS). The RTS exhibited better performance than the tested versions of GA. Performance study of the algorithms provides some good insight to some weaknesses in GA and RTS as well as future prospective combination of them to gain better performance.

1 INTRODUCTION

Truss structure optimization is a problem that is attractive due to its direct applicability in design of structures. Optimization of trusses can be classified into three main categories i) sizing, ii) configuration and iii) topology. This classification is slightly different from that of continuum structures, given in (Chapman *et. al.*, 1993) In the sizing optimization, cross-sectional areas of members in the truss are design variables and the coordinates of the nodes and connectivity are held constant (Goldberg 1986). The sizing problem is made even more interesting and practical through restricting the choice of truss members to a discrete set of available standard cross-sections (Rajeev and Krishnamoorthy, 1992). In configuration optimization, the member cross-sections and connectivity (*i.e*. topology) remain constant, but the nodal position locations are the design variables. In topology optimization, the connectivity is the objective of the optimization (Bendose and Kikuchi 1988) and (Jakiela *et. al.*, 2000). Combining the categories has also been performed. Gil and Andreu (2001) combined the configuration and sizing problems. Deb and Gulati (2001) combined topology and sizing through real coded genetic algorithms. A fully connected ground structure is taken as a start, then during optimization, members having close to zero cross-sectional areas are then deleted.

Optimization methods applied included gradient-based methods such as the work of (Taylor and Rossow 1976) and (Kirsch 1979), simulated annealing (Moh and Chiang 2000) and genetic algorithms. Analytical methods have generally been limited by approximations due to the complexity of the real-world problem, which is nonlinear and often has no closed form objective function and constrains.

To the best of the authors' knowledge, most previous work was directed to developing optimization models for general trusses rather on a "high-level," without going deep into the design details of the truss. In this paper, a particular class of plane trusses (N-Shaped) is considered. While restricted to that class of trusses, the parametric model formulated goes deep into the design details and combines all truss optimization categories of sizing, configuration and topology. The optimization problem has a large search space which makes direct exhaustive search methods totally impractical. In addition, structural optimization problems are known to have many local optima, which encourages the use of heuristic global optimizers. Three implementations of genetic algorithms (GA) are tested as well as reactive taboo search (RTS) which also seems to be an attractive global optimizer (Battiti and Tecchiolli 1994).

The paper starts with a review of truss optimization then proceeds to describe the parametric model of the N-shaped truss. Following the description of the parametric model, the implemented GA and RTS are presented, then an actual real-life truss is used as a bench-mark problem to compare the performance of the optimizers. Results and discussion are then presented.

2 PARAMETRIC MODEL OF NST

2.1 TERMINOLOGY

Some of the terminology used in practice for the design of trusses is to be used in this paper. The following is a quick summery of such terminology:

- An N-Shaped Truss (NST): is a plane truss (Fig.1) that has a certain general shape resembling the letter "N."
- Upper Chord: are all the inclined members on the top part of the truss (Fig. 2). All upper chord members of an N- Shaped Truss form one straight line.
- Lower Chord: are all the horizontal members on the lower part of the truss (Fig. 2). All lower chord members of an N- Shaped Truss form one horizontal straight line.
- Vertical Members: are (as the name suggests), the vertical members in the truss (Fig. 2).
- Diagonal Members: are those internal inclined members (Fig. 2).
- Truss Projection: is the distance the truss protrudes after the centerline of the carrying column (Fig. 2).
- Bays: Are the spans between the trusses in the top view (Fig. 2).
- End Bay: is a last bay in a building.
- Purlins: are light members positioned across the bays and are carried on top of the upper chord (Figs. 2-3). Purlins, in turn carry the roof cladding.
- Roof braces: are X-shaped sets of members (Fig. 2) that are present in some bays in order to increase the overall structure stiffness.
- Longitudinal Braces: are sets of members across the bays that are included to increase the overall rigidity of the structure (Figs. 2-3).

Figure 1: Photo of Actual N-Shaped Truss

2.2 DESIGN VARIABLES

Twenty seven variables that a designer can modify are used as design variables in this parametric model. The design variables are categorized into i) variables concerned with topology and configuration and ii) variables concerned with sizing of the truss members. The design variables are given as:

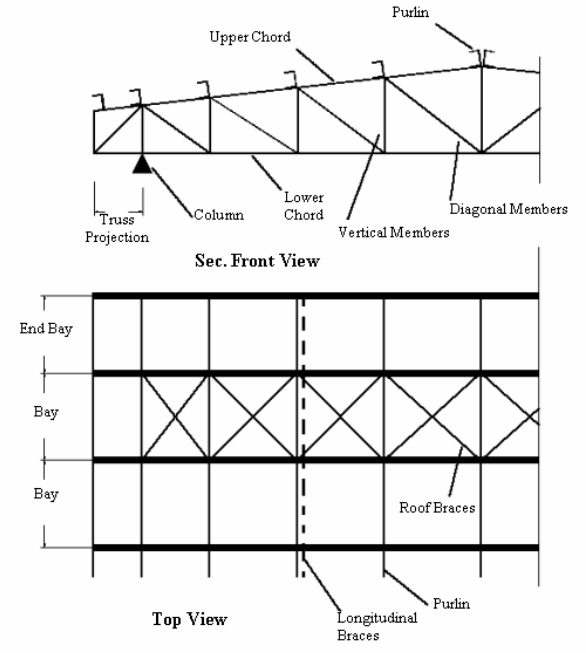


Figure 2: Typical N-Shaped Truss

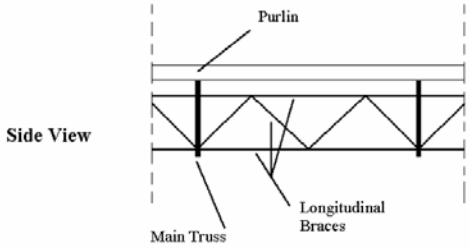


Figure 3: Longitudinal Braces

Topology and Configuration Variables:

X1: is an integer number that defines the selected roof layout plan (from up to 5 user-defined choices), this variable subsequently sets the number of main bays, bays' lengths and end-bays' lengths.

X2: is the length of the vertical member directly on top of the support. Normally, this variable is continuous, but it is discretized in this model to avoid the necessity of using mixed integer/continuous optimizers. However, discretization doesn't impose much deviation from practicality, since the fabrication often favors "rounded-off" and similar dimensions.

X3: is the number of purlins on top of the truss. This normally dictates the general truss topology, since every purlin must have a vertical member in the truss underneath it. The space between two purlins (or their two verticals) will be referred to as a truss "cell".

X4: is the number of sub-divided truss cells near the support. Subdividing the cells near the support (the portion which has low truss depth as opposed to the middle part of the truss), generally improves the angle of the diagonal members which in turns gives better distribution of the axial forces in the members.

X5: is the number of truss cells which have reinforced diagonal members. Normally, the diagonals closer to the support are subjected to higher axial loads, therefore it is often efficient to choose a different cross section for the first one or few diagonal members.

X6: is the number of merged cells near the middle of the truss. The purpose of merging cells at the portion of bigger truss depth is also to improve the angle of the diagonals to give better stress distribution.

X7: is the number of verticals that are nearer to the column and are taking a different cross-section than the rest of the verticals.

The model also allows for two different configurations of longitudinal braces to be used, thus the longitudinal braces passing near the mid-span (with higher depth) may be different from those passing above the support.

X8: is the total number of longitudinal braces lines across the roof.

X9: is half the number of longitudinal braces lines close to the support (Type-1).

X10: is the number of nodes (equal to number of cells minus one) on Type-1 Longitudinal Braces.

X11: is the number of nodes on Type-2 Longitudinal Braces.

Member Sizing Variables:

X12 to X27 are integer variables defining the selected standard cross-section from the available database for 16 groups of truss members. The truss member groups are: purlins, main truss upper chord, lower chord, 3 groups of verticals, 4 groups of diagonals, longitudinal braces 2 groups of chords, 2 groups of verticals and 2 groups of diagonals.

It should be noted that the truss members grouping employed in this parametric model keeps the number of design variables fixed, but the number of truss members is variable. Also, all variables being integer allows for pure-integer GA and RTS to be used in optimization, without the loss of practicality of the model.

2.3 CONSTRAINTS

Constraint evaluation is the main costly event in terms of CPU time. It involves generating a finite element (FE) mesh of the truss, solving the FE model for different load cases then performing safety check on truss members. The safety constraints involve:

- Load Cases include Dead load, Live Load and Wind Load. Load Case Combinations are Dead Load + Live Load (DL), Dead Load + Wind Load (DW) and Dead Load + Live Load + Wind Load (DLW)
- Mild steel members subjected to tension must not exceed allowed under any of the load case combinations.
- Members subjected to compression must not exceed allowed compressive stress under any of the load case combinations. Allowed compressive stress depends on member slenderness.
- Bending stresses in Purlins must be safe under all load cases and also capable of carrying a specified concentrated load in its mid-span.
- Depth of the beam cross-section chosen for Purlins should not be less than a certain portion of its length.
- Deflection under live load is not to exceed a certain amount.
- Slenderness of all members subjected to compression is not to exceed a certain value.
- Slenderness of any member is not to exceed a certain value.
- Purlin spacing should be within a certain range.
- Diagonal members angle from horizontal should be within a certain range.

Constraints are enforced through adaptive penalty (Chen 2001). To ensure that the search converges to a feasible design, additional cost is added to the objective function to make the cost of any infeasible design more than that of the current best feasible design. The penalty cost also depends upon the amount of violation. Typically, the penalty cost is high at the beginning of the search and is then gradually lowered as better feasible designs are found. A crucial matter for efficient employment of adaptive penalty is to have a feasible initial design.

2.4 OBJECTIVE FUNCTION

In many applied cases, truss optimization is a multi-objective process regarding issues such as weight, cost, stiffness and natural frequencies. However, the particular class of trusses considered finds its main domain of application in industrial and commercial clear-span buildings. For such applications, there is usually the single objective of minimizing the overall cost. In many practical cases, the overall cost is directly associated with the total steel weight. Thus for the current study, the objective is to minimize the overall weight. Such weight includes the main truss members, longitudinal bracing members, purlins and estimates of all connection plates (by empirical formulas in terms of other truss parameters).

The objective function (OF) combines the truss total weight plus a penalty term to prevent constraints violation. There are two cases for the objective function:

- **No constraints are violated**

In this case: $\mathrm{OF} = W_{\mathrm{t}}$

- **One or some of the constraints are violated**

In this case: $\mathrm{OF} = \max(W_{\mathrm{t}}, W_{\mathrm{b}}) \times C_{\mathrm{Pen}} \times I_{\mathrm{Pen}}$

Where:

W_t : is the total weight of the considered structure

W_b : is the total weight of the best feasible structure encountered so far during the optimization

C_{Pen} : is a penalty constant

I_{Pen} : is the number of truss members that violate the safety constraints

A key implemented feature is the adaptive penalty which aims at preventing "over-penalizing" the infeasible designs while making sure that no infeasible design has a better OF value than the best feasible encountered design.

3 GENETIC ALGORITHM

3.1 GENERAL PURPOSE GA

The general purpose genetic algorithm (GA_1) tested in this paper implements variable storage as integer variables, 4 crossover operators, 12 mutation operators, fitness scaling, population distribution, roulette wheel selection along with elitist selection.

Integer Storage: For efficiency of storage, variables are stored directly as integers rather than binary strings (Goldberg 1989) and are translated to their equivalent binary strings when need during crossover and mutation.

Crossover Operators:

- Binary string crossover.
- Inner Crossover (adopted from real coded GA). The new variable values are computed as:

$ChildVal_1 = Round\ (\alpha\ ParentVal_1 + (1 - \alpha)\ ParentVal_2)$

$ChildVal_2 = Round\ (\ (1 - \alpha)\ ParentVal_1 + \alpha\ ParentVal_2)$

Where α is a randomly generated number between 0 and 1

- Outer Crossover (adopted from real coded GA). The new variable value is computed as:

$ChildVal = Round\ (StrongerParentVal + \alpha\ (StrongerParentVal - Weaker\ ParentVal)\ )$

- Uniform crossover (Liang-Jie et al., 1995). In which the variables are unchanged, but exchanged between the parents with a 50% probability of exchange.

Mutation Operators:

- Binary bit flipping.
- Binary bit shift left.
- Binary bit shift right.
- Binary bit inversion.
- Shifting value to nearest boundary.
- New random number generation.

Another similar set of mutation operators is also used that only act if the member fitness is below average.

An overall probability for crossover and mutation is specified for a search. For each mutation or crossover operation of mating members, selection of which operator to use is performed randomly according to an assigned probability of use for each operator.

Fitness Scaling: linear fitness scaling is implemented to give a fair survival chance for strong population members.

Speciation: members further away from population average get a fitness bonus to encourage diversification.

Roulette Wheel Selection: is used for selecting members of old population for mating and producing new members of next population.

Elitist Selection: one copy of best member in a population passes unchanged to the next population to ensure that any optimized value is no worse than the best previously attained. And the rest of the new population is filled by the traditional selection, crossover and mutation.

Seeding: one feasible point is included in the initial population and rest of the population is chosen randomly. Due to the nature of the problem, a purely random initial population may end up with a population of all-infeasible designs. Such an initial population will cause failure of the adaptive penalty strategy, as it requires knowing the OF value of some feasible design.

3.2 GA WITH CACHING

The second implementation of GA tested in this paper (GA_2) is the same as GA_1, but all evaluations of objective function are stored. Thus, when performing population members OF evaluation, only un-explored regions of the search space will require the FE solution of the truss.

By nature, OF caching is inherent in RTS and is one of the strong points in favor of it. Therefore history storage is implemented into GA in order to even up the advantage RTS has and allow for a better comparison.

3.3 GA WITH NORMALLY DISTRIBUTED INITIAL POPULATION

RTS benefits from a good starting point, so an interesting study would be to have a biased initial population. Thus, the third implementation of GA (GA_3) is the same as GA_2, but has its all members of the initial population normally distributed about the initial feasible design.

4 REACTIVE TABOO SEARCH

4.1 GENERAL SCHEME

Reactive taboo search is a heuristic global optimization technique that has less stochastic content than genetic algorithm. In fact, save for a small portion of the algorithm, it is almost completely deterministic. The basic idea in taboo search (Glover 1986, 1989, 1990) is to make use of previously evaluated points within the search space

to direct the future sampling and prevent entrapment at a local minimum by applying taboo conditions. Reactive taboo search (Battiti and Tecchiolli 1994) proposes a scheme for adaptively varying the way the taboo conditions are applied based on the objective function history, thus the search "reacts" to the objective function behavior. Pseudo-code of RTS is given as:

```
1   Begin at a starting point
2   Examine Non-Tabooed Neighboring Points
    and move to the best of them
3   If new point has been not been visited before
4       Goto 2
5   Else If cycling is not "excessive"
6       Put a taboo condition upon point
7       Goto 2
8   Else perform "quick escape" and Goto 2
```

The single starting point in the search space is set as the "current point". RTS then evaluates the entire neighborhood of the current point and moves to the best point in it which then becomes the new current point. An important feature in RTS, is that all the previously evaluated points are stored in the memory, this leads to lots of savings in computational time when evaluating the neighborhood of the new point. Memorizing all evaluated points is costly in terms of required storage resources since the total memory required for the algorithm grows linearly as more points are being evaluated, however, such memorizing saves a lot of computational time if the OF is costly in terms of CPU evaluation time.

At the start of the search RTS, simply behaves like a steepest descent search until it hits a local minimum. Whereas steepest descent stops upon reaching a local minimum, RTS continues to search the neighborhood of the current point and move to best point within it even if it is worse than the current point. To prevent infinite cycling back and forth around a local minimum, TS imposes a taboo condition upon the last visited point, that is, "a previously visited point cannot be visited again until a certain number of iterations is completed", and such number of iterations is typically referred to as the "taboo list length".

In RTS, the taboo list length is adaptively changed according to the search behavior within a minimum and a maximum value. If the search still gets stuck in a large basin of attraction of the objective function, which the maximum taboo list length is not enough to overcome, a "quick escape" is performed.

The search is typically stopped after performing a specified number of moves or objective function evaluations. The best point encountered is returned.

4.2 NEIGHBORHOOD EVALUATION

RTS performs a complete neighborhood evaluation. Unlike the version of RTS proposed by Battiti and Tecchiolli (1994) where all variables were either zero or one, the implemented version in this paper uses integer values for the variables. The neighborhood is defined as the set of points that have all their variables equal to those of the current point except for one variable, which is different by a value of ±1. Thus, the number of points in the neighborhood is twice the number of variables (or less for points touching the upper and lower limits of the variable ranges).

4.3 RTS REACTION TO SEARCH BEHAVIOR

At each move (iteration), RTS places a taboo condition on the previous point, the taboo condition lasts a number of iterations equal to the current taboo list length. RTS also keeps track of when was each point visited, and the number of visits. If a point is visited twice, the taboo list length is increased. Thus, near a local minimum, the taboo list length keeps increasing until it is enough to explore regions further away. If a number of iterations pass without any cycles occurring (visiting the same point several times), the taboo list length is decreased.

Typically, a maximum taboo list length is specified. It is generally not beneficial to have the maximum taboo list length greater than the number of points in the neighborhood, because it can lead to a situation when all the points in the neighborhood are tabooed. When such a situation arises, the taboo conditions are relaxed, and the new current point is chosen as the last visited point in the neighborhood.

Sometimes if a large basin of attraction exists in the objective function, there could be a situation when taboo conditions are not enough to overcome the domain of the local minimum and that is when the "quick escape" is performed.

4.4 QUICK ESCAPE MECHANISM

RTS keeps a record of the average cycle length. When it approaches the maximum taboo list length, this indicates that tabooing is not enough to overcome the current basin of attraction, and quick escape is necessary. Quick escape is performed by randomly changing the values of some of the variables of the current point. It is simply just like re-starting the search at new starting point that is not entirely random.

5 APPLICATION

5.1 TRUSS DATA

Data of a real N-shaped truss is used as a starting point for the optimization algorithms. The truss data is given in Table 1.

Table 1 Truss Data

Number of Main Bays	2
Building Clear Span	21.0 m
Material Young's Modulus	207 GPa
Allowed Stress	140 MPa
Max. Slenderness (Compression Members)	180
Max. Slenderness (All Members)	300
Max. Deflection under live load	1/300 of Span
Live Load	50 kg/m^2
Wind Pressure	50 kg/m^2
Dead Load	Weight + 20 kg/m^2
Available Database Contains L-sections (LPN), C-sections (UPN & C.F.) and I-sections (IPN & IPE)	

A photo of the actual truss during erection procedure is given in Fig. 1. This design (topology, configuration and sizing) is used as the starting point for optimization. Topology and configuration are shown in Fig. 4. Truss member cross-sections are given in Table 2.

5.2 GA PARAMETERS

Among the several available tuning options for the implemented GA, the following settings are chosen:

- Population Size: 100, 150, 200 and 250
- Number of Generations: (Unlimited), search stops when maximum number of objective function evaluations is reached.
- Max. number of OF evaluations: (Tested Several)
- Overall crossover probability: 0.9
- Equal probability for different crossover operators
- Overall mutation probability: 0.25
- Equal probability for different operators
- Fitness scaling constant: 1.6

Choice of the search parameters was based on practical published values and the available computational resources. Further tuning is possible.

5.3 RTS PARAMETERS

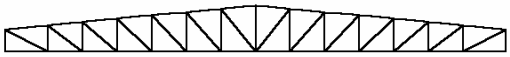
RTS has less tuning parameters than GA. The following settings are chosen:

- Number of moves: (Unlimited), search stops when maximum number of objective function evaluations is reached.
- Max. number of OF evaluations: (Tested Several)

5.4 RESULTS

Each of the design variables concerned with truss member sizing has 48 possible choice options, variables concerning configuration and topology range between 3 to 20 options. The total search space (all possible combinations of variables) is 1.58814×10^{37}. Practicality limits for reasonable CPU time made it preferable to limit the comparison of optimization algorithms to 10,000 OF evaluations. Some reasonably good results are obtained even though 10,000 OF evaluations comprise only 6.3×10^{-34} of the total search space.

Topology and configuration of the initial design, an intermediate design during optimization and final best obtained design are shown in Fig. 4. A listing of the chosen cross-sections for truss member groups and overall design weight is given in Table 2. The intermediate design is shown as a demonstration of topology change as well as sizing.



a) Initial Design, Weight = 9903.2 kg

b) Intermediate Design, Weight = 6328.8 kg

c) Best Obtained Design, Weight = 5655.3 kg

Figure 4: Truss Topology and Configuration

Table 2 Chosen Truss Member Groups Cross-sections

Variable	**Designs**		
	Initial	**Intermediate**	**Final Best**
X12	C.F. C140x4	C.F. C140x3	C.F. C140x3
X13	2xLPN 70x7	2xLPN 70x7	2xUPN 65
X14	2xLPN 70x7	2xLPN 70x7	2xIPN 80
X15	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X16	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X17	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X18	2xLPN 50x5	2xLPN 50x5	2xIPN 80
X19	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X20	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X21	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X22	2xLPN 70x7	2xLPN 70x7	2xLPN 50x5
X23	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X24	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X25	2xLPN 60x6	2xLPN 60x6	2xLPN 50x5
X26	2xLPN 50x5	2xLPN 50x5	2xLPN 30x3
X27	2xLPN 50x5	2xLPN 50x5	2xLPN 30x3
Truss Weight	**9903.2 kg**	**6328.8 kg**	**5655.3 kg**

Table 3 Optimization Results

# of OF Eval.	Objective Function Value							Standard Deviation		
	RTS	Avg. of 20 Runs			Best of 20 Runs					
		GA_1	GA_2	GA_3	GA_1	GA_2	GA_3	GA_1	GA_2	GA_3
500	7781	9518	9487	9034	8197	7703	7534	587	574	474
1000	6687	9346	9209	8825	7599	7656	7534	666	725	443
1500	6491	9216	8973	8775	7599	7656	7534	673	687	432
2000	6430	9088	8929	8759	7599	7656	7534	697	691	432
2500	6430	8987	8929	8706	7599	7656	7534	690	691	452
3000	6430	8866	8840	8658	7599	7656	7259	619	673	545
4000	6430	8754	8616	8538	7270	7236	7259	665	734	555
5000	6430	8664	8591	8463	7270	7236	7259	594	731	569
6000	6430	8513	8293	8427	7270	7236	7259	604	640	602
7000	6430	8436	8226	8358	7270	7194	7259	571	633	594
8000	5704	8396	8115	8255	7174	7034	7259	581	639	554
9000	5655	8263	7998	8150	7174	7034	7259	496	534	523
10000	5655	8213	7935	7969	7174	7034	7259	479	547	479

Since RTS has very little stochastic content compared with GA, only one optimization run is used as a representative of RTS. Twenty runs are performed for each of GA_1, GA_2 and GA_3 using four different population sizes (five runs for each population size). The results of optimization performance are summarized in Table 3 and plotted in Figures 5 – 6.

The results in Table 2 and Fig. 5 are for the number of *new* objective function evaluations, thus caching in GA_2 and GA_3 resulted in improvement of the performance over the traditional GA_1. Furthermore having the initial population normally distributed about the starting point in GA_3 improves the consistency of the search (as seen in the standard deviation of the 20 runs) and results in a quicker descent of the objective function at the start of search. GA_3 however has little or no advantage over GA_2 towards the end of the search.

Further examination of Figs 5 – 6 and Table 2 shows an appreciably better performance of RTS over GA. To analyze possible reasons for RTS being better suited for the examined optimization problem than the implemented forms of GA.

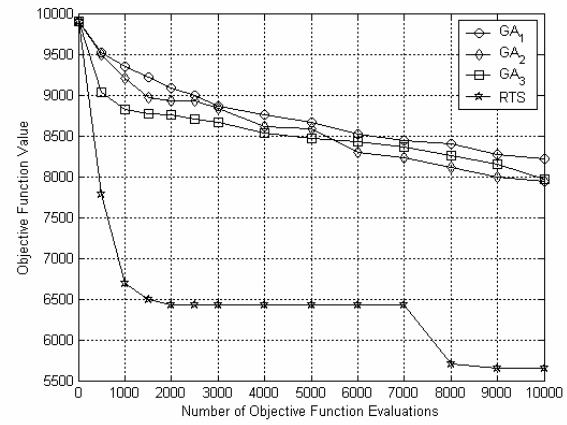


Figure 5 Optimization Progress – Average of GA Runs

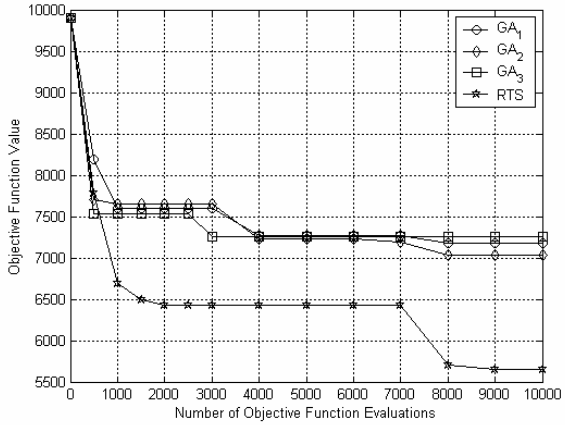


Figure 6: Optimization Progress – Best of GA Runs

6 DISCUSSION

GA relies on having several points that are distributed over the search space (population) to achieve diversification. According to the schemata theory (Goldberg 1989), selection along with crossover provides intensification by attracting the population points to zones of higher fitness. Eventually the whole population gets attracted to the global optimum. In general, the intensification properties of GA are not as good as those of local optimizers (Erbatur and Hasancebi 2001). Mutation is generally used to increase diversification, especially when the whole population gets too closely attracted to a certain region.

The main weakness GA suffers when the problem has large dimensionality is that a moderate population size (100 to 200 members) becomes insufficient to achieve enough diversification over the search space and insufficient schemata pool, which also confounds the intensification. Increasing the population size beyond certain limits is on the other hand very costly in terms of the number of objective function evaluations.

Another problem that GA encounters is due to the complexity of the constraints which makes GA unable to converge without seeding with an initial feasible point. Seeding itself decreases the GA efficiency.

RTS has separate mechanisms for intensification and diversification. For intensification, RTS relies on a local optimizer that nails down the local optimum. Thus, finding the local optimum is fast, efficient and has less sensitivity to large dimensionality than GA. This accounts for the fast descent of the OF value encountered at the beginning of the RTS search in Figs. 5 – 6. Upon reaching a local optimum, RTS switches to diversification by imposing taboo conditions to prevent moving to already explored points. If the taboo conditions are not enough to escape a large basin of attraction, RTS performs its quick escape move and "hopes" it will be enough to escape the current basin of attraction. It can be seen in Figs. 5 – 6 as well as Table 3 that after the good start, RST remained incapable of finding any better designs for a long period.

Given N number of objective function evaluations, the memory requirement is constant for the traditional GA (GA_1), but of order N for RTS, GA_2 and GA_3 because of caching. Caching also incurrs additional computational effort of order less than N^2 but such computational effort has little overall effect when the OF is costly to evaluate.

It is seen in this study that RTS has better capabilities for intensification as well as exploiting a good starting point while GA has better diversification. Future research aspects may include combining both to get even better. One such possibility would be to use RTS, but perform large OF attraction basin detection, once the quick escape mechanism becomes inefficient, the search may be switched to a population-based search until a new basin of attraction is found, then switch back to RTS.

7 CONCLUSIONS

Design optimization of a real-world class of plane trusses is considered. A parametric model of the truss is developed, which takes into account most of the practical aspects for design applicability. Optimization of the model is pretty challenging since it involves sizing, configuration and topology, large dimensionality and costly objective function. Three implementations of general purpose GA as well as RTS are tested to see if they can come up with better designs than an actual erected design. Through a number of objective function evaluations that is only a very small fraction of the total search space, both GA and RTS succeeded in coming up with better designs. Although RTS performed better, observation reveals that RTS has better intensification, while GA has better diversification. This motivates future work for combining aspects of GA and RTS.

Acknowledgments

This work is an extension of a course project of ME558 Discrete Design Optimization, offered in Fall 2001 at the University of Michigan, Ann Arbor. MECO, Modern Egyptian Contracting provided the data of the previously erected truss, used as starting point in this paper.

References

R. Battiti and G. Tecchiolli (1994), "The Reactive Tabu Search," ORSA Journal on Computing, V 6, pp. 126-140.

M.P. Bendose and N. Kikuchi (1988), "Generating Optimal Topologies in Structural Design using a Homogenization Method," Computer Methods in Applied Mechanics and Engineering, V 71, pp. 197-224.

C. Chapman, K. Saitou and M. Jakiela (1993), "Genetic Algorithms as an Approach to Configuration and Topology Design," Advances in Design Automation, V 65, pp. 485-498.

S. Y. Chen (2001), "An approach for impact structure optimization using the robust genetic algorithm," Finite Elements in Analysis and Design, V 37, pp. 431-446.

K. Deb and S. Gulati (2001), "Design of truss-structures for minimum weight using genetic algorithms," Finite Elements in Analysis and Design, V 37, pp. 447-465.

F. Erbatur and O. Hasancebi (2001), "Layout optimization using GAs and SA," Optimal Structural Design Workshop, GECCO-2001, pp. 102-107.

L. Gil and A. Andreu (2001), "Shape and cross-section optimization of a truss structure," Computers and Structures, V 79, pp. 681-689.

F. Glover (1986), "Future Paths for Intege Programming and Links to Artificial Intelligence," Computers and Operations Research, V 13, No. 5, pp. 533-549.

F. Glover (1989), "Tabu Search – Part I," ORSA Journal on Computing, V 1, pp. 190-206.

F. Glover (1990), "Tabu Search – Part II," ORSA Journal on Computing, V 1, pp. 4-32.

D. Goldberg and M. Samtani (1986), "Engineering Optimization via Genetic Algorithms," Proceeding of the 9th Conf. on Electronic Computations, ASCE, Birmingham, pp. 471-482.

D. Goldberg (1989), "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley.

M. Jakiela, C. Chapman, J. Duda, A. Adewuya, and K. Saitou (2000), "Continuum structural topology design with genetic algorithms," Computer Methods in Applied Mechanics and Engineering, V 186, No. 2, p 339—356.

U. Kirsch (1979), "Optimal Design of Trusses by Approximate Compatibility," Computers and Structures, V 12, pp. 93-98.

Z. Liang-Jie, M. Zhi-Hong and L. Yan-Da (1995), "Mathematical analysis of crossover operator in genetic algorithms and its improved strategy," Proceedings of the IEEE Conference on Evolutionary Computation, V 1, pp. 412-417.

J. Moh and D. Chiang (2000), "Improved Simulated Annealing Search for Structural optimization," AIAA Journal, V 38, pp. 1965-1973.

S. Rajeev and C.S. Krishnamoorthy (1992), "Discrete Optimization of Structures using Genetic Algorithms," Journal of Structural Engineering, V 118, No. 5, pp. 1233-1250.

J.E. Taylor and M.P. Rossow (1976), "An Optimal Structural Design using Optimality Criteria," Advances in Engineering Science, 13th Annaul Meeting, Hampton, VA, pp. 521-530.

Application of Genetic Programming to Motorway Traffic Modelling

Daniel Howard and Simon C. Roberts
Software Evolution Centre
Building U50, QinetiQ
Malvern, Worcs. WR14 3PS
United Kingdom
dhoward@qinetiq.com
Phone: +44-1684-894480

Abstract

This paper describes two innovative projects that the investigators are undertaking for the Highways Agency network operator. Both projects use genetic programming to develop traffic management systems. The first project addresses the reliable prediction of motorway journey times for high-flow low-speed conditions, i.e. congested periods. The second project concerns the detection of motorway incidents in low-flow high-speed conditions, i.e. late at night. Genetic programming manipulates traffic readings from the Motorway Incident Detection and Automatic Signaling system to arrive at solutions to both real world problems.

1 Introduction

The UK Highways Agency (HA) has invested heavily in the Motorway Incident Detection and Automatic Signaling (MIDAS) system primarily for the purpose of queue protection. Its secondary function is to provide, at zero cost, one-minute averaged traffic data from sites spaced at 500m intervals. On the M25 motorway which encircles London, MIDAS produces readings of traffic velocity, flow, occupancy, headway and flow categorized by vehicle length. These quantities are measured directly by loop sensors which are incorporated into the road surface in each motorway lane.

Ideally, a traffic system would be controlled by obtaining a comprehensive understanding of its dynamics. For example, by modelling driver behaviour characteristics in different traffic states, and by using number plate recognition to monitor lane changing. However, a detailed understanding requires an extensive development cycle which is incompatible with the HA's demand for short delivery times. Rather than seek a thorough understanding of the traffic system, computational techniques are being investigated to develop control functions that learn and operate on the available MIDAS data, e.g. artificial neural networks and genetic programming (GP).

2 Journey Time Prediction

The HA needs to produce highly accurate short-term journey time predictions to: allow tactical control systems such as ramp metering to be pro-actively deployed; give motorists advance warning of when and where a tactical control system will be operational on the network; improve the performance of real-time route assignment models; and signal predicted journey times to motorists.

This GP investigation aimed to discover mathematical and logical relationships between predicted journey times and recent MIDAS measurements. For this purpose the journey time, JT, was defined as the average time taken by a driver already on the motorway to cover the distance between two junctions, e.g. junction A merging and junction B diverging.

2.1 Forecasting options

Two alternative strategies for journey time prediction were investigated:

- the direct prediction of journey time.
- the prediction of velocity at each site along the journey. The journey time was then computed by integrating these velocity predictions along the path of the virtual journey into the future.

More GP evolution work was involved in the latter strategy because velocity predictions were carried out

at each site of the virtual journey. In contrast, the former strategy makes one forecast for the entire journey.

The latter strategy required a forecasting period to be chosen which was commensurate with the maximal expected JT. This allowed the predicted velocity at the terminating site to always be available and thus the time of the complete journey to be calculated. However, predictions were less accurate when the forecasting period was increased. Hence, a balance was achieved by setting the period to 15 minutes and by using a *naive* prediction (defined in Section 2.4) when JT exceeded this period.

Both strategies processed present and past traffic quantities for current and *downstream* sites. It was crucial to use downstream information to anticipate the future traffic state at the current site. This was especially important during high-occupancy periods because traffic queues building downstream tended to produce congestion waves which propagated upstream.

2.2 Transforming MIDAS data into a GP terminal set

MIDAS records minute-averaged traffic quantities ϕ_t^{sl} for site s on lane l at minute t:

$$\phi_t^{sl} = (V_t^{sl}, O_t^{sl}, F_t^{sl})$$

where V_t^{sl} is velocity in km/h, O_t^{sl} is the percentage lane occupancy and F_t^{sl} is the flow rate in vehicles per minute. A correction algorithm was devised to interpolate missing MIDAS data. This was invoked when measurements were simply absent and when no vehicles happened to pass a MIDAS sensor for at least one minute. In the latter case, the values $F_t^{sl} = 0$ and $O_t^{sl} = 0$ were valid, but V_t^{sl} must be estimated to avoid discontinuities with the neighbouring measurements (in time and location).

Lanes are numbered from the offside fast lane[1]. The prediction inputs combined quantities across the offside and the two adjacent lanes as follows. The velocities at each site were averaged across the lanes and weighted by the flow in each lane:

$$V_t^s = \frac{F_t^{s1}V_t^{s1} + F_t^{s2}V_t^{s2} + F_t^{s3}V_t^{s3}}{F_t^{s1} + F_t^{s2} + F_t^{s3}}$$

The input flow was the total across the lanes and the input occupancy was averaged:

$$F_t^s = \sum_{l=1}^{3} F_t^{sl} \qquad O_t^s = \frac{1}{3}\sum_{l=1}^{3} O_t^{sl}$$

[1]Sites between diverging and merging junctions have three lanes while all other sites have four lanes. The fourth lane is used to enter and exit the motorway.

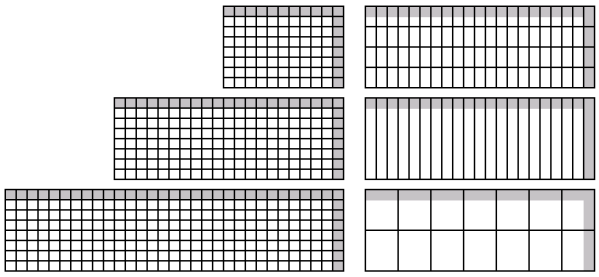


Figure 1: Examples of raw and box-averaged input windows. Time is shown horizontally and site is shown vertically. The grey cells are at the current time or the current site. Each square represents either a raw value or a box-averaged value.

The lane-independent traffic quantities were transformed into a GP terminal set by defining a window comprising previous minutes and downstream sites. At minute t and site s, the window spanned back to minute $t-T$ and down to site $s+S$. Journey time predictors simultaneously processed all sites in the journey, i.e. $S = 27$, and typically used $T = 15$. Velocity predictors typically used $S = 7$ and a wide variety of T values were investigated.

Figure 1 illustrates the different types of input windows. Clearly, when T and S were increased the size of the GP terminal set could become excessive and result in an over-complicated search space. Thus data reduction techniques were investigated. The most effective was found by dividing the input window into boxes of size $b_t \times b_s$, and averaging the raw values in each box to give a single value per box for each traffic quantity.

The most recent data was found to possess a stronger predictive power, as expected, but time windows which extended tens of minutes into the past were also found to be beneficial. To strike a balance between these extremes, overlapping boxes which extended further into the past were investigated as shown in Figure 2. Each box reached back from minute t to $t - b_t$ where b_t was initialized to 1 and then increased for each subsequent box. This produced superimposed multiscale time windows which gave a crisp short-term view but evermore blurred longer-term views. This box-averaging technique produced among the best predictors.

Data reduction was also investigated by fusing quantities based on flow-velocity plot descriptors, namely the flow-velocity gradient G or the flow-velocity magnitude M.

Some experiments also investigated feeding previous velocity predictions, $\hat{V}$, back in as input data. Fig-

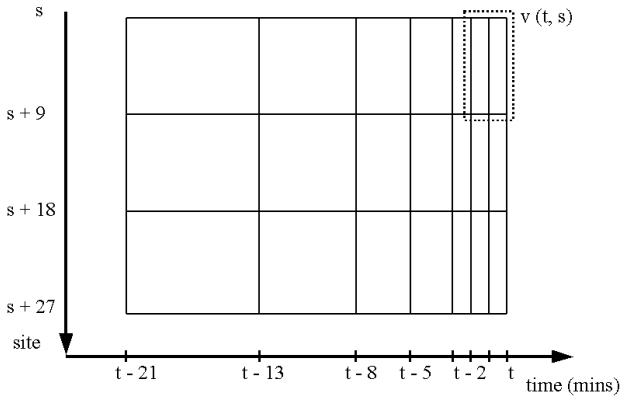


Figure 2: Multiscale input windows superimposed in the time dimension. The window size b_t increased based on the Fibonacci sequence.

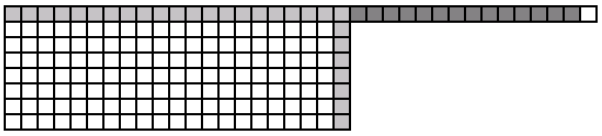


Figure 3: Velocity prediction $\hat{V}^s_{t+15}$ which uses the 14 former predictions of velocity, shown as dark grey cells.

ure 3 illustrates this special case where $(\hat{V}^s_{t+1}...\hat{V}^s_{t+14})$ was used in a form of autoregressive prediction. Journey time predictors could also exploit the recent known JT values, e.g. the JT of the most recent journey completed by time t. The experiments investigated which data types gave the most predictive power from:

$$\phi = \left(V, O, F, M = (F^2 + V^2), G = \frac{60F}{V}, \hat{V} \right)$$

Two years of historical MIDAS data was inspected on the counter-clockwise carriageway of the M25 between junctions 15 and 11. This identified days when the sensors were fully operational and the traffic involved congestion waves. Figure 4 illustrates the MIDAS data showing different traffic quantities on different lanes. The sub-image for each given quantity and lane shows progressive time as left to right and progressive sites as top to bottom. Brighter cells represent lower quantity values, e.g. congestion appears as low speed and high occupancy. Journey times between junctions 15 and 11 take between 6 minutes to over 25 minutes. For the experiments reported in this study, GP processed the period 1300h to 2200h.

2.3 Experimental GP study

Many GP runs explored different evolution options: a velocity predictor; a velocity-gradient predictor; a journey time predictor. The first two options were driven by velocity errors or JT errors via the fitness function. The third option could only be driven by JT error.

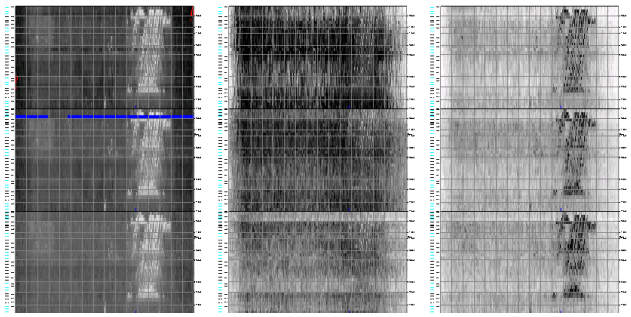


Figure 4: MIDAS data: 0600h to 2200h on 2nd September 1999, M25 counter-clockwise carriageway, junctions 15 to 11. From left to right: velocity, flow and occupancy. From top to bottom: offside (fast) lane, offside-1 lane and offside-2 lane.

Each option involved at least 15 independent GP runs continued for 50 generations. They used the steady-state GP method with a typical population size of 2000 individuals, 90% cross-over, 10% mutation, a breed tournament size of 4, a kill tournament size of 2 and the function set: $+$, $-$, $*$, protected division, min, max and *if-less-then-else*.

Both root mean square error, $\|\cdot\|$, and max error, $\max(\cdot)$, were applied to measure absolute error, e_a, and relative error, e_r. However, the experiments investigated various fusions of $\|\cdot\|$ and $\max(\cdot)$ errors of ϕ, the quantity whose error was to be minimized[2]. The errors were calculated as follows, where $\hat{\phi}$ was the evolved prediction and e_n was the error relative to a naive prediction, ϕ_N:

$$e_a = |\hat{\phi} - \phi| \qquad e_r = \frac{e_a}{\phi}$$

$$e_n = |\hat{\phi} - \phi_N| \qquad \text{e.g. } \|e_a\| = \sqrt{\frac{1}{n}\sum e_a^2}$$

Various fitness measures, f, were investigated:

$$f = -\left(\|e_a\|\|e_r\| + \max(e_a)\max(e_r) \right) ;$$

$$f = -\left(\|e_a e_r\| + \max(e_a e_r) \right) ;$$

$$\text{or by defining } e_c \text{ as: if } \left(\frac{e_a}{e_n}\right) > 1$$

$$\text{then } e_c = \left(\frac{e_a}{e_n}\right)^2 \text{ else } e_c = \left(\frac{e_a}{e_n}\right)$$

[2] V^s_t or ΔV^s_t or J_t.

$$f = -\|e_c\| \; ; \; f = -\left(\|e_c\| + \frac{\max(e_c)}{100}\right)$$

Predictors were compared based on JT prediction errors, regardless of whether the evolution was driven by JT or by velocity errors. Case-wise JT prediction errors were obtained for the 10 fittest predictors from each GP run. The following four error calculations were then used to quantify the overall JT errors: $\max(e_a)$, $\max(e_r)$, $\|e_a\|$ and $\|e_r\|$. This posed a multi-objective optimization problem in four dimensions and thus a *Pareto* ranking scheme was used to compare the different predictors. Pareto ranking was first performed on the $\max(e_r)$ and $\|e_r\|$ dimensions. A second independent Pareto ranking process was then performed on the $\max(e_a)$ and $\|e_a\|$ dimensions. Predictors were then ordered according to e_r rank and then e_a rank. The best predictor for the experiment was then subjectively designated from the top ranking predictors, and the associated JT errors were tabulated.

2.4 Naive predictions

Computational solutions find it challenging to perform markedly better than naive predictions. For example, it is difficult to beat the rule "the journey time in the future will be the same as the time of the journey just completed". A method to overcome this problem is to target the predictors at specific traffic states, and this method is currently under investigation.

Thus, five naive formulae were used to judge the performance of the GP evolved predictors. The simplest naive formula was $\hat{\phi}_{t+p} = \phi_t$ where p is the prediction period. This formula produced the best naive predictions and hence the presented naive results pertain only to this formula.

ϕ_t was either velocity at the current site or JT of the most recent journey *completed* by time t. For example, let the virtual journey which commenced 9 minutes ago take JT_{t-9} minutes to complete, and that which commenced 8 minutes ago take JT_{t-8} minutes to complete. If $JT_{t-9} = 8.5$ and $JT_{t-8} = 8.1$ then ϕ_t was set to JT_{t-9}. However, if instead $JT_{t-8} = 7.9$ then ϕ_t was set to JT_{t-8}.

The prediction period, p, was 15 minutes for velocity predictors as discussed in Section 2.1. JT predictors were required to predict the time of the journey *starting* on the next minute and thus $p = 1$.

Table 1: JT errors for naive and evolved predictors of velocity and journey time. Absolute errors are in seconds.

Scheme	$\|e_a\|$	$\|e_r\|$	$\max(e_a)$	$\max(e_r)$
naive V_{t+15}	99.4	0.121	343.3	0.499
evolved V_{t+15}	78.7	0.099	256.3	0.35
naive JT_{t+1}	110.6	0.122	352	0.463
evolved JT_{t+1}	65.5	0.074	250.7	0.25

2.5 Results summary

More than 100 sets of experiments were run under different options. Each run processed traffic from 4 weekdays in August 1998. Table 1 shows the JT errors for the best prediction schemes, where V_{t+15} is the prediction of velocity in 15 minutes time, and JT_{t+1} is the direct prediction of the time of the journey starting on the next minute. It can be seen that the evolved predictors consistently gave better accuracy than the naive predictors, for both the prediction of velocity and JT. Furthermore, the evolved JT predictor gave the lowest error for each error type, despite the naive JT predictor being marginally worse than the naive velocity predictor.

The following general recommendations were drawn from the overall comparative study.

Predictor type: Predicting JT_{t+1} gave lower JT errors than predicting V_{t+15}. JT predictors were also much faster to evolve and apply than velocity predictors because velocity must be predicted at each site in the journey. However, JT predictors may not generalize across journeys traversing a different number of sites.

Data type: All traffic quantities (V, O and F) were required whereas the fused quantities (M and G) gave greater errors. Feeding intermediate predictions, $\hat{V}$, back into the input tended to reduce prediction accuracy, probably because this "autoregression" complicated the search space by increasing the size of the terminal set. Box averaging proved to be the most beneficial data reduction method.

Error drivers: Recall that various prediction errors drove the evolution via the fitness function. JT errors were better drivers than velocity errors. Both absolute and relative errors were needed and $\|\cdot\|$ errors had to be weighted stronger than $\max(\cdot)$ errors.

Table 2: JT errors for naive and evolved JT_{t+1} predictors on training and test data. Absolute errors are in seconds.

Training: 4853 cases over 15 weekdays				
b_s	$\|e_a\|$	$\|e_r\|$	$\max(e_a)$	$\max(e_r)$
(naive)	68.4	0.086	358	0.503
9	45.5	0.057	213.2	0.229
3	51.3	0.066	228.2	0.271
Testing: 2008 cases over 5 weekdays				
b_s	$\|e_a\|$	$\|e_r\|$	$\max(e_a)$	$\max(e_r)$
(naive)	65.1	0.089	269.4	0.546
9	45.9	0.06	217.9	0.269
3	52.3	0.072	229.3	0.398

The JT_{t+1} predictor was scaled-up by using more data from August 1998 and the results are shown in Table 2. Box averaged inputs were used with various b_s (Section 2.2) and $b_s = 9$ proved to be the optimum. The naive predictions are indicated by the b_s column in the table. It can be seen that the evolved predictor consistently gave better accuracy than the naive predictor. Furthermore, comparison of the results on training and testing shows that the evolved predictor generalized well.

3 Incident Detection

The HA's MIDAS system uses the HIOCC algorithm (Collins et al., 1979) to detect incidents. The HIOCC (high occupancy) algorithm is essentially a queue detection algorithm. It is very capable of detecting incidents during the peak and interpeak periods, but it gives poor results at night because the traffic flows are too low to allow queues to form. HIOCC or California (Payne et al., 1975) algorithms cannot detect late-night incidents because of the low-occupancy traffic states.

The Staged GP (Howard and Roberts, 1999) method had proved successful in the detection of objects of large variability in poorly specified domains. For this reason, Staged GP was selected to detect subtle anomalies in the late-night MIDAS data, in order to warn for the presence of poorly specified incidents. Success would be much cheaper than alternative approaches such as the capital cost required for upgrades to the traffic monitoring infrastructure.

The sought incident detector should detect the incident as early as possible to allow the human MIDAS operator to take action. For example, the operator could react by positioning a motorway camera to examine the scene of the incident, or by alerting the police who need a maximal response time. Late-night incidents have many causes, e.g. driver fatigue, roadworks, bad weather or animals on the motorway, resulting in a high variability in the characteristics of the incident onsets.

GP was used in a supervised learning role. Hence, it required a "truth" from which to learn to detect incident onsets. Unfortunately, until now very few incidents had been archived, and even where they had their reasons were not always correctly recorded. This investigation manually scrutinized the MIDAS data and marked up an incident "truth". The data was analyzed by inspecting the velocity in all lanes and identifying the approximate time and site of each incident onset. Incidents were then graded by a subjective measure of their significance. Grade 1 incidents were the most obvious type, corresponding to serious accidents which resulted in sustained lane closures. Grade 5 incidents were the most subtle type and appeared only as blips in the viewed MIDAS data. Major incidents were rarer than minor incidents and thus the significance grading was incorporated into the GP fitness measure, to counteract the bias towards detecting simply the most common types of incident (Howard and Roberts, 1999).

The detection task was tackled by a two-stage evolution strategy (Howard and Roberts, 1999). The first stage evolved for itself which traffic data best characterized an incident onset, by distinguishing data proximate to incidents from a sample of non-incident data. The second stage was then required to minimize the false alarm rate whilst retaining at least a single detection point per incident.

3.1 Traffic data input to GP

Currently we are evolving an incident detector with a training, validation and test set that we have painstakingly derived from three years worth of MIDAS data. However, in this paper we report on a smaller earlier study. Traffic data was taken from 60 consecutive sites along the M25 between sites 4727 and 5010. Both carriageways were used for 7 nights in August 1998, which comprised 32 incidents of grade 1 to 4. Grade 5 incidents were ignored.

3.2 Fitness measure

A GP chromosome was said to have detected an incident when it output a positive value. In other words, the traffic data being processed at the given time and site was deemed to represent an incident. If an actual

incident had occurred then the output was called a *true positive*, otherwise it was called a *false positive* or a false alarm. After a chromosome had processed all the training cases, let TP denote the number of resulting true positives and FP denote the number of false positives.

The task was to distinguish incidents of grade 1 to 4 from non-incidents. Furthermore, lower numbered incident grades were more important than higher numbered grades. These issues were captured by the following fitness measure which drove the evolution runs,

$$\text{fitness} = \frac{TPS}{(TPS_{max} + \beta FP)} \quad (1)$$

where TPS was the score for the incidents detected and TPS_{max} was the score obtained when all incidents were detected. TPS was calculated as follows to weight the incidents according to grade.

$$TPS = \sum_{i=0}^{i<TP} 5 - \text{grade of incident}_i \quad (2)$$

The variable β was used to balance the importance of incident detection against the expense of detecting false alarms. For example, as β tended to zero the number of false alarms became irrelevant.

The *figure of merit* (FOM) was used to compare different GP runs. The FOM is the same as the fitness when $\beta = 1$. Note that the minimum FOM is 0 for the case when no incidents are detected or when the number of false alarms approaches infinity, and the maximum FOM is 1 when all incidents are detected with no false alarms.

3.3 Evolving first stage detectors

Even though incidents happen at a single time and location, the incidents generally manifest themselves in the traffic data over multiple minutes and a number of sites (more upstream sites than downstream). Consequently GP was trained to detect the onset of each incident by sweeping from 5 minutes before to 5 minutes after the incident time, and by sweeping from 4 upstream sites to 4 downstream sites for each minute. However, an incident was said to be detected if a GP chromosome returned a positive output for any one of these times and locations. Therefore, TP represented the number of incidents detected and had a maximum value of 32. Non-incident training data was sampled at 10 minute steps using all possible sites at each minute, giving a total of 21,181 non-incident training cases.

The GP terminal set was taken from an input window similar to those illustrated in Figure 1. However, the window encompassed only local traffic data by using $T \leq 8$ previous minutes and one downstream site, i.e. $S = 1$. Furthermore, the input data consisted of the *lane-specific* MIDAS quantities: velocity, flow, occupancy, headway and flow categorized by vehicle length.

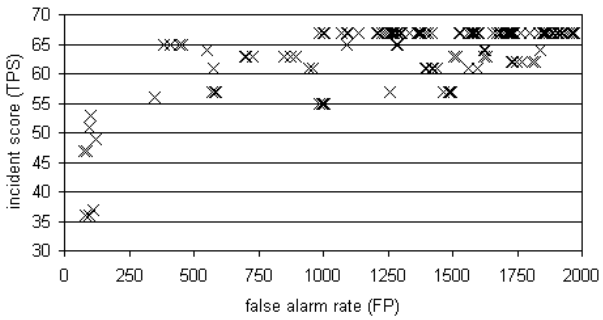


Figure 5: Incident score against false alarm rate for the first stage detectors.

Approximately 60 GP runs were conducted for the first stage each using $T = 3$. A first stage detector's task was to detect *all* incident onsets whilst producing a minimal false alarm rate. The fitness variable β was thus set to low values and the range 0.01 to 0.1 tended to give the best results. Higher settings caused incidents to be missed whilst lower settings tended to give an excessive false alarm rate. The population size was set to 1000 and the other GP parameters were the same as those listed in Section 2.3.

3.4 Validating first stage detectors

The evolved first stage detectors were validated on the 32 incidents and a maximum score of 67 was achieved if all incidents were detected (i.e. $TPS_{max} = 67$). The detectors processed all non-incident traffic data in validation which totalled to 200,010 cases. Figure 5 plots the incident score against false alarm rate for the 10 fittest detectors from each GP run.

The lowest false alarm rate when all incidents were detected was 987 out of the 200,010 non-incident cases. However, another first stage detector was judged to be the best because it produced 142 hits distributed across 30 incidents and it gave only 382 false alarms, and sacrificed only two grade 4 incidents in order to achieve this (i.e. $TP = 30$ and $TPS = 65$). Even by careful manual inspection it was difficult to determine whether these grade 4 incidents were actually true.

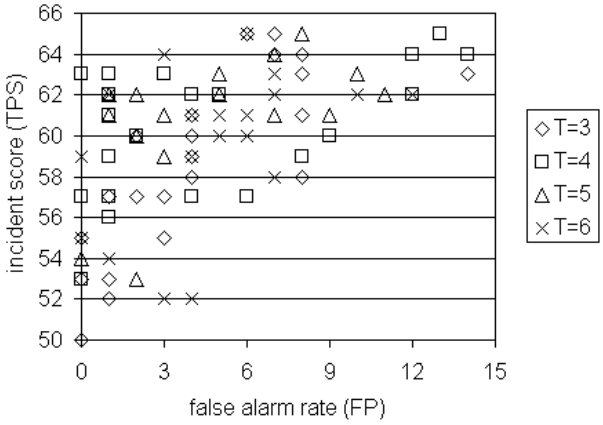


Figure 6: Incident score against false alarm rate for the second stage detectors.

3.5 Evolving second stage detectors

Second stage detectors were trained to reduce the false alarms whilst retaining at least a single hit per incident. Note that TPS_{max} reduced to 65 because of the two grade 4 incidents missed by the first stage detector.

The parameters β and T were optimized by conducting at least 50 GP runs for various settings. The detection performance was largely insensitive to T between values of 2 and 8. The results showed that β must exceed 0.1 in order to achieve no false alarms, and that a good balance between maximizing detected incidents and minimizing false alarms was achieved by setting β to 0.5.

More GP runs were conducted with $\beta = 0.5$ and a population size of 4000. Figure 6 plots the resulting incident scores against false alarm rate for the best detector from each GP run for various T values.

Recall that two grade 4 incidents were missed at the first stage and so a second stage detector could identify a maximum of 30 incidents with a TPS of 65. This was achieved at the expense of giving 6 false alarms, i.e. less than a single false alarm per night on average. These false alarms are arguably grade 5 incidents and two of them actually refer to the same event which was detected at adjacent minutes and sites. The maximum FOM was achieved when two further grade 4 incidents were missed to bring TPS down to 63, but this detector abolished all false alarms.

The incidents detected by the best second stage detector (i.e. the one which gave the maximum FOM) are shown in Table 3. The *lag* column gives the lag in minutes between the marked incident onset and the time of detection. This was negative when the incident started to manifest itself in the traffic data before the manually marked onset, e.g. because of a gradual onset spread across lanes. The column shows that most incidents were detected within two minutes. The last column gives the number of hits per incident and shows that grade 1 incidents received multiple hits but higher grades tended to be hit only once.

Table 3: Incidents detected by the best second stage detector. The lag is in minutes and the number of detections per incident is given.

date	time	site	grade	lag	no. of hits
3	2320	4989a	1	-1	5
4	0319	4927a	4	0	1
9	2253	4912a	3	0	4
10	0203	4935a	3	0	1
11	2200	4742a	1	0	3
11	2232	4762a	2	0	1
12	0540	4912a	3	1	1
28	0035	4932a	3	0	1
5	2334	4955b	3	2	1
5	2337	4940b	4	-1	1
6	0040	4757b	2	1	2
6	0336	4955b	4	5	1
6	0515	4955b	4	2	2
9	2243	4945b	4	0	3
10	0015	4935b	3	0	2
11	0346	5002b	4	0	1
11	0431	4949b	2	5	1
11	2258	4949b	2	1	1
11	2340	4949b	2	1	1
12	0050	4945b	3	0	1
12	0128	4752b	3	3	1
12	0227	4949b	3	-4	4
26	2346	4888b	2	-5	1
26	2352	4797b	2	5	1
27	0227	4949b	4	0	1
27	0311	5002b	3	0	1
27	2151	4832b	1	0	6
28	0436	4737b	2	3	1

3.6 Results summary

The task was to detect incidents on the M25 at periods of low traffic occupancy. This was approached by training GP to detect the onset of incidents which occurred during the night (approximately between 2200h and 0600h) whilst producing a near-zero false alarm rate.

The precise onset of incidents was often poorly defined and so the task was tackled by a two-stage evolution strategy. The first stage evolved for itself which traffic data best characterized an incident onset by distinguishing data proximate to incidents from a sample of non-incident data. The second stage was then required to minimize the false alarm rate whilst retaining at least a single detection per incident.

The best first stage detector missed two grade 4 incidents (the least obvious incidents) and gave 382 false alarms from 7 nights worth of data using both carriageways. The best second stage detector missed another two grade 4 incidents but abolished all the false alarms. Other second stage detectors retained all incident detections but gave 6 false alarms.

3.7 Current work

The activity of manually marking more than 1000 incidents over three years worth of traffic data has shed light into the nature of the incidents, which in turn has increased the subjective threshold as to what constitutes an incident. This has improved the quality of the "truth" in the scaled-up project, and it is hoped that this will be reflected in the detection performance. Tests are currently being undertaken to assess the generalization of the detection performance. Other modelling parameters such as the effect of weather, e.g. rain and fog measurements, may be incorporated as inputs to future detectors.

4 Conclusions

This paper describes two projects that apply genetic programming to real world problems proposed by the UK Highways Agency. The first project evolved motorway journey time predictors which were required to be reliable during high-flow low-speed conditions. These peak travel periods contained the most variable traffic characteristics and notably included congestion waves. The evolved predictors processed lane-independent traffic quantities and improved on the accuracy of equivalent naive predictors. Naive predictions are typically difficult to out-perform due to the inherent irregularities in the traffic data during congested periods.

The second project evolved motorway incident detectors for low-flow high-speed conditions, i.e. late at night. A Staged GP method was employed to identify the subtle anomalies which correspond to late-night incident onsets. The evolved detectors processed lane-specific traffic quantities and achieved near-zero false alarm rates whilst only missing very few minor incidents. Work is currently underway to scale this project up to extensive traffic data sets and to assess detection generalization.

It is hoped that an insight can be gained into the principles underlying the two projects by interpreting the structures of the evolved processors. Further understanding could be obtained by analyzing the behaviour of the evolved processors in different traffic states.

Acknowledgment

The authors wish to thank the UK Highways Agency for facilitating the data for this study which is highly specific to MIDAS and to the M25 motorway.

References

[Beale, 2002] Beale, S. (2002). Traffic Data: Less Is More. HA Internal Report, Highways Agency, Bristol, UK.

[Collins et al., 1979] Collins J. F., Hopkins C. M. and Martin J. A. (1979). Automatic incident detection - TRRL algorithms HIOCC and PATREG. TRL Report SR 526, Transport Research Laboratory, Cowthorne, UK.

[Howard and Roberts, 1999] Howard D. and Roberts S. C. (1999). A Staged Genetic Programming Strategy for Image Analysis. In Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela and Smith (eds), *Proceedings of the Genetic and Evolutionary Computation Conference, 1047–1052*, Morgan Kaufmann.

[Mahalel and Hakkert, 1985] Mahalel D. and Hakkert A. S. (1985). Time Series Model for Vehicle Speeds. Transportation Research Vol. 19B(3), 217–225.

[Moorthy and Radcliffe, 1988] Moorthy C. K. and Radcliffe B. G. (1988). Short term traffic forecasting using time series methods. Tansportation Planning & Technology, Vol. 12, 45–46.

[Payne et al., 1975] Payne H. J., Goodwin D. N. and Teener M. D. (1975). Evaluation of existing incident detection algorithms. Technology Service Corporation, Santa Monica, California.

Fitness Approximation in Evolutionary Computation - A Survey

Yaochu Jin and Bernhard Sendhoff
Future Technology Research
Honda R&D Europe (D) GmbH
Carl-Legien-Strasse 30
63073 Offenbach/Main, Germany
Email: yaochu_jin@de.hrdeu.com

Abstract

Fitness evaluation is a basic operation in evolutionary computation. However, an explicit fitness function is not always available in real-world applications. In many cases, it is necessary to estimate the fitness function by constructing an approximate model. In this paper, a short survey on fitness approximation in evolutionary computation is given. Main issues like approximation models, model management schemes, learning methods as well as data sampling techniques are presented. Finally, interesting open problems are discussed.

1 Introduction

Evolutionary algorithms are receiving increasing interest both in academy and industry. Among other applications, evolutionary algorithms have been widely used as global optimizers. Generally, evolutionary algorithms outperform conventional optimization algorithms for problems which are discontinuous, non-differential, multi-modal, noisy and not well-defined problems, such as art design, music composition and experimental designs [47]. Besides, evolutionary algorithms are also well suitable for multi-criteria problems.

With all the great successes achieved in real-world applications, evolutionary algorithms have also encountered some challenging difficulties. For most evolutionary algorithms, a large number of fitness evaluations (performance calculations) are needed before a well acceptable solution can be found. In many real-world applications, fitness evaluation is not trivial. There are several situations in which fitness evaluation becomes difficult and a computationally efficient approximate model (also known as meta-model) will be very helpful.

If an approximate model is needed for fitness evaluation, an important issue is then to select the most important one for fitness approximation. So far, there are several models that can be used for fitness approximation. The most popular ones are polynomials (often known as Response Surface Methodology), the kriging model, also known as design and analysis of computer experiments (DACE) and feedforward neural networks, including multi-layer perceptrons and radial-basis-function networks. Due to the lack of data and the high dimensionality of input space, it is very difficult to obtain a perfect global approximation of the original fitness function. To address this problem, two main measures can be taken. Firstly, the approximate model should be used together with the original fitness function. In most cases, the original fitness function is available, although it is computationally very expensive. Therefore, it is very important to use the original fitness function efficiently. This is called evolution control [24] or model management. Secondly, the quality of the approximate model should be improved as much as possible given a limited number of data. Several aspects are important to improve the model quality, such as selection of the model, use of active data sampling and weighting (both on-line and off-line), selection of training method and selection of error measures.

The work on fitness approximation in evolutionary computation has been distributed in several different areas. Therefore, this survey aims at providing readers a relatively comprehensive picture of fitness approximation in evolutionary computation. In Section 2, different motivations for using approximation in evolutionary computation are presented. The issue of model management is described in Section 3, where existing methods are divided into three main schemes. The main approximation models that have been used in fitness approximation are introduced in Section 4. Data

sampling techniques, which are very important for the quality of models, are given in Section 5. Finally, open questions and promising research topics are discussed in Section 6.

2 Motivations

The concept of approximation in optimization is not new [1]. Generally, there are two basic approaches, i.e., functional approximation and problem approximation. In functional approximation, an alternate and explicit expression is constructed for the objective function (in evolutionary computation, it is usually called fitness function). In contrast, problem approximation tries to replace the original statement of the problem by one which is approximately the same to the original problem but which is easier to solve. In this survey, the main focus is function approximation, nevertheless, the issue of problem approximation will also be discussed briefly in the last section.

So far, approximation of the fitness function in evolutionary computation has been applied mainly in the following cases.

- The computation of the fitness is extremely time-consuming. One good example is structural design optimization [19, 30, 29, 21, 34, 48, 38, 25]. In aerodynamic design optimization, it is often necessary to carry out computational fluid dynamics (CFD) simulations to evaluate the performance of a given structure. A CFD simulation is usually computationally expensive, especially if the simulation is 3-dimensional, which takes over ten hours on a high-performance computer for one calculation. Therefore, approximate models have widely been used in structure optimization [1].

  Fitness approximation has also been reported in protein structure prediction using evolutionary algorithms [35, 37]. A neural network has been used for feature extraction from amino acid sequence in evolutionary protein design [46].

- There is no explicit model for fitness computation. In many situations, such as in art design and music composition as well as in some areas of industrial design, the evaluation of the fitness depends on the human user. Generally, these problems can be addressed using interactive evolutionary computation [50]. However, a human user can easily get tired and an approximate model that embodies the opinions of the human evaluator is also very helpful [2, 27].

- The environment of the evolutionary algorithm is noisy. Usually, there are two methods to deal with noisy fitness functions. The first one is to sample the fitness several times and to average [16]. However, this method requires a large number of additional fitness evaluations. The second method is to calculate the fitness of an individual by averaging the value of this individual as well as that of other individuals in its neighborhood. To avoid additional computational cost, the individuals that participate in the averaging can be chosen from the current and previous generations [5]. A more flexible alternative is to estimate the fitness of the individuals in the neighborhood using a statistical model constructed with history data [45, 6].

- The fitness landscape is multi-modal. The basic assumption is that a global model can be constructed to approximate and smoothen out the local optima of the original multi-modal fitness function without changing the global optimum and its location [31]. Similar ideas have also been reported in conventional optimization methods [3, 4]. However, it is generally difficult to build an approximate model that has the same global optimum on the same location when the dimensionality is high with limited number of samples.

3 Approximate Model Management

The application of approximation models to evolutionary computation is not as straightforward as one may expect. There are two major concerns in using approximate models for fitness evaluation. First, it should be ensured that the evolutionary algorithm converges to the global optimum or a near-optimum of the original fitness function. Second, the computational cost should be reduced as much as possible. One essential point is that it is very difficult to construct an approximate model that is globally correct due to the high dimensionality, ill distribution and limited number of training samples. It is found that if an approximate model is used for fitness evaluation, it is very likely that the evolutionary algorithm will converge to a false optimum. A false optimum is an optimum of the approximate model, which is not one of the original fitness function, refer to Fig.1 for an example. Therefore, it is very essential in most cases that the approximate model should be used together with the original fitness function. This can be regarded as the issue of model management or evolution control. By evolution control, it is meant that in evolutionary computation using approximate models, the original fitness function is used to evaluate some of the individuals or all indi-

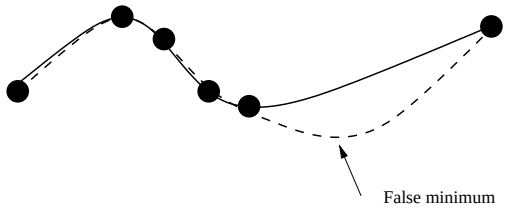


Figure 1: An example of a false minimum in the approximate model. Solid line denotes the original fitness function, dashes line the approximate model and the dots the available samples.

viduals in some generations [24]. An individual that is evaluated using the original fitness function is called a controlled individual. Similarly, a generation in which all its individual are evaluated using the original fitness function is called a controlled generation.

Generally, existing work on approximation in evolutionary computation can be divided into three main approaches from the viewpoint of evolution control.

3.1 No Evolution Control

Very often, the approximate model is assumed to be of high-fidelity and therefore, the original fitness function is not at all used in evolutionary computation, such as in [2, 43, 27].

3.2 Fixed Evolution Control

The importance to use both the approximate model and the original function for fitness evaluation has been recognized [41]. There are generally two approaches to evolution control, one is individual-based [19, 10], and the other is generation-based [41, 42]. By individual-based control, it is meant that in each generation, some of the individuals use the approximate model for fitness evaluation and others the original function for fitness evaluation. In individual-based evolution control, either a random strategy or a best strategy can be used to select the individuals to be controlled [24]. In the best strategy, the best individual (based on the ranking evaluated by the approximate model) in the current generation is reevaluated using the original function [19], see Fig.2. To reduce the computational cost further, individual-based evolution control can be carried out only in a selected number of generations [10]. In contrast, the random strategy selects certain number of individuals randomly for reevaluation using the original fitness function [24]. An alternative to the best strategy and the random strategy is to evaluate the mean of the individuals in the current population [34].

Generation-based evolution control can also be implemented [41, 42]. In [41], generation-based evolution control is carried out when the evolutionary algorithm converges on the approximate model. More heuristically, evolution control is carried out once in a fixed number of generations, see Fig. 3.

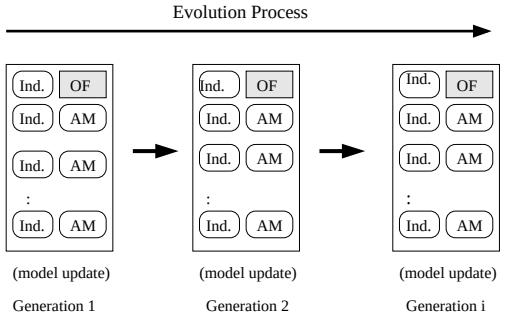


Figure 2: The best individual is controlled in each generation. AM: approximate model; OF: original function.

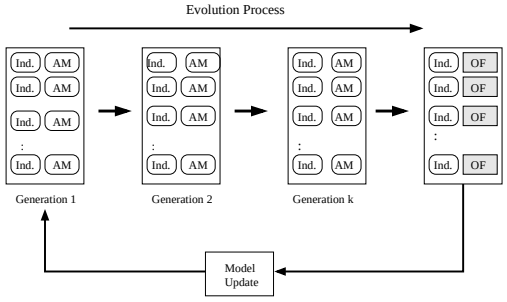


Figure 3: Generation-based evolution control. AM: approximate model; OF: original function.

One drawback in the aforementioned methods is that the frequency of evolution control is fixed. This is not very practical because the fidelity of the approximate model may vary significantly during optimization. In fact, a predefined evolution control frequency may cause strong oscillation during optimization due to large model errors, as observed in [41].

3.3 Adaptive Evolution Control

It is straightforward to imagine that the frequency of evolution control should depend on the fidelity of the approximate model. A method to adjust the frequency of evolution control based on the trust region framework [14] has been suggested in [34], in which the generation-based approach is used. A framework for approximate model management has also been suggested in [25], which has successfully been applied to 2-dimensional aerodynamic design optimization, see Fig. 4.

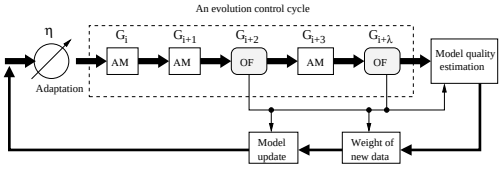


Figure 4: Adaptive generation-based evolution control. In the evolution control cycle, there are λ generations, η ($\eta \leq \lambda$) generations will be controlled. AM: approximate model; OF: original function.

4 Approximation Models

4.1 Polynomial Models

The most widely used polynomial approximation model is the second-order model which has the following form:

$$\hat{y} = \beta_0 + \sum_{1 \leq i \leq n} \beta_i x_i + \sum_{1 \leq i \leq j \leq n} \beta_{n-1+i+j} x_i x_j, \quad (1)$$

where β_0 and β_i are the coefficients to be estimated, and the number of terms in the quadratic model is $n_t = (n+1)(n+2)/2$ in total, where n is the number of input variables.

To estimate the unknown coefficients of the polynomial model, both least square method (LSM) and gradient method can be used:

- **Least Square Method** To get a unique estimation of the coefficients using LSM, it is required that the number of samples (N) drawn from the original function should be equal to or larger than the number of coefficients n_t. Let

$$\boldsymbol{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(N)}]^T, \quad (2)$$

and

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & (x_n^{(1)})^2 \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & (x_n^{(2)})^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & (x_n^{(N)})^2 \end{bmatrix}, \quad (3)$$

then the following equation holds:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\Theta}, \quad (4)$$

The LSM algorithm works as follows,

$$\hat{\boldsymbol{\Theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \quad (5)$$

where $\hat{\boldsymbol{\Theta}}$ denotes the estimate of $\boldsymbol{\Theta}$. One assumption here is that the rows of $\boldsymbol{X}$ are linearly independent.

- **Gradient Method** The main drawback of the least square method is that the computational expense becomes unacceptable as the dimensionality increases. To address this problem, the gradient method can be used. Define the following square error function for the k-th sample:

$$E^{(k)} = \frac{1}{2}(y - y^{(k)})^2, \quad (6)$$

where y is defined in equation (1), it is then straightforward to get the update rule for the unknown coefficients:

$$\Delta\beta_0 = -\xi \cdot (y - y^{(k)}) \quad (7)$$

$$\Delta\beta_i = -\xi \cdot (y - y^{(k)}) x_i^{(k)} \quad (8)$$

$$\Delta\beta_{n-1+i+j} = -\xi \cdot (y - y^{(k)}) x_i^{(k)} x_j^{(k)}, \quad 1 \leq i \leq j \leq n. \quad (9)$$

4.2 Kriging Models

The kriging model can be seen as a combination of a global model plus a localized "deviation":

$$y(\boldsymbol{x}) = g(\boldsymbol{x}) + Z(\boldsymbol{x}), \quad (10)$$

where $g(\boldsymbol{x})$ is a known function of $\boldsymbol{x}$ as a global model of the original function, and $Z(\boldsymbol{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. Usually, $g(\boldsymbol{x})$ is a polynomial and in many cases, it is reduced to a constant β.

The covariance of $Z(\boldsymbol{x})$ is expressed as

$$\mathrm{Cov}[Z(\boldsymbol{x}^{(j)}), Z(\boldsymbol{x}^{(k)})] = \sigma^2 \boldsymbol{R}[R(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(k)})] \quad (11)$$
$$j, k = 1, \cdots, N,$$

where R is the correlation function between any two of the N samples, and $\boldsymbol{R}$ is the symmetric correlation matrix of dimension $N \times N$ with values of unity along the diagonal. The form of the correlation matrix can be selected by the user, and the following form has often been used [9, 18, 49]:

$$R(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(k)}) = \exp[-\sum_{i=1}^{n} \theta_i |x_i^{(j)} - x_i^{(k)}|^2], \quad (12)$$

where θ_i are the unknown correlation parameters, $x_i^{(j)}$ and $x_i^{(k)}$ are the i-th component of sample points $\boldsymbol{x}^{(j)}$ and $\boldsymbol{x}^{(k)}$. Thus, the prediction of $y(\boldsymbol{x})$ is a function of unknown parameters β and $\theta_i, i = 1, 2, \cdots, n$:

$$\hat{y} = \hat{\beta} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{y} - \beta\boldsymbol{I}), \quad (13)$$

where, $\hat{y}$ is the estimated value of y given the N samples and the current input $\boldsymbol{x}$, $\hat{\beta}$ is the estimated value of β, $\boldsymbol{y}$ is a vector of length N as defined in Eqn. (2), $\boldsymbol{I}$ is a unit vector of length N, and $\boldsymbol{r}$ is the correlation vector of length N between the given input $\boldsymbol{x}$ and the samples $\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}\}$:

$$\boldsymbol{r}^T(\boldsymbol{x}) = [R(\boldsymbol{x}, \boldsymbol{x}^{(1)}), R(\boldsymbol{x}, \boldsymbol{x}^{(2)}), \cdots, R(\boldsymbol{x}, \boldsymbol{x}^{(N)})]^T. \tag{14}$$

The estimation of the parameters can be carried out using the maximum likelihood method. Note that it is necessary to perform matrix inversions for estimating the output in the kriging model, which increases the computational expense significantly when the dimensionality becomes high.

4.3 Neural Networks

Neural networks have shown to be effective tools for function approximation. Both feedforward multilayer perceptrons and radial-basis-function networks have widely been used.

- **Multilayer perceptrons** An MLP with one input layer, two hidden layers and one output neuron can be described by the following equation:

$$y = \sum_{l=1}^{L} v_l f(\sum_{k=1}^{K} w_{kl}^{(2)} f(\sum_{i=1}^{n} w_{ik}^{(1)} x_i)), \tag{15}$$

  where, n is the input number, K and L are the number of hidden nodes, and $f(\cdot)$ is called activation function, which usually is the logistic function

$$f(z) = \frac{1}{1 + \mathrm{e}^{-az}}, \tag{16}$$

  where a is a constant.

- **Radial-Basis-Function Networks** The theory of radial-basis-function (RBF) networks can also be tracked back to interpolation problems [40]. An RBF network with one single output can be expressed as follows:

$$y(\boldsymbol{x}) = \sum_{j=1}^{N} w_j \boldsymbol{\phi}(\| \boldsymbol{x} - \boldsymbol{x}^{(j)} \|), \tag{17}$$

  where $\boldsymbol{\phi}(\cdot)$ is a set of radial-basis functions, $\| \cdot \|$ is usually a Euclidean norm, the given samples $\boldsymbol{x}^{(j)}, j = 1, \cdots, N$ are the centers of the radial-basis function, and w_j are unknown coefficients. However, this model is expensive to implement if the number of samples is large. Therefore, a generalized RBF network is more practical

$$y(\boldsymbol{x}) = \sum_{j=1}^{L} w_j \boldsymbol{\phi}(\| \boldsymbol{x} - \boldsymbol{\mu}^{(j)} \|). \tag{18}$$

  The main difference is that the number of hidden nodes (L) is ordinarily smaller than the number of samples (N), and the centers of the basis functions ($\boldsymbol{\mu}^{(j)}$) are also unknown parameters that have to be learned. Usually, the output of a generalized RBF network can also be normalized:

$$y(\boldsymbol{x}) = \frac{\sum_{j=1}^{L} w_j \boldsymbol{\phi}(\| \boldsymbol{x} - \boldsymbol{\mu}^{(j)} \|)}{\sum_{j=1}^{L} \boldsymbol{\phi}(\| \boldsymbol{x} - \boldsymbol{\mu}^{(j)} \|)}. \tag{19}$$

4.4 Comparative Remarks

There are several papers that compare the performance of different approximation models [11, 12, 18, 49, 48, 23]. However, no clear conclusions on the advantages and disadvantages of the different approximation models have been drawn. This is reasonable not only because the performance may depend on the problem to be addressed, but also because more than one criterion needs to be considered. The most important factors are accuracy, both on training data and test data, computational complexity and transparency. It has been found in [24] that an approximate model may introduce false optima, although it has very good performance on the training data, refer to Fig. 1. This is more harmful than a lower approximation accuracy if the model is used in global optimization such as evolutionary optimization. Methods to prevent a neural network model from generating false minima have been suggested in [24], which are very effective for lower dimensional problems.

Although it is difficult to provide explicit rules on model selection, some general remarks can still be made on different approximation models. Firstly, it is recommended to implement first a simple approximate model for a given problem, for example, a lower order polynomial model to see if the given samples can be fit reasonably. If a simple model is found to underfit the samples, a model with higher complexity should be considered, such as higher order polynomials or neural network models. However, if the input space (design space) is high-dimensional and the number of samples is limited, a neural network model is preferred. It is recalled that to estimate the unknown parameters of a second-order polynomial model, at least $(n + 1) \times (n + 2)/2$ data samples are required. Otherwise, the model will be undetermined.

Secondly, if a neural network model, in particular a multilayer perceptrons network is used, it is necessary

to consider regulating the model complexity to avoid overfitting. It may also be necessary to try other more efficient training methods [44] if the gradient descent based method is found to be of slow convergence. Besides, RBF networks have found to be of good accuracy as well as of fast training in some studies [48, 23].

5 Data Sampling Techniques

If an approximate model is used for evolutionary computation, both off-line and on-line training will be involved if the evolution is controlled. Off-line learning denotes the training process before the model is used in evolutionary computation. In contract, on-line learning denotes the update of the model during optimization. Usually, the samples for off-line learning can be generated using Monte-Carlo method, however, it has been shown in different research areas that active selection of the samples will improve the model quality significantly. During on-line learning, data selection is strongly related to the search process.

5.1 Off-line Data Sampling

Several data sampling methods have been suggested in the fields of design of experiments [33, 20], statistics and machine learning. Some popular methods are:

- **Design of experiments (DOE)** Orthogonal arrays (OA), central composite designs (CCD), and D-optimality are most widely used in design of experiments. A first-order orthogonal design is one for which $\boldsymbol{X}^T\boldsymbol{X}$ is a diagonal matrix, where $\boldsymbol{X}$ is the extended sample array as defined in Eqn. (3). In other words, the columns of $\boldsymbol{X}$ are mutually orthogonal.

  Central composite design enables the efficient construction of second-order polynomial models. CCDs are basically first-order (2^n) designs augmented by $2n$ additional center and "star" points to allow estimation of the coefficients of a second-order model. An example of CCD designs is given in Fig. 5 for a two-dimensional problem.

  D-optimality takes advantage of the properties of polynomial models in data sampling. The accuracy of the least square estimate in Eqn. (5) is defined as:

  $$\mathrm{Var}(\hat{\Theta}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2, \qquad (20)$$

  where σ^2 is the variance of the estimate error. From Eqn. (20), it can be seen that to improve the quality of fit, one should maximize the determinant of $\boldsymbol{X}^T\boldsymbol{X}$. Therefore, the D-optimality is to select the samples in such a way that the determinant of $\boldsymbol{X}^T\boldsymbol{X}$ is maximized. Although developed from the polynomial models, the D-optimality has also shown to be beneficial in data selection for constructing neural networks [13].

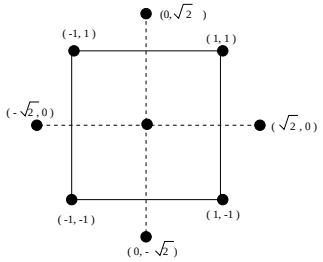


Figure 5: Central composite designs for $n = 2$. The dots represent the sample points. The samples on the solid lines are the first-order design and those on the dashed lines are central and star points ($\sqrt{n}$).

- **Active Learning** Active learning has widely been studied in the field of neural network learning [32, 51, 28]. The basic idea is to select the location of the next sampling data in such a way that an objective function is optimized. The objective function can be information gain, entropy reduction, or generalization error. It has been shown that active data selection can improve the generalization ability of neural networks without increasing the number of training samples.

5.2 On-line Data Sampling

Active data selection is also important in the case that the training data has been collected and therefore, the target is how to select a subset of the data for efficient training.

- **Bagging and boosting** Bagging [7] and boosting [17] are two statistical learning methods that have been developed to improve the quality of approximate model using bootstrap techniques [15]. In bagging (bootstrap aggregating), a number of bootstrap models are constructed using different bootstrap samples and the final output is the average of the models. It is shown that bagging is able to reduce the variance of estimate error efficiently. An adaptive bagging technique can reduce both variance and bias [8].

  Boosting algorithms are able to boost a weak learning algorithm into a strong one. A weak algorithm can be inaccurate rules of thumb that are slightly better than a random guess. The main

difference between boosting and bagging is that in boosting, the bootstrap samples are affected by the performance of the current model. In addition, the final output is a weighted average of the different models.

- **Active data selection** Some of the statistical active learning methods can also be applied to this type of data selection [39]. A special case of integrated mean square error, called integrated squared bias is used as the criterion to select a subset from available data to improve learning performance. However, it is assumed that the data is noiseless.

- **Data weighting guided by evolution** In [25], a method to weight the available data using the information from the evolutionary algorithm has been suggested. The basic idea is that if information on search direction of the evolutionary algorithm is available, then a larger weight should be given to the data samples located in the region where the evolutionary algorithm will most probably visit in the next generation.

  In [42], several strategies for data sampling have been studied. For example, some strategies use the best individuals to replace the worse ones in the training samples, or the ones that are randomly selected. Some strategies create new points randomly and replace the worst ones in the training samples. It has been found that the strategy that simply re-evaluates the best individuals (best in the sense of the approximate model) with the original fitness function exhibits the best performance. This is actually the best strategy in individual-based evolution control.

6 Discussions

Fitness approximation in evolutionary computation is a research area that has not yet attracted sufficient attention in the evolutionary computation community. In fact, the following points still need to be clarified:

- Although several studies have shown very promising results using approximate models in evolutionary computation, it is theoretically still unclear in which way the evolutionary algorithm can benefit from the approximate model. In the least sense, as pointed out in [42], the approximate model can prevent the information in the history of optimization from being lost, although approximate models themselves do not create new information.

- Which type of models helps most, a local one or a global one? It is straightforward to imagine that a global model is able to simplify the search process if the approximate model does not change the properties of the original fitness function. However, from the viewpoint of model construction, to build a local model is much more feasible than to build a global model.

In addition, there are also several topics that deserve further research. Some of them are:

- Development of learning algorithms that are efficient and less sensitive to the number of training data. Learning of problem class [22] and incorporation of *a priori* knowledge [26] are two possible approaches.

- Approximate model with a variable input dimension. During optimization, the input dimension may change in many cases. For example, if an adaptive representation is used in design optimization, the number of parameters increases or decreases during optimization [36].

- Management of different levels of approximation. So far, only functional approximation has been discussed. However, there are different levels of problem approximation in many applications. For example, in computational fluid dynamics simulation, 2D Euler-Lagrange equations, Navier-Stokes equations, quasi 3D simulations and 3D simulations are different approximations of the original problem. Thus, combining different levels of approximation with the approximate model is very interesting.

References

[1] J.-F.M. Barthelemy. Approximation concepts for optimimum structural design - a review. *Structural Optimization*, 5:129–144, 1993.

[2] J. A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of International Computer Music Conference*, pages 131–137, 1994.

[3] J. A. Boyan and A. W. Moore. Learning evaluation functions. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 14–25, Bari, Italy, 1997. Morgan Kaufmann.

[4] J.A. Boyan and A.W. Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machines Learning Research*, 1:77–112, 2000.

[5] J. Branke. Creating robust solutions by means of evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 119–128. Springer, 1998.

[6] J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environment. In L. Spector et al, editor, *Proceedings of Genetic and Evolutionary Computation*, pages 243–250, San Francisco, CA, July 2001. Morgan Kaufmann.

[7] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[8] L. Breiman. Using adaptive bagging to debias regressions. Technical Report 547, Dept. of Statistics, University of California, Berkeley, 1999.

[9] A. J. Brooker, J. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17:1–13, 1998.

[10] L. Bull. On model-based evolutionary computation. *Soft Computing*, 3:76–82, 1999.

[11] W. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. Technical Report 92-2247, AIAA, 1992.

[12] W. Carpenter and J.-F. Barthelemy. Common misconceptions about neural networks as approximators. *ASCE Journal of Computing in Civil Engineering*, 8(3):345–358, 1994.

[13] M.H. Choueiki and C.A. Mount-Campbell. Training data development with the d-optimality criterion. *IEEE Transactions on Neural Networks*, 1999.

[14] J. Dennis and V. Torczon. Managing approximate models in optimization. In N. Alexandrov and M. Hussani, editors, *Multidisciplinary design optimization: State-of-the-art*, pages 330–347. SIAM, 1997.

[15] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[16] J.M. Fitzpatrick and J.J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.

[17] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[18] A.A. Giunta and L. Watson. A comparison of approximation modeling techniques: Polynomial versus interpolating models. Technical Report 98-4758, AIAA, 1998.

[19] D.E. Grierson and W.H. Pak. Optimal sizing, geometrical and topological design using a genetic algorithm. *Structural Optimization*, 6(3):151–159, 1993.

[20] R.T. Haftka, E.P. Scott, and J.R. Cruz. Optimization and experiments: A survey. *Applied Mechanics Review*, 51(7):435–448, 1998.

[21] P. Hajela and J. Lee. Genetic algorithms in multidisciplinary rotor blade design. In *Proceedings of 36th Structures, Structural Dynamics, and Material Conference*, New Orleans, 1998.

[22] M. Hüsken and B. Sendhoff. Evolutionary optimization for problem classes with lamarckian inheritance. In *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 98–109, 2000.

[23] R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodeling techniques under miltiple modeling criteria. Technical Report 2000-4801, AIAA, 2000.

[24] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann, 2000.

[25] Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 592–599, May 2001.

[26] Y. Jin and B. Sendhoff. Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, 10(3):231–242, 1999.

[27] B. Johanson and R. Poli. GP-music: An interactice genetic programming system for music generation with automated fitness raters. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Proceedings of the Third Annual Conference on Genetic Programming*, pages 181–186, 1998.

[28] F. Kenji. Statistical active learning in multilayer perceptrons. *IEEE Transactions Neural Networks*, 11(1):16–26, 2000.

[29] S. Kodiyalam, S. Nagendra, and J. DeStefano. Composite sandwich structural optimization with application to satellite components. *AIAA Journal*, 34(3):614–621, 1996.

[30] J. Lee and P. Hajela. Parallel genetic algorithms implementation for multidisciplinary rotor blade design. *Journal of Aircraft*, 33(5):962–969, 1996.

[31] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscape. *International Journal of Knowledge-based Intelligent Engineering Systems*, 4(3):172–183, 2000.

[32] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):305–318, 1992.

[33] R. Myers and D. Montgomery. *Response Surface Methodology*. John Wiley & Sons, Inc., New York, 1995.

[34] P.B. Nair and A.J. Keane. Combining approximation concepts with algorithm-based structural optimization procedures. In *Proceedings of 39th AIAA/ASMEASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1741–1751, 1998.

[35] A. Neumaier. Molecular modeling of proteins and mathematical prediction of protein structures. *SIAM Review*, 39(3):407–460, 1997.

[36] M. Olhofer, Y. Jin, and B. Sendhoff. Adaptive encoding for aerodynamic shape optimization using evolutiona stratgies. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 576–583, May 2001.

[37] A. Piccolboni and G. Mauri. Application of evolutionary algorithms to protein folding prediction. In J.-K. Hao et al, editor, *Proceedings of the Artificial Evolution 97*, volume 1363 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 1997.

[38] S. Pierret. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *ASME Journal of Turbomachinery*, 121(3):326–332, 1999.

[39] M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, 1993.

[40] M. Powell. Radial basis functions for multi-variable interpolation: A review. In C. Mason and M.G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Oxford University Press, Oxford, U.K., 1987.

[41] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96, 1998.

[42] A. Ratle. Optimal sampling strategies for learning a fitness model. In *Proceedings of 1999 Congress on Evolutionary Computation*, volume 3, pages 2078–2085, Washington D.C., July 1999.

[43] J. Redmond and G. Parker. Actuator placement based on reachable set optimization for expected disturbance. *Journal Optimization Theory and Applications*, 90(2):279–300, August 1996.

[44] R.D. Reed and R.J. Marks II. *Neural Smithing*. MIT, Cambridge, MA, 1999.

[45] Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history. In M. Schoenauer et al, editor, *Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*. Springer, 2000.

[46] G. Schneider, J. Schuchhardt, and P. Wrede. Artificial neural networks and simulated molecular evolution are potential tools for sequence-oriented protein design. *CABIOS*, 10(6):635–645, 1994.

[47] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.

[48] W. Shyy, P. K. Tucker, and R. Vaidyanathan. Response surface and neural network techniques for rocket engine injector optimization. Technical Report 99-2455, AIAA, 1999.

[49] T. Simpson, T. Mauery, J. Korte, and F. Mistree. Comparison of response surface and Kriging models for multidiscilinary design optimization. Technical Report 98-4755, AIAA, 1998.

[50] H. Takagi. Interactive evolutionary computation. In *Proceedings of the 5th International Conference on Soft Computing and Information / Intelligent Systems*, pages 41–50, Iizuka, Japan, October 1998. World Scientific.

[51] S. Vijayakumar and H. Ogawa. Improving generalization ability through active learning. *Neural Computing*, 1998.

A Two Levels Evolutionary Modeling System For Financial Data

Zhou Kang
Computation Center,
Wuhan University,
Wuhan, 430072, China

Yan Li
Computation Center,
Wuhan University,
Wuhan, 430072, China

Hugo de Garis
Computer Science
Department, Utah State
University, Logan, Utah
84322, USA

Li-Shan Kang
State Key Laboratory of
Software Engineering,
Wuhan University,
Wuhan, 430072, China
kang@whu.edu.cn

Abstract

The discovery of evolutionary laws of financial market is always built on the basis of financial data. Any financial market must be controlled by some basic laws, including macroscopic level, submicroscopic level and microscopic level laws. How to discover its necessity-laws from financial data is the most important task of financial market analysis and prediction. Based on the evolutionary computation, this paper proposes a multi-level and multi-scale evolutionary modeling system which models the macro-behavior of the stock market by ordinary differential equations while models the micro-behavior of the stock market by natural fractals. This system can be used to model and predict the financial data(some time series), such as the stock market data of Dow-Jones index and IBM stock price, and always get good results.

1 INTRODUCTION

The financial data or data get from some real-world systems such as stock market usually are very complex, but any complex system is bound to be controlled by some basic laws, including macroscopic level laws, submicroscopic level laws and microscopic level laws. Consider the following financial data as the time series:

$$x(t_0), x(t_1), \cdots, x(t_m) \qquad (1)$$

where $t_i = t_0 + i?t$, $?t$ is the time stepsize. As for the time series, besides the traditional method of time series analysis[1], evolutionary algorithm is usually used to cope with these data[2][3][4].

Suppose that the financial data are controlled by macroscopic, sub-macroscopic and microscopic rules. We take the multi-level, multi-scale models for analyzing and predicting the financial data. In this paper, we use the ordinary differential equation (ODE) model to describe the macroscopic behavior of the stock market, while we use the natural fractal models (a kind of natural discrete wavelets) to describe its microscopic behavior. In this way, we build a multi-level and multi-scale evolutionary modeling system for financial data. This system provides a strong tool for the analysis and prediction of complex time series. The observed data of the Dow-Jones index and IBM stock price are used as the test data for this system.

The rest of the paper is arranged as follows: in section 2, we introduce the macroscopic ODE model; in section 3, we introduce the microscopic natural fractal model; numerical experiments are given in section 4; and in the end, section 5 is some conclusions.

2 MACROSCOPIC ODE MODEL

The complex time series are usually characteristics of multi-level and multi-scale. Assume that it has two levels: macro and micro. In order to describe it in macroscopic level, the researchers take many kinds of methods to pre-handle the time series.

2.1 DECOMPOSITION OF ORIGINAL DATA

In order to find out macroscopic laws from complex data, the first step is to decompose the original data $x(t_i)$, i=0,1,2,...,m as in (1) into two parts: the smooth part and the coarse part (non-smooth part). We assume that the evolutionary process of the smooth part is controlled by macroscopic factors, and the evolutionary process of the coarse part is controlled by microscopic factors. The smooth part will be modeled by ordinary differential equations (ODE), while the coarse part be modeled by natural fractals (a kind of multi-scale discrete wavelets).

For the time series (1), we decompose it into two parts:

$$x(t_i) = \bar{x}(t_i) + \tilde{x}(t_i) \qquad i = 0,1,\cdots,m \qquad (2)$$

where the smooth part $\bar{x}(t_i)$ is defined as:

$$\bar{x}(t_i)=\begin{cases}\dfrac{1}{i+1}\sum_{j=0}^{i}x(t_j) & i<l\\ \dfrac{1}{l+1}\sum_{j=i-l}^{i}x(t_j) & l\le i\le m\end{cases} \quad (3)$$

and the coarse part:

$$\tilde{x}(t_i)=x(t_i)-\bar{x}(t_i),\ i=0,1,\cdots,m \quad (4)$$

Notice: (3) and (4) is related to the value of the smooth parameter l which is often proportional to m, when l is bigger, the time series $\{\bar{x}(t_i)\}$ is more smooth.

2.2 MACROSCOPIC HIGHER-ORDER ODE MODEL

In this subsection, we will mainly introduce how to model and predict the smooth data $\{\bar{x}(t_i)\}_{i=0}^{m}$. Since the smooth data describe the macro behavior of dynamic system and determine the macroscopic tendency of the system, it is the essential part of observed data. Because it is the smooth part of observed data, we assume that $\bar{x}(t)$ is sufficiently smooth, that is, assume that $\bar{x}(t)\in C_n[t_0,T]$, ⊨n=4.

The modeling problem of the dynamic system $\bar{x}(t)$ is to find an initial value problem of the nth–order ordinary differential equation:

$$\begin{cases}x^{(n)}(t)=f(t,x(t),x'(t),x''(t),\cdots,x^{(n-1)}(t))\\ x^{(i)}(t)\big|_{t=t_0}=x^{(i)},\quad i=0,1,\cdots,n-1\end{cases} \quad (5)$$

such that the mean square error between the values of its solution $x^*(t)$ at t_i, i = 0,1,...,m and the series $\{\bar{x}(t_i)\}$ as small as possible.

Denote

$$\left\|x^*-\bar{x}\right\|\equiv\frac{1}{m+1}\sqrt{\sum_{i=0}^{m}(x^*(t_i)-\bar{x}(t_i))^2} \quad (6)$$

That is to say, to find f in function space F, such that

$$\min_{f\in F}\left\|x^*-\bar{x}\right\|. \quad (7)$$

This problem is solved by evolutionary modeling algorithm described in [7]. The main idea of the algorithm is to embed a genetic algorithm in genetic programming used to discover and optimize the structure of a model, while GA is used to optimize its parameters.

The evolutionary modeling algorithm for higher-order ordinary differential equation can be simply described as follows:

PROCEDURE 1

begin

Initialize population $P(0)=\{p_1(0),p_2(0),\ldots,p_N(0)\}$; (produce N parse trees randomly)

t : = 0;

Evaluate the fitness of $p_i(t)$, i = 1, 2, …, N;

while not terminate **do**

begin

$P_c(t)$:= crossover $\{P(t)\}$;

$P_m(t)$:= mutation $\{P_c(t)\}$;

Evaluate $P_m(t)$;

$P(t+1)$:= selection $\{P_m(t), P(t)\}$;

t : = t + 1;

Output solution of $p_{best}:\bar{x}^*(t_i),i=0,1,\cdots,m+q$;

end

For the details of the process of modeling, please refer to [7].

3 MICROSCOPIC FRACTAL MODEL

For the coarse part $\{\tilde{x}(t_i)\}_{i=0}^{m}$ of the time series (1):

$$\tilde{x}(t_i)=x(t_i)-\bar{x}(t_i),\ \ i=0,1,\cdots,m,$$

we are going to build a multi-scale micro natural fractal model.

3.1 CONSTRUCTION OF NATURAL WAVELETS

Denote $$\bar{x}=\sum_{i=0}^{m}\tilde{x}(t_i)\Big/(m+1) \quad (8)$$

In order to search an l-scale basic natural wavelet of series (4), we divide the series $\{\tilde{x}(t_i)\}_0^m$ into l groups (from row to column to form the following matrix (see Table 1), each column as a group, including l groups), where xi denotes $\tilde{x}(t_i)$.

Table 1

	1	2	...	S+1	...	l
1	x_0	x_1	...	x_S	...	x_{l-1}
2	x_l	x_{l+1}	...	x_{l+S}	...	x_{2l-1}
⋮						
k	$x_{(k-1)\,l}$	$x_{(k-1)\,l+1}$	...	$x_{(k-1)l+S}$		
average	$\bar{x}_1^l$	$\bar{x}_2^l$	...	$\bar{x}_{S+1}^l$	...	$\bar{x}_l^l$

$$\bar{x}_i^l=\left(\sum_{j=1}^{k_i^*}x_{(j-1)l+i-1}\right)\Big/k_i^* \quad (9)$$

where

$$k_i^* = \begin{cases} k & i \geq S+1 \\ k-1 & i > S+1 \end{cases} \quad (10)$$

When $S = l$ -1 , $k_i^* = k$, then $(m+1)/\, l = k$.

Through the points $(t_{i-1}, \bar{x}_i^l), i = 1,2,\cdots,l$ in the $x-t$ plane, we can get a polygonal line as follows:

$$x^l(t) = \begin{cases} \dfrac{(t-t_i)\bar{x}_{i+1}^l - (t-t_{i+1})\bar{x}_i^l}{t_{i+1}-t_i} \\ \quad \text{if } t_i \leq t \leq t_{i+1} \text{ and } 0 \leq i \leq l-2 \\ 0 \quad \text{otherwise} \end{cases} \quad (11)$$

Evidently, the function $x^l(t)$ has a local compact support $[t_0, t_{l-1}]$, we call it l–scale basic natural wavelet.

In order to test whether $x^l(t)$ is a basic natural wavelet of time series (4), we introduce the variance ratio:

$$E_l = \frac{\sum_{i=1}^{l} k_i^* (\bar{x}_i^l - \bar{x})^2 \Big/ (l-1)}{\sum_{i=1}^{l}\sum_{j=1}^{k_i^*} (x_{(j-1)l+i-1} - \bar{x}_i^l)^2 \Big/ (m-l+1)} \quad (12)$$

Assume that E_l has an F-distribution with (l-1, m-l+1) degrees of freedom. For different confidence level a and (l-1, m-l+1) degrees of freedom, we can get $F_a(l\text{-}1, m\text{-}l\text{+}1)$ from a F-distribution table.

If $E_l = F_a(l\text{-}1, m\text{-}l\text{+}1)$, then the time series (4) exists l–scale basic natural wavelet $x^l(t)$ in confidence level a. If El < F_a (l-1, m-l+1), then the series (4) does not exist l–scale basic natural wavelet in confidence level a.

3.2 MICROSCOPIC NATURAL FRACTAL MODEL

In order to build the mathematical model for the coarse part $\tilde{x}(t)$ of the time series (4), we construct a multi-scale natural fractal model with scale l=2, 3, ..., L. The process can be described as follows:

PROCEDURE 2

begin

initialize $\bar{x} := \tilde{x}$; *where* $\tilde{x} = \{\tilde{x}(t_i)\}_{i=0}^{m}$

$x^* := 0$; *where* $x^* = \{x^*(t_i)\}_{i=0}^{m+q}$

for l =2, L, **do**

using $\{\bar{x}(t_i)\}_{i=0}^{m}$ calculate $\bar{x}^l = \{\bar{x}_1^l, \bar{x}_2^l, \cdots, \bar{x}_l^l\}$;

if $E_l = F_a(l\text{-}1, m\text{-}l\text{+}1)$ **then**

$x^* := x^* + x^l$; *where* $x^l(i) = \bar{x}_{j+1}^l,\ j \equiv i (\mathrm{mod}\ l)$

end for

$$\varepsilon := \sum_{i=o}^{m} \frac{\tilde{x}(i) - x^*(i)}{m+1};$$

for $i = 0, m+q$ **do**

$x^*(i) := x^*(i) + \varepsilon$;

end for

end

Remark 1: The output $\{\tilde{x}^*(t_i)\}_{i=0}^{m}$ is the fitting part of $\{\tilde{x}(t_i)\}_{i=0}^{m}$ and $\{\tilde{x}^*(t_i)\}_{i=m+1}^{q}$ is the prediction part of $\tilde{x}(t)$. ε is the average fitting error, and it has been eliminated as the correction (random error).

Remark 2: The first part of the procedure is to test successively whether the time series exists basic natural wavelet with scale l which is less than L, usually L=(m+1)/3, for some special problems, where m is relatively small, L can be magnified to L=(m+1)/2. if it exists, then prolong it periodically to whole interval $[t_0, t_{m+q}]$ and add to the time series $\{x^*(i)\}_0^{m+q}$.

Remark 3: The second part of the procedure is to evaluate the random error of $\{\tilde{x}(t_i)\}$, and then correct it when fitting and prediction.

3.3 THE MULTI-LEVEL AND MULTI-SCALE EVOLUTIONARY MODELING SYSTEM

Using the macroscopic modeling PROCEDURE 1 of nth-order ordinary differential equation (5), and the microscopic modeling PROCEDURE 2 of natural fractal, we can build a multi-level and multi-scale evolutionary modeling system for fitting and prediction of complicated time series. Firstly, call PROCEDURE 1 to build the ODE (5), and use Runge-Kutta method to solve it to get the fitting and prediction values of the smooth part $\bar{x}^*(t_i)$, $i = 0,1,\cdots,m+q$, then call PROCEDURE 2 to get the fitting and prediction values of the coarse part: $\tilde{x}^*(t)$, $i = 0,1,\cdots,m+q$. Adding up these data, we can get the needed fitting and prediction values:

$$\bar{x}^*(t_i) + \tilde{x}^*(t_i) = x^*(t_i),\ i = 0,1,\cdots,m+q$$

This procedure can be described as follows:

PROCEDURE 3

begin

Decompose data x $[0, m]$ into $\bar{x}[0,m]$ and $\tilde{x}[0,m]$;

Call PROCEDURE 1 to get $\bar{x}^*[0,m+q]$;

Call PROCEDURE 2 to get $\tilde{x}^*[0,m+q]$;

for i =0, m **do**

$x^*(t_i) := \bar{x}^*(i) + \tilde{x}^*(i)$;

$e(i) := x(i) - x^*(t_i)$;

endfor

for $i = m+1, m+q$ **do**

$x^*(t_i) := \bar{x}^*(i) + \tilde{x}^*(i)$;

endfor

output $x^*(t_i), i = 0,1,..., m+q;$

output $e(i), i = 0,1,..., m;$

end

Remark 1: The first step of the procedure is to decompose the original time series into two parts: the smooth part and the coarse(non-smooth) part.

Remark 2: The second step of the procedure is to call PROCEDURE 1 to deal with the smooth data $\{\bar{x}(t)\}$ and get an ODE model of and the values of its solution: $\bar{x}^*(t_0), \bar{x}^*(t_1), \cdots, \bar{x}^*(t_{m+q})$, where the first m+1 values are the fitting values of $\bar{x}(t)$, and the later q values are the prediction values of $\bar{x}(t)$.

Remark 3: The third step of the procedure is to call PROCEDURE 2 to deal with the coarse data $\tilde{x}(t)$ and get a multi-scale natural fractal model and its solution: $\tilde{x}^*(t_0), \tilde{x}^*(t_1), \cdots, \tilde{x}^*(t_{m+q})$, where the first m+1 values are the fitting values of $\tilde{x}(t)$, and the later q values are the prediction values of $\tilde{x}(t)$ at $t_{m+1}, t_{m+2}, ..., t_{m+q}$.

Remark 4: The fourth step of the procedure is to combine the fitting values of the smooth part with those of the coarse part of $x(t)$ to get the time series $\{x^*(t_i)\}_0^m$ and the fitting error $\{e(i)\}_0^m$. The fifth step of the procedure is to combine the prediction values of the smooth part with those of the coarse part of $x(t)$ to get the prediction values of $x(t)$ at the time $t_{m+1}, t_{m+2}, ..., t_{m+q}$, where the prediction length q can be decided by the users.

4 NUMERICAL EXPERIMENTS

In this section, we mainly study the applications of multi-level and multi-scale evolutionary modeling system to the financial data.

Firstly we use the smooth data of BUMP problem as the test data of the smooth model.

$$\text{Maximize } f_n(x) = \frac{\left|\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)\right|}{\sqrt{\sum_{i=1}^{n} i x_i^2}}$$

$$\text{subject to } 0 < x_i < 10,\ i = 1,2,\cdots,n,\quad \prod_{i=1}^{n} x_i >= 0.75 \text{ and } \sum_{i=1}^{n} x_i <= 7.5n$$

4.1 MODELING OF SMOOTH SCIENTIFIC DATA

In 1994, Keane [8] proposed the BUMP problem in optimum structural design as follows:

The solutions of the BUMP problem are unknown. According to this problem, Liu proposed a challenge problem in his doctoral dissertation [9] as follows:

$$\lim_{n\to\infty} Max\ f_n(X) \quad \text{s.t.} \quad 0 \le x_i \le 10,\ 1 \le i \le n,$$

$$\text{where} \quad \prod_{i=1}^{n} x_i >= 0.75 \quad \text{and} \quad \sum_{i=1}^{n} x_i <= 7.5n$$

Table2. Solution table of BUMP problem

n	f_n	*n*	f_n	*n*	f_n
1		18	0.79717388	35	0.82743885
2	0.36497975	19	0.79800887	36	0.82783593
3	0.51578550	20	0.80361910	37	0.82915387
4	0.62228103	21	0.80464587	38	0.82896840
5	0.63444869	22	0.80833226	39	0.83047389
6	0.69386488	23	0.81003656	40	0.82983459
7	0.70495107	24	0.81182640	41	0.83148885
8	0.72762616	25	0.81399253	42	0.83226201
9	0.74126604	26	0.81446495	43	0.83226624
10	0.7473103	27	0.81694692	44	0.83323002
11	0.76105561	28	0.81648731	45	0.83285734
12	0.76256413	29	0.81918437	46	0.83397823
13	0.77333853	30	0.82188436	47	0.83443462
14	0.77726156	31	0.82210164	48	0.83455114
15	0.78244496	32	0.82442369	49	0.8318462
16	0.78787044	33	0.82390233	50	0.83526201
17	0.79150564	34	0.82635733		

Liu got the best solutions of the BUMP problem for n = 2,3,...,50 as showed in Table 2, where f_n = Max f_n (x). The best solutions are depicted in Fig. 1.

We want to discover higher-order ODEs to model the time series $f_2, f_3, f_4, ..., f_{50}$. Denote $f_i = f(t_i)$, where $t_i = t_0 + i?t$, t_0 =2, and $?t$ = 0.01.

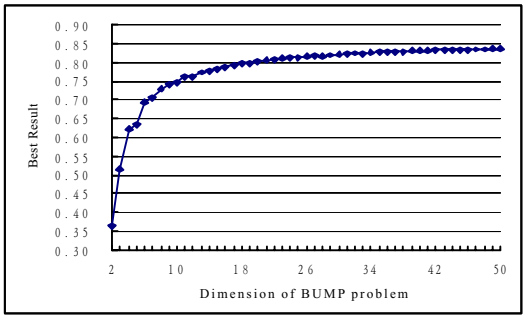


Fig. 1: Best results of f_2 to f_{50}

Using the method described in section 2, we discoverd the following model by computer automatically:

$$d^2 f(t)/dt^2 = -15.658156 \quad df(t)/dt\,(t + df(t)/dt)$$
$$f(2) = 0.36497978$$
$$df(t)/dt\big|_{t=2} = 15.08058$$

Where the modeling error is 0.00095677, this means the model fits the solutions of BUMP problems very well. We use it to predict the solution of the challenge problem by using Runge-Kutta method with ?t =0.01 in 1000000 steps, the results f(100), f(200),..., f(1000 000) are shown in Table 3. The results of f100, f200, ..., f1000000 of the BUMP problem[9] got by Liu on a massively parallel computer are compared in Table 3.

Table 3: the Comparison of f_n and $f(n)$

n	f_n	$f(n)$
100	0.8448539	0.8445141
200	0.8468442	0.84503153
300	0.8486441	0.84503450
400	0.8511074	0.84503451
500	0.8504975	0.84503451
1500	0.8449622	0.84503451
10000	0.8456407	0.84503451
20000	0.8455883	0.84503451
100000	0.8448940	0.84503451
1000000	0.8445861	0.84503451

These results show that the smooth model got by the new modeling system gives a good long-range prediction.

4.2 MODELING OF THE DATA OF DOW-JONES INDEX

The observed data shown in Fig.2 are taken from [10] giving the daily Dow-Jones index over 132 days in 2000. We take the observed data of the first 126 days as historical data (training data) to build models which are used to predict the Dow-Jones index of the last 6 days.

Parameter settings of the modeling experiments are m=126, l=4 for smoothing, m=126, q=6, t_0=0,?t = 0.01 (one day), N=100, n=2 (the second-order ODE) for macroscopic ODE model, and m=126, q=6, L=53,a = 0.1 for microscopic natural fractal model. We get a second-order ODE model as follows:

$$\frac{d^2x}{dt^2} = -17228.009766 + \frac{3672.875732 / \sin(\frac{dx}{dt})}{\sin(\cos t * 1115.356812)}$$

The results are shown in Fig.2.

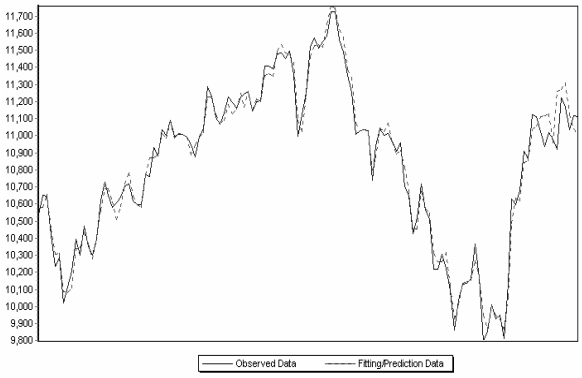


Fig. 2: the fitting and prediction curves for Dow-Jones index

4. 3 MODELING OF IBM STOCK PRICE DATA

The observed data shown in Fig.3 are taken from [10] giving the daily stock price of IBM Company from May 17,1961 to November 2,1962. We take the observed data of the first 359 days as the training data to build models which are used to predict the stock price of the last 10 days.

Parameter settings of the modeling experiments are m=54, l=10 for smoothing, m=359, q=10, t_0=0, ?t=0.01(one day), N=100, n=2 (the second-order ODE) for macroscopic ODE model, and m=359, q=10, L=120,a =0.1 for microscopic natural fractal model. We get a second-order ODE model as follows:

$$\frac{d^2x}{dt^2} = \frac{-1507.55.537}{\cos x} - \cos^2 x$$

The results are shown in Fig.3.

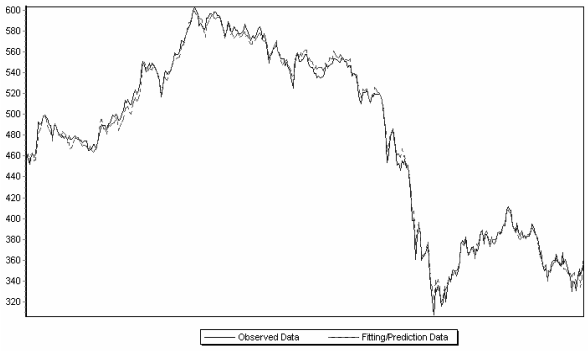


Fig.3: the fitting and prediction curves for IBM stock price

5 CONCLUSION

Compared with most available modeling methods, the multi-level and multi-scale evolutionary modeling system has the following advantages:

Firstly, the entire process is automatic and requires little information in the way of the real-world system or

expertise.

Secondly, it allows one to model the macro-behavior of the system by ordinary differential equations and to model the micro-behavior of the system by multi-scale natural fractals simultaneously.

Finally, the models discovered by computers from the complicated financial data can fit the original data quite well, and the structures of the ODE models are unimaginably to humans.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Nos. 70071042, 60073043, 60133010).

References

[1] C.Chatfield, The Analysis of Time Series: An Introduction, Fifth Edition, Chapman & Hall, 1996.

[2] K.R.Vazquez, Genetic programming in the time series modeling: An application to meteorological data, in Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, Korea, 261-266, May 27-30, 2001.

[3] Y.Liu and X.Yao, Evolving neural network for Hang Seng stock index forecast, in Proceedings of the 2001 IEEE congress on Evolutionary Computation, Seoul, Korea, 256-260, May 27-30,2001.

[4] T-C.Fu, F-L.Chung, V.Ng and R.Luk, Evolutionary segmentation of financial time series into subsequences, in Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, Korea, 426-430, May 27-30, 2001.

[5] C.Hafner and J.Frohlich, Generalized function analysis using hybrid evolutionary algorithms, Proceedings of the Congress on Evolutionary Computation, Washington D.C, USA, Vol.1, 287-294, July 6-9, 1999.

[6] L.Kang, Y.Li and Y. Chen, A tentative research on complexity of automatic programming, Wuhan University Journal of Natural Sciences, Vol.6, No.1-2, 59-62, 2001.

[7] H.Cao, L.Kang and Y. Chen, Evolutionary modeling of systems of ordinary differential equations with genetic programming, Genetic Programming and Evolvable Machines, Vol.1, No.4, 309-337, 2000.

[8] Keane,A.J., "Experiences with optimizers in structural design", in Proc. of the Conf. on Adaptive Computing in Engineering Design and Control 94, ed. Parmee, I.C., Plymouth, 1994, pp. 14 -27.

[9] Liu, P., Evolutionary Algorithms and Their Parallelization, Doctoral Dissertation, Wuhan University, 2000.

[10] R.C. Gan, The Statistical Analysis of Dynamic Data, Beijing, Beijing University of Science and Technology Press, China, 1991.

A Genetic Hybrid for Critical Heat Flux Function Approximation

Yung-Keun Kwon
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu
Seoul, 151-742 Korea
kwon@soar.snu.ac.kr

Sung-Deok Hong
Korea Atomic Energy Research Institute
Deokjin-dong, Yuseong-gu
Daejeon, 305-353 Korea
sdhong1@nanum.kaeri.re.kr

Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu
Seoul, 151-742 Korea
moon@soar.snu.ac.kr

Abstract

Function approximation is the problem of finding a function that best explains the relationship between independent variables and a dependent variable. We propose a genetic hybrid for the critical heat flux function approximation which critically affects the performance of nuclear plants. The problem is represented for genetic algorithm in a way that exploits the relationships between parameters. The experimental result significantly improved the existing function at KAERI (Korea Atomic Energy Research Institute). The framework is not just for the tested problem; it is believed to be applicable to other function approximation problems.

1 Introduction

Given N data pairs $\{X_i, y_i\}$, $i = 1, 2, ..., N$, where each X_i is an n-dimensional vector of independent variables ($X_i =< x_{i_1}, x_{i_2}, \ldots, x_{i_n} >$) and y_i is a dependent variable, the function approximation problem (FAP) is finding a function that best explains the N pairs of X_i and y_i. Assume that the samples are derived from an underlying system of the following form:

$$y_i = f(X_i) + \triangle_i = f(x_{i_1}, x_{i_2}, \ldots, x_{i_n}) + \triangle_i.$$

A popular measure for the error with respect to a candidate function $\hat{f}$ is the LSE (Least Squares Error) which is defined as follows:

$$LSE(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(X_i))^2.$$

In a linear parametric model, we can find an optimal function by traditional regression analysis. When it is nonlinear, we cannot guarantee to find an optimal function in most cases. There have been a number of attempts to do function approximation with trainable dynamic systems using neural nets and fuzzy systems [8][30].

Derivative-based algorithm is a popular approach in a parametric model. However, every derivative-based algorithm converges to the nearest local minimum associated with the initial solution. Thus selecting a good starting point is critical for a derivative-based algorithm. A naive solution for this problem is the multi-start approach which applies a local optimization algorithm, such as a derivative-based algorithm, on a number of random starting points and returns the best result out of them. Another way is the Large-Step Markov Chain (LSMC) method which repeats a chain of "perturbation + local optimization" starting at an initial point. LSMC was popular in the 1990's, particularly for the traveling salesman problem [13][20]. Another is the hybrid genetic algorithms which showed notable successes on combinatorial optimization problems [5][6][21][23]. They generate diverse initial solutions by genetic operators and provide them as inputs for local optimization algorithms.

A hybrid of an adaptive regression splines algorithm and a genetic algorithm was used to solve some FAPs [25][26]. In the regression splines algorithm, the terms in a regression equation take the form of splines of the descriptors. If the numbers of descriptors and functional forms are small, the space of possible fitting equations can be explored exhaustively. However, if the numbers are large, this is not possible. In the hybrid, the search for function models was replaced by a genetic search. It showed better performance with shorter computation times. Another GA approach for designing a universal function approximator with a combination of trigonometric and polynomial basis functions was also proposed [1]. The result was reported to be better than that of a statistical regression

based on polynomials, trigonometrics or cubic splines. It also outperformed a neural-network-based solution.

In this paper, we present a hybrid genetic algorithm for the critical-heat-flux (CHF) function approximation. It is a problem that critically affects the performance of nuclear plants. We use the nonlinear Levenberg-Marquardt algorithm for local optimization and combine it with a genetic search. In a genetic algorithm, it is known that the encoding of solutions significantly affects the performance [5]. We devise a parameter-reordering algorithm for genetic encoding to exploit the geographical relationships of parameters in the genetic search process.

This paper is organized as follows. In section 2, we explain the basics of Levenberg-Marquardt algorithm and critical heat flux, and present the objective. In section 3, we describe our approach for the critical-heat-flux function approximation. In section 4, we provide our experimental results and compare them against existing ones. Finally, conclusions are given in section 5.

2 Preliminaries

2.1 Levenberg-Marquardt algorithm

In linear systems, the steepest-descent algorithm, which moves in the steepest downhill direction determined by the gradient, is the basis for most derivative-based algorithms. Newton's method improves the steepest-descent algorithm by more efficiently determining the movement direction using a Hessian matrix, a matrix of the second partial derivatives. A major disadvantage of Newton's method is that calculating the inverse of the Hessian matrix is computationally expensive and may introduce numerical problems due to round-off errors. If the Hessian matrix is not positive definite, Newton's method may also move to a local maximum (saddle point) instead of a local minimum. Levenberg [18] and Marquardt [19] added a positive definite matrix to the Hessian matrix to make the Hessian positive definite. In this way, one can avoid being directed to a saddle point. This approach is generally called the Levenberg-Marquardt algorithm. For nonlinear systems, the starting point is the Gauss-Newton method which uses a Taylor series expansion to obtain a linear model that approximates the original nonlinear model. Then the least-square methods can be applied. Of course, the Levenberg-Marquardt algorithm can also be applied to this model; it is called the nonlinear Levenberg-Marquardt algorithm.

2.2 Critical Heat Flux

When a heated surface is wet with cooling liquid and most of the heat transferred is absorbed by the latent heat of vaporization, a large heat transfer can be achieved with a small temperature difference between the surface and liquid. However, the region of highly effective boiling heat transfer has a limiting boundary, and the limiting condition is called the *critical heat flux* condition.

The CHF condition is characterized by a sharp reduction of the local heat transfer coefficient which results from the replacement of liquid by vapor adjacent to the heat transfer surface. An occurrence of CHF is accompanied by an inordinate increase in the surface temperature for a surface-heat-flux-controlled system, and an inordinate decrease in the heat transfer rate for a surface-temperature-controlled system [27].

This can be explained with Newton's law of cooling as follows:

$$q = h(T_w - T_f)$$

where q, h, T_w, and T_f represent the heat flux, heat transfer coefficient, wall temperature, and fluid temperature, respectively. If h decreases significantly due to the occurrence of the CHF condition, T_w will increase for fixed q and T_f while q will decrease for fixed T_w and T_f.

The understanding of CHF phenomenon and accurate prediction of the CHF condition are important for safe and economic design of many heat transfer units including nuclear reactors, fossil-fuel boilers, fusion reactors, electronic chips, etc. Therefore, the phenomenon has been investigated extensively over the world since Nukiyama [24] first characterized it.

If CHF occurs in an atomic reactor, it lowers the thermal efficiency and hence causes a serious loss, which endangers the safety. Therefore, it is an important task to predict the occurrence of CHF under a certain condition. To find the CHF function, statistical techniques have been widely studied [2][10]; neural networks [22][31] and genetic programming [17] have also been tried.

2.3 The Objective

There can be a number of measures to evaluate the performance of an approximate function: the sum of the squared errors, the sum of the absolute errors, the maximum overshoot, etc. In our problem, we use LRE (Least Ratio Error) following the convention of the

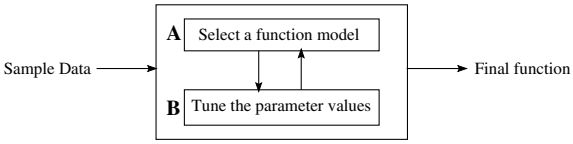


Figure 1: The process of function approximation

CHF studies [7][28]:

$$LRE(\hat{f}) = \frac{\sigma(\frac{y}{\hat{f}(X)})}{E(\frac{y}{\hat{f}(X)})}.$$

To clarify the meaning of the measure, we rewrite it as follows:

$$LRE(\hat{f}) = \sigma(\frac{y}{E(\frac{y}{\hat{f}(X)}) \times \hat{f}(X)}).$$

The final function that we evaluate is $E(\frac{y}{\hat{f}(X)}) \times \hat{f}(X)$ since the new function has the following useful property:

$$E(\frac{y}{E(\frac{y}{\hat{f}(X)}) \times \hat{f}(X)}) = 1.$$

2.4 The Dataset

Each data set consists of eight independent variables $x_1, \ldots, x_8$ and one dependent variable *CHF*. We were given 1607 sets of observed data from KAERI. The best known function with respect to *LRE* that KAERI has from years of tuning is as follows [14][15]:

$$\begin{aligned} CHF \;=\; & -0.019278x_1 - 0.17253\frac{x_2}{1000} \\ & - 0.1396 tanh(0.05461(x_3 + x_4) - 1.97) \\ & - 0.38082x_5 - 0.054003x_6 \\ & - \{\frac{1.8987x_2}{1000} + \frac{0.047388x_1x_2}{1000} - 0.10821x_1 \\ & - 0.67613(\frac{x_2}{1000})^2\}x_7x_8 \\ & + 0.134698x_7 + 1.25103. \end{aligned} \quad (1)$$

Since the actual meaning of the variables are beyond the focus of a methodological study, we renamed the original variables $x_1, \ldots, x_8$.

3 The Suggested GA

3.1 Our Approach

An ideal structure for an FAP is given in Figure 1. Given a sample data set, it repeats the process "i) select a function model, ii) tune the parameter values."

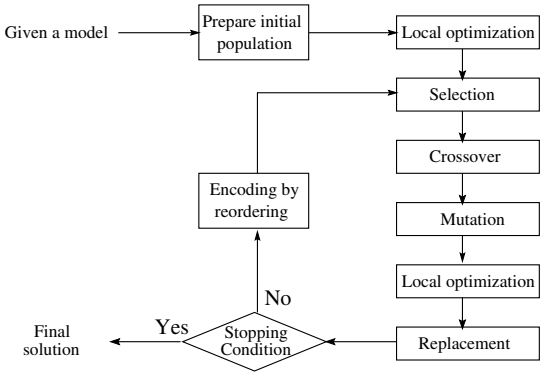


Figure 2: The structure of the RHGA

We may use a two-level genetic algorithm that finds both the function model and the coefficients by two genetic algorithms. In the scheme, the upper level GA provides function models and the other GA tunes the coefficients of each function model. This is an example of non-parametric optimization. The search space may be much wider than the GA can effectively solve in a practical time budget since both the function models and the sets of coefficients have unlimited numbers of eligible candidates. To cut the search space, we start with the best function model at KAERI mentioned in Section 2.4, and attempt to modify it by an analytical method. That is, the GA in this paper is used just for tuning the coefficients. We name this GA a Reordered Hybrid GA (RHGA).

The RHGA was applied to find coefficients in part B of Figure 1. Part A is tuned by an analytical method. The structure of the RHGA is shown in Figure 2. Given the function model (1) of Section 2.4, the coefficient distribution for training is as follows:

$$\begin{aligned} CHF \;=\; & a_1x_1 + a_2\frac{x_2}{1000} \\ & + a_3 tanh(a_4(x_3 + x_4) + a_5) \\ & + a_6x_5 + a_7x_6 \\ & - \{\frac{a_8x_2}{1000} + \frac{a_9x_1x_2}{1000} + a_{10}x_1 \\ & + a_{11}(\frac{x_2}{1000})^2\}x_7x_8 \\ & + a_{12}x_7 + a_{13}. \end{aligned} \quad (2)$$

The problem is to find the best set of coefficients a_1 through a_{13} with respect to the objective *LRE* in Section 2.3. In the following subsections, we describe each part of the RHGA in detail.

3.1.1 Problem Representation by Reordering

In the problem, the coefficients are all real numbers. Each solution is a set of 13 coefficient values. In a

GA, a solution is represented by a chromosome; here, a chromosome is a real array of 13 elements. Although binary representation has been popular in the GA community, real representation also has a long history dating back to the early 1960's [3][29]. Each element of the array is called a gene and we restrict the range of each gene to [-50, 50].

3.1.2 Operations: Selection, Crossover, Mutation

Two parent chromosomes are selected with probabilities that are proportional to their fitness values. The fitness values are normalized in such a way that the best chromosome is chosen with a probability four times higher than that of the worst chromosome. This is a general practice in the GA community [11]. The normalized fitness value of a chromosome in the population is computed as follows:

$$F_k = Q_w - Q_k + (Q_w - Q_b)/3,$$
$$Q_k = \sigma(\frac{y}{\hat{f}_k(X)})/E(\frac{y}{\hat{f}_k(X)})$$

where

F_k : fitness of chromosome k
$\hat{f}_k$: the function corresponding to chromosome k
b, w : the indices of the best and the worst chromosomes in the population

A crossover operator creates a new offspring chromosome by combining parts of the two parent chromosomes. RHGA uses 3-point crossover that works as follows. It randomly selects three cut points in the same positions on both parent chromosomes. The cut points divide each chromosome into four disjoint parts. It makes an offspring by alternately copying the parts from the two parents. RHGA then perturbs the solution with the following mutation operator. It generates a random number for each gene of the offspring. If the random number for the gene is smaller than a preset probability P_1, it is replaced with an arbitrary number in the range [-50,50].

3.1.3 Local Optimization

Local optimization is performed on each offspring after crossover and mutation. Generally a GA is inefficient in fine-tuning around local optima. A local optimization algorithm helps a GA fine-tune and improves its convergence. RHGA uses the nonlinear Levenberg-Marquardt algorithm for local optimization. The Levenberg-Marquardt algorithm takes a set of initial coefficients as input, and outputs a locally optimized set of coefficients. The GA provides diverse initial solutions by crossover and mutation for the Levenberg-Marquardt algorithm.

3.1.4 Replacement Operation and Stopping Criterion

RHGA uses the replacement operator used in [5]. The offspring first attempts to replace the parent more similar to itself, measured by the sum of the distances between all coefficient pairs. If it fails, it attempts to replace the other parent (replacement is done only when the offspring is better than one of the parents). If the offspring is worse than both parents, it replaces the most inferior member of the population. It stops after a given number of generations.

3.2 Reordering and the Modification of Function Models

3.2.1 Coefficient Reordering

A *schema* is a pattern inside chromosomes. Given a set of alphabets S, a schema is defined to be an n-tuple $s_1 s_2 \ldots s_n$ where $s_i \in S \cup \{*\}$. In a schema, the symbol "$*$" specifies the don't-care positions and the other symbols are *specific symbols* which specify the pattern. The *defining length* of a schema is defined to be the length from the leftmost specific symbol to the rightmost specific symbol. We call a schema with k specific symbols a k^{th}-*order* schema. Some schemas survive and some do not by a crossover operator. The survival of high-quality schemas is important since GAs can be explained as a growing process from low-order schemata to high-order schemata [12]. In a single-point crossover, schemas with short defining lengths have higher probabilities to survive over generations. If we use multipoint crossovers, a schema is not disrupted when an even number of crossover points fall between the two specific symbols of every pair of adjacent specific symbols. The survival probability of a schema is not only affected by its defining length and we have to consider the distribution of specific symbols [6]. For example, consider two 6-order schemas H_1 and H_2 with the same defining length of 20. Specific symbols are evenly distributed in H_1 but they are highly clustered in H_2. When two-point crossover is used, the survival probability of H_1 is $45/325$, and that of H_2 is $120/325$. H_2 has a much higher probability of survival.

H_1 : **************************
H_2 : **************************

This example shows the importance of genes' geographical distribution in the chromosomal representation of a GA. If two genes have a strong relationship, it is advantageous to locate them closely [4][5]; in this problem, we suggest a reordering algorithm that uses

```
Calculate Corr(c_i, c_j)(i, j = 1, 2, ..., L);
Find the pair (c_m, c_n)(m ≠ n) having the highest
                                        correlation;
S = c_m c_n;
U = {c_1, c_2, ..., c_L} − {c_m, c_n};
while ( U ≠ ∅ )
{
        Find c_l having the highest value F_l(c_l, S);
        Find c_r having the highest value F_r(S, c_r);
        if ( F_l(c_l, S) > F_r(S, c_r) ) {
                S = c_l · S; //concatenation
                U = U − {c_l};
        } else {
                S = S · c_r; //concatenation
                U = U − {c_r};
        }
}
```

Figure 3: Reordering algorithm

the correlations between all the pairs of coefficients. Figure 3 shows the coefficient-reordering algorithm. In the algorithm, functions F_l and F_r compute the correlation between a coefficient c and a string, S, of coefficients as follows:

$F_l(c, S) = \alpha \times Corr(c, c_1) + (1 - \alpha) \times Corr(c, c_2)$
$F_r(S, c) = \alpha \times Corr(c_k, c) + (1-\alpha) \times Corr(c_{k-1}, c)$,
where
$S = c_1 c_2 \ldots c_k$
$Corr(a, b) = \frac{E[(a-\bar{a})(b-\bar{b})]}{\sigma_a \sigma_b}$
α : a weight.

In computing the correlation between a coefficient and a string S, it only considers the two leftmost or rightmost coefficients in the string S. The reasonable range for α is [0.5, 1]. If α=1, only the leftmost or rightmost coefficient is considered. The main purpose of the reordering is to reduce the probability that a crossover operator separates coefficients with high correlations. The reordering helps the pairs of coefficients having high correlations to stay close in chromosomes.

3.2.2 Modification of Function Models

Although we do not intend non-parametric optimization, we attempt to modify the function model (2) of page 3. We examine whether each term of the function properly explains the data with the solution obtained by RHGA. We modify the function model according to that examination. Formally, we transform the function with respect to a coefficient x_k as follows:

The function model	KAERI	RHGA
$E(y/\hat{f}(X))$	1.0026714	0.9982438
$\sigma(y/\hat{f}(X))$	0.1072649	0.0996506
LRE	0.1069791	0.0998259

Table 1: Quality of KAERI and RHGA model

$$y = G(x_1, x_2, \ldots, x_n)$$
$$\iff G_1(x_k) = G_2(y, x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n).$$

We plot the relationship between $G_2(y,\ x_1,\ \ldots,\ x_{k-1},\ x_{k+1},\ \ldots,\ x_n\ )$ and x_k, and also plot another relationship between $G_1(x_k)$ and x_k. If those two relationships are visibly different from each other, it is considered a signal to modify the function model with respect to the variable x_k. Not all variables are capable of being examined in this way; the variables x_3, x_4, x_5, x_6 and x_8 are those so capable.

We attempted to modify the function model in this way and by adding some linear terms. We have observed that G_1 and G_2 are inconsistent with respect to x_4, x_5, and x_6. We modified the terms relevant to them as follows:

$$\begin{aligned} CHF \quad = \quad & a_1 x_1 + a_2 \frac{x_2}{1000} \\ & + a_3 tanh(a_4 x_3 + a_5 log(x_4) + a_6) \\ & + a_7 x_5 + a_8 x_5^{a_9} + a_{10} x_6 + a_{11} x_6^{a_{12}} \\ & - \{\frac{a_{13} x_2}{1000} + \frac{a_{14} x_1 x_2}{1000} + a_{15} x_1 + \\ & + a_{16} (\frac{x_2}{1000})^2\} x_7 x_8 \\ & + a_{17} x_7 + a_{18} + a_{19} \frac{x_8}{100}. \end{aligned} \tag{3}$$

4 Experimental Results

For robust comparison between the KAERI model and the RHGA model we follow the 10-fold cross-validation approach [9][16]. We randomly split the entire dataset D into 10 mutually exclusive subsets $D_1, D_2, \ldots, D_{10}$ of approximately equal size. The RHGA is trained and tested 10 times; the k^{th} experiment was trained with $D \setminus D_k$ and tested with D_k.

The cross-validation estimate of the average and the standard deviations of the observed CHF value over the predicted value are shown in Table 1. In the table, LRE, described in section 2.3, is the most popular measure for errors in the CHF approximation in the nuclear engineering community. The RHGA approach outperformed the KAERI function by about 7%.

	# of Generations	*LRE*	std-dev	trials
RHGA without reordering	7980.20	0.1006804	0.0005521	50
RHGA with reordering	5105.69	0.1001852	0.0002063	50

Table 2: The Effect of Reordering

Table 2 shows the effect of reordering. In the table, "# of Generations" represents the average generation in which the best solution has appeared. The reordering improved the solution quality in visibly less time.

5 Conclusions

In this paper, we proposed a genetic algorithm for the CHF function approximation problem that combines the genetic search with a nonlinear Levenberg-Marquardt algorithm. The Levenberg-Marquardt algorithm helps the GA to fine-tune, and the GA helps the Levenberg-Marquardt algorithm to overcome its narrow scope. We also proposed a coefficient-reordering algorithm to exploit the geographical relationships of genes in the genetic encoding, which also turned out to contribute to the performance improvement. We should note that the function models were not decided by a search method (e.g., a genetic algorithm) but by analytic modification. It may be worth giving more freedom to the forms of function models under a fully non-parametric optimization model. There is a trade-off. We are sure that giving *full* freedom is not the right approach unless the computing power is strengthened by exponential orders of magnitude. Our current result is 7% better than the best known solution.

Acknowledgement

This work was supported by Statistical Research Center for Complex Systems and Brain Korea 21 Project at Seoul National University. The RIACT at Seoul National University provided research facilities for this study.

References

[1] M. A. Ahmed and K. A. De Jong. Function approximator design using genetic algorithm. In *IEEE Conf. on Evolutionary Computation*, pages 519–524, 1997.

[2] R. W. Bowring. A simple but accurate round tube, uniform heat flux, dryout correlation over the pressure range 0.7-17 mn/m. In *AEEW-R*, page 789, 1972.

[3] H. Bremermann. Optimization through evolution and recombination. *Self-Organizing Systems*, 1962.

[4] T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *International Conference on Genetic Algorithms*, pages 102–109, 1993.

[5] T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.

[6] T. N. Bui and B. R. Moon. GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(3):193–204, 1998.

[7] G. P. Celata, M. Cumo, Y. Katto, and A. Mariani. Prediction of the critical heat flux in water subcooled flow boiling using a new mechanistic approach. *International Journal of Heat and Mass Transfer*, 42:1457–1466, 1999.

[8] J. A. Dickerson and B. Kosko. Fuzzy function approximation with ellipsoidal rules. *IEEE Trans. on Systems, Man and Cybernetics*, 26(4):542–560, 1996.

[9] B. Efron and R. Tibshirani. *Cross-validation and the bootstrap: Estimating the error rate of a prediction rule.* Technical Report (TR-477), Dept. of Statistics, Stanford University., 1995.

[10] Y. Fujita. Prediction methods of heat transfer coefficient and critical heat flux in mixture boiling. *Experimental Heat Transfer, Fluid Mechanics and Thermodynamics*, pages 831–842, 1997.

[11] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[12] J. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, 1975.

[13] I. Hong, A. B. Kahng, and B. R. Moon. Improved large-step markov chain variants for the symmetric TSP. *Journal of Heuristics*, 3(1):63–81, 1997.

[14] D. H. Hwang. Study on the critical heat flux. KAERI/AR-339/91, Korea Atomic Energy Research Institute.

[15] D. H. Hwang, Y. J. Yoo, J. R. Park, and Y. J. Kim. Evaluation of the thermal margin in a KOFA-loaded core by a multichannel analysis methodology. *Journal of Korea Nuclear Society*, 27:518–531, 1995.

[16] R. Kohavi. A study of cross-validation and bootstrap of accuracy estimation and model selection. In *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143, 1995.

[17] D. G. Lee. Critical heat flux prediction using genetic programming for water flow in vertical round tubes. *International Comm. Heat Mass Transfer*, 24:919–929, 1997.

[18] K. Levenberg. A method for the solution of certain problems in least squares. *Applied Mathematics*, 2:164–168, 1944.

[19] D. W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 1963.

[20] O. Martin, S. W. Otto, and E. W. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326, 1991.

[21] P. Merz and B. Freisleben. Genetic local search for the TSP:New results. In *IEEE Conf. on Evolutionary Computation*, pages 159–164, 1997.

[22] S. K. Moon, W. P. Baek, and S. H. Chang. Parametric trends analysis of the critical heat flux based on artificial neural networks. *Nuclear Engineering and Design*, 12:29–49, 1996.

[23] Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem. In *International Conference on Genetic Algorithms*, pages 450–457, 1997.

[24] S. Nukiyama. The maximum values of the heat Q transmitted from metal to boiling water under atmospheric pressure. *Trans: JSME*, 37:357, 1934.

[25] D. Rogers. G/splines: A hybrid of friedman's multivariate adaptive regression splines (mars) algorithm with holland's genetic algorithm. In *International Conference on Genetic Algorithms*, pages 384–391, 1991.

[26] D. Rogers. Evolutionary statics: Using a genetic algorithm and model reduction to isolate alternate statistical hypotheses of experimental data. In *International Conference on Genetic Algorithms*, pages 735–742, 1997.

[27] T. G. Theofanous. The boiling crisis in nuclear reactor safety and perfomance. *International Journal of Multiphase Flow*, 10:69–95, 1980.

[28] K. C. Tu, C. H. Lee, S.J. Wang, and B. S. Pei. A new mechanistic critical heat flux model at low-pressure and low-flow conditions. *Nuclear technology*, 124:243–254, 1998.

[29] R. Weinberg. *Computer Simulation of a Living Cell*. Ph.D. thesis, University of Michigan, 1970.

[30] S. S. Yang and C. S. Tseng. An orthogonal neural network for function approximation. *IEEE Trans. on Systems, Man and Cybernetics*, 26(5):779–785, 1996.

[31] T. C. Yapo, M. J. Embrechts, S. T. Cathey, and R. T. Lahey. Prediction of critical heat fluxes using a hybrid kohonen-backpropagation neural network. *Intelligent Engineering Systems through Artificial Neural Networks*, 2:853–858, 1992.

Search Improvement by Genetic Algorithms with a Semiotic Network

Sang-yon Lee, Sung-Soon Choi and Byung-Ro Moon
School of Computer Science and Engineering,
Seoul National University, Seoul, Korea
{slee,irranum,moon}@soar.snu.ac.kr

Abstract

The explosive expansion of the World Wide Web makes the search problems more challengeable. In this paper we present a search improvement method based on a semiotic connection network and a genetic algorithm. The semiotic connection network expands given keywords to an extended set of keywords. The genetic algorithm tunes up the parameters for search. The experimental results showed 6% improvement over Google's search results. The proposed method was incorporated into a commercial product.

1 Introduction

The quantity of available information in the World Wide Web (WWW) is explosively growing. Effective search became crucial due to the huge volume of information. To date, diverse search methods and exploring agents have been presented.

Since 1994 [12][13][18], lots of Internet search engines have been developed; some of them have been commercially utilized. Early Internet search methods were to find the documents containing requested word strings in a bunch of documents collected from the WWW by crawlers.

However, simple Internet search methods like word-string matching turned out to have problems. With the sharp increase of Web pages, they tended to show a considerable number of useless URLs (Uniform Resource Locators) rather than the Web pages that users want.

A notable approach is to evaluate Web pages from the view point of text documents. There were studies that represent each document as a vector of words included in the document and evaluate documents with the similarities between the vectors [8]. Another notable approach is extended-word method; when a user provides a query, it extends the query by generating additional words [5].

A new approach exploits the fact that Web pages are hypertext documents containing tags such as links and anchors. It evaluates the importance of each Web page with the number of backward links. The Web pages having more incoming links are thought to be more important pages [17]. This approach produced a famous search engine Google [4]. This method includes counting the incoming and outgoing links of a Web page; it led to the study that models the entire WWW structure as a graph, by conceptualizing URLs and links as nodes and edges, respectively [11].

There were studies with stochastic optimization methods for Internet search engines. In [21], genetic algorithm (GA) was used for tuning up parameters of a Web agent for information retrieval. In [22], simulated annealing was used for an Internet search engine.

The link-based analysis exploits the fact that the importance of a Web page depends on the number of incoming links from other Web pages. This method puts emphasis upon the connections of Web pages. It is efficient for finding popular Web sites. However, if a user wants to find a specific content directly, a Web page with only a lot of backward links would not be satisfactory. Besides, it is sometimes not easy to evaluate the relative importance of the documents with few incoming links.

On the other hand, although the content-based analysis exploited various analytic methods, the vagueness of evaluating the importance of documents makes the evaluation difficult. Among many elements present in a document, it is not easy to clarify which elements are important and close to the themes that a user wants to find.

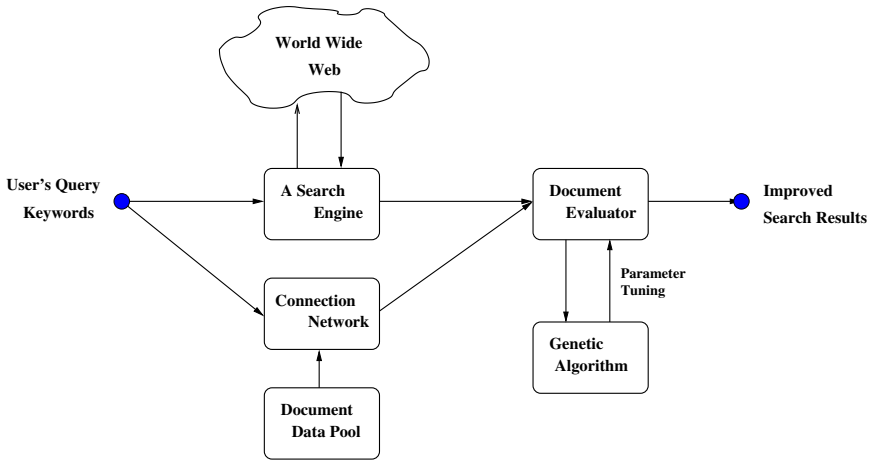


Figure 1: Operational frameworks

In this paper, we suggest a number of elements for document evaluation and optimize them using a genetic algorithm. Our primary purpose is to improve users' satisfaction with search engines.

Semiotics is a general theory of signs and symbols including the analysis of the nature and relationship of signs. The document analysis based on semiotics utilizes the relationship of words appearing in documents. Rather than thinking of each word independently, it takes a relational viewpoint [7]. We reflect the semiotic relationship of words into a connection network and use it for genetic evaluation of documents. The system uses GA as a method of evolutionary optimization to tune up the ranking factors deciding the relative importance of Web pages

The rest of this paper is organized as follows. In section 2 the architecture of the system and the data for experiments are presented. Section 3 deals with the methodology on how to use the connection network to expand keywords and how to use TF-IDF (Term Frequency, Inverse Document Frequency) for document evaluation. In section 4, we describe the parameter tuning by genetic algorithms. In section 5, we give our experimental results and compare our results with those of Google. Finally, the conclusion is given in section 6.

2 System Architecture

The purpose of this study is to find a method to improve the results of search engines. We designed the system to begin with the results of existing search engines. For experiments, the system was designed as a type of a meta-search engine, and we utilized Google search engine (www.google.com). We chose Google since it is one of the best Internet search engines.

The system begins with a considerable number of results from a search engine. In this experiment, we used the 100 top-ranked pages. In some cases, the search engine provides less than 100 results. We also excluded broken Web pages.

The system gets an expanded word list associated with the query. The expansion is performed by the connection network (CN). The Web pages are evaluated using the expanded vocabulary. Each page is transformed to a vector with TF-IDF method [20] [8] before the evaluation. Finally, Web pages are ranked by the document evaluator which was tuned by GA.

3 Methodology

3.1 Connection Network

To date, a number of techniques were suggested for representing the relationship between words as a network. They were used for various fields such as natural language processing, Web search, etc. [15] [5] [7]. These techniques represent the conceptual relationship between words [15], connect the related words [5], or classify words under the categories [7]. However, these techniques considered only the existence of relationship and did not quantify the degree of relationship between words.

The problem of quantifying the relationship between words can be considered as a special case of the "market-basket" problem. The "market-basket" problem is a general problem to quantify the degree of relationship between items in a market. Various metrics have been introduced to quantify the relationship be-

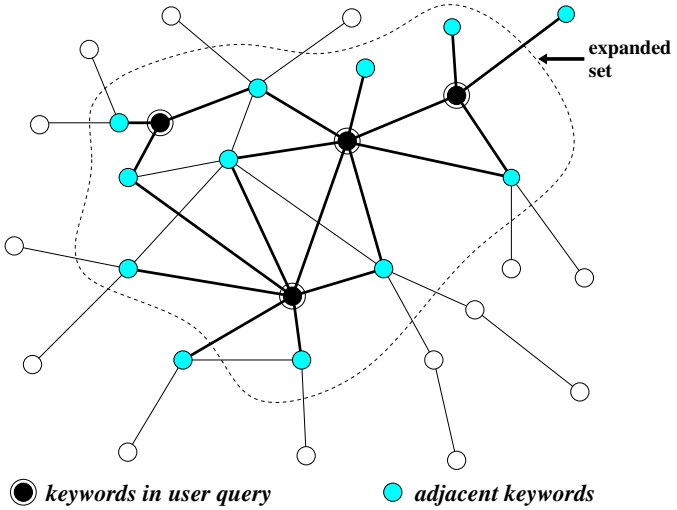


Figure 2: Semiotic connection network

tween items [2] [1] [3] [9] [6]. We design a new metric to quantify the relationship between words and construct a connection network to represent the relationship based on the metric.

3.1.1 Metric for the degree of relationship

Let D be the whole document set and W be the whole word set. For two words $x, y \in W$, we define $n(x)$, $n(y)$ to be the number of documents in which the words x and y occur, respectively, and define $n(x, y)$ to be the number of documents in which both x and y occur. We define the function $f : W \times W \longmapsto R$ to represent the degree of relationship between x and y as follows (R : the set of real numbers) :

$$f(x,y) = \begin{cases} \dfrac{log(n(x,y)) + C}{n(x) + n(y)} & , \text{ if } \quad n(x,y) \neq 0 \\ 0 & , \text{ if } \quad n(x,y) = 0 \end{cases}$$

, where C is a constant.

The function f has a similar shape to *interest* [3] or *similarity* [6] in data mining area.[1]

3.1.2 Keyword Expansion Based on Connection Network

We represent each word as a vertex. For each pair of words x and y such that $n(x, y) \neq 0$, we connect two vertices x and y with an edge and assign $f(x, y)$ as the weight of the edge. Then, we have the connection-network graph $G = (V, E)$ where V is the set of vertices and E is the set of edges.

Using the connection network, we process the expansion of keywords as follows. We assume that a user asked a query which consists of $x_1, x_2, \ldots, x_k$ ($x_i \in W, 1 \leq i \leq k$) . Let $N(x_i)$ be a neighbor set of x_i ($1 \leq i \leq k$) on the connection network. We define the keyword expansion score function $s : W \longmapsto R$ as follows:

$$s(y) = \begin{cases} \displaystyle\sum_{1 \leq i \leq k} f(x_i, y) & , \text{ if } \quad y \in \bigcup_i N(x_i) \\ 0 & , \text{ otherwise .} \end{cases}$$

We choose a set of words with large enough expansion scores.

3.2 TF-IDF (Term Frequency, Inverse Document Frequency)

TF-IDF is a method to represent a document in a vector space [20] [8]. Each word in the document is assigned a scalar value. The scalar value reflects the relative importance of the word in the document and in the whole document set.

For a word w and a document d containing the word, TF(w, d) means the frequency that the word w occurs in the document d. DF(w) means the number of documents in which the word w occurs. IDF is defined as

$$IDF(w) = log \frac{|D|}{DF(w)}$$

where D is the set of all documents and $|D|$ is the

[1] $interest(x,y) = \frac{|D| \cdot n(x,y)}{n(x) \cdot n(y)}$ and $similarity(x,y) = \frac{n(x,y)}{n(x)+n(y)-n(x,y)}$.

number of all the documents.

A vector element $d^{(i)}$ associated with a word w_i is represented by the product of TF and IDF.

$$d^{(i)} = TF(w_i, d) \times IDF(w_i)$$

Then, the document d can be represented by a vector

$$\vec{d} = (d^{(1)}, d^{(2)}, ..., d^{(n)}).$$

From these scalar vectors we can compare the similarities of documents and evaluate a document. The similarity of two documents, V_j and V_k, is defined as

$$Sim(V_j, V_k) = \frac{V_j \cdot V_k}{|V_j| \times |V_k|} \qquad j, k \in \{1, 2, ..., n\}$$

where $V_j \cdot V_k$ means the inner product of V_j and V_k, and $|V_j|$ and $|V_k|$ mean the norms of V_j and V_k, respectively.

3.3 Document Evaluation

To find a document which a user wants the most, we should extract information as much as possible from queries, expanded-words obtained from the connection network, and other elements. Because the elements have different roles and relative importance, we need a process to optimize their roles. To optimize document evaluation, we should choose the documents that users have intended to find.

Let U be a set of words and t be a document. We define the TF-IDF vector related to U and t, $V_{U,t}$, as

$$\overrightarrow{V}_{U,t} = (v_1, v_2, ..., v_{|U|})$$

where $v_i = TF(w_i, t) \times IDF(w_i)$ and $w_i \in U$.

In Section 3.1.2, we showed that the connection network not only gives expanded words, but also gives the real values that represent the strengths between queries and the words. We denote by $S_{U,t}$ the score vector related to a word set U and a document t.

We define the vector $S_{U,t}$ as

$$\overrightarrow{S}_{U,t} = (s_1, s_2, ..., s_{|U|})$$

where

$$s_i = \begin{cases} s(w_i) & , \text{ if } \quad w_i \text{ occurs in } t \text{ for } w_i \in U \\ 0 & , \text{ otherwise .} \end{cases}$$

, and $s(w_i)$ is the keyword expansion score value of w_i in Section 3.1.2

We denote by $W(q)$ and $W(CN)$, the sets of the words in the user query q and the expanded-words by the connection network, respectively. Let the document length of d be $l(d)$.

The attractiveness $e(d)$ of a document d is defined as

$$e(d) = \frac{a_1 |V_{W(q),d}|^{x_1} + a_2 |V_{W(CN),d}|^{x_2} + a_3 |S_{W(CN),d}|^{x_3}}{a_4 \cdot l(d)^{x_4}} \tag{1}$$

.

In the above formula (1), a long document length is a disadvantage.

3.4 The Evaluation of Evaluation Methods

A set of parameters in the formula (1) corresponds to an evaluation method of documents. The GA attempts to find an optimal evaluation method, i.e. an optimal set of parameters. When a parameter set is generated in GA, its fitness has to be evaluated. This is the "evaluation of evaluation methods."

Among the several methods, recall-precision (RP) is usually used as an information retrieval standard for various studies [19][10]. Recall means the returned ratio among all the appropriate documents. Precision means the returned ratio of appropriate documents among all the appropriate documents [10].

Here, we suggest two metrics to evaluate evaluation methods. Internet search engines usually provide a lot of documents in response to a user query. However, the most important would be those in the first page. The first page usually shows around 10 URL-links. We focus on the 10 top-ranked URLs in the training. This is a concept altered from the RP method. Summing up the points of 10 top-ranked links is available if each document was rated previously.

We assume a document set has a ranking order by the points of formula (1) for each query. Let i be the ranking number of a document and $p_{q,i}$ be the rating of i-th ranked documents for a query $q \in Q$ (Q : all the query set).

The first measure for the attractiveness of evaluation methods is as follows :

$$fitness_1 = \sum_{q \in Q} \sum_{i=1}^{10} p_{q,i} \tag{2}$$

Our second measure gives a weight to each of the 10-ranked. From the tenth to the first, we assign 1.1 to 2.0 as the weights.

When i means the rank of a document and $p_{q,i}$ is the rating for the document and a query $q \in Q$, the second

```
steady-state GA
    create initial population P;
    repeat {
        Choose two chromosomes p1, p2;
        offspring = crossover(p1, p2)
        offspring = mutation(offspring)
        replace (offspring, P);
    } until (stopping condition)
    return the best solution;
```

Figure 3: Steady-State GA Framework

a_1	a_2	a_3	a_4	x_1	x_2	x_3	x_4
1.02	2.13	1.03	3.03	2.11	0.54	2.33	0.22

Figure 4: Problem encoding example

measure is defined as

$$fitness_2 = \sum_{q \in Q} \sum_{i=1}^{10} \frac{(21-i)p_{q,i}}{10} \quad (3)$$

We apply the above two expressions (2) and (3) to maximize each of the total amounts.

4 Parameter Tuning by Genetic Algorithm

We use a steady-state GA. The template is given in Figure 3. Based on the formula (1) of Section 3.3, we use the GA to search for an attractive parameter set $S = \{a_1, a_2, a_3, a_4, x_1, x_2, x_3, x_4\}$. The problem is to find the best set S maximizing the fitnesses of formulas (2) or (3).

- **Problem Encoding and Crossover**
  In the problem, the parameters are all real numbers. Each solution is a set of 8 parameter values. In our GA, a chromosome is represented by an array with real numbers. Each element of the array is called a gene and we restrict the range of each gene to [0.01,4].

  In a variable set $S = \{a_1, a_2, a_3, a_4, x_1, x_2, x_3, x_4\}$, four parameters a_1, a_2, a_3, and a_4 are coefficients. We assumed that the ratio among them would not be over 1:400. Because x_1, x_2, x_3, x_4 are exponents, we limit them real numbers in [0,4]. Figure 4 shows an example chromosome.

  We use the arithmetic crossover operator [14, pp.104-5]. It creates a new offspring by assigning the average of the corresponding gene values in the parents for each gene. Figure 5 shows an example crossover operator. We should note that a GA with the arithmetic crossover is prone to premature converge. The diversity of solutions needs to be carefully controlled by mutation.

*Parent*1

a_1	a_2	a_3	a_4	x_1	x_2	x_3	x_4
1.20	2.30	1.03	2.00	2.00	0.50	2.33	0.22

*Parent*2

a_1	a_2	a_3	a_4	x_1	x_2	x_3	x_4
4.80	4.30	1.01	4.00	2.20	0.70	2.67	0.44

offspring

a_1	a_2	a_3	a_4	x_1	x_2	x_3	x_4
3.00	3.30	1.02	3.00	2.10	0.60	2.50	0.33

Figure 5: Arithmetic crossover example

- **Selection, Mutation and Replacement**
  We use the tournament selection to choose parents. A parent chromosome is selected as a result of competition among a member of randomly chosen individuals.

  The GA then perturbs the offspring by mutation operator. It replaces each gene with a random number in the proper range with the probability 1/32.

  We replace a chromosome having the worst fitness in the population with the offspring.

5 Experimental Results

5.1 Experimental plan

The experiment begins with results from a search engine. When we finished the preparation with 33 queries, we had on average 85 Web pages for each query. Table 1 shows the statistics.

The eventual performance of the system relies on the users' satisfaction. The prepared Web pages (2812 in total) were rated from 1 to 5 by 11 people. To avoid any prejudice, we shuffled the pages for the evaluators not to know about the rankings produced by the search engine. The most satisfactory Web pages earned 5, and the least satisfactory pages earned 1. Table 2 shows the distribution of the evaluated results. The average rate of the pages returned by Google was 2.48.

We divide the data set into three disjoint sets. We perform three symmetric experiments on the sets. In each experiment, we choose one of them in turn as the test set and perform training with the other two sets. This is a type of experimental design called k-fold cross validation [16].

Table 1: Statistic of Collected Data

Number of queries	33
Number of web pages	2812
Average number of web pages per query	85
Average rating	2.48

Table 2: Distribution of Ratings

Point	Number of Web-pages
1	1020
2	571
3	486
4	330
5	405
Total	2812

We used a steady-state GA with population size 50. If the same chromosomes are generated for five consecutive iterations, we assume the population has converged and stop the GA.

The training set is again divided into real-training set and validation set. The validation set is not directly used for parameter tuning but used for monitoring over-fitting. The performance on the training set usually shows a monotonic increase; on the other hand, that on the validation set usually shows a bitonic curve. We take the solution that corresponds to the peak of the bitonic curve. Finally, we test with the remaining test set and compare the result with the Google's search result.

5.2 Results

We set $k = 3$ for k-fold cross validation. Both of the two formulas of Section 3.4 were used for evaluation.

The Table 3 and Table 4 show the experimental results. Overall, the suggested system showed 6% improvement of satisfaction against Google's search results.

6 Conclusions and Future Work

In this paper we introduced a search ranking method that uses a semiotic connection network to retrieve contextual words.

From the experimental results, we can conclude that i) the connection network helps a search engine better satisfy the intentions of users, and ii) the GA tunes up parametric factors needed for document evaluation, and helps better ranking.

Table 3: 3-Fold Cross Validation with Formula 2

Test Set	1	2	3	Sum
Google	324	268	315	907
Our System	329	302	326	957
Performance	1.02	1.13	1.03	1.06(Avg)

Table 4: 3-Fold Cross Validation with Formula 3

Test Set	1	2	3	Sum
Google	508.4	411.4	493.7	1413.5
Our System	527.5	467.4	508.4	1503.3
Performance	1.04	1.14	1.03	1.06(Avg)

We should also note that the suggested method is not for a full search engine such as Google, Yahoo, etc; it can be used as an engine inside a full search engine or as a postprocessor for re-ranking the results. Currently it is incorporated into a commercial product.

The connection network is under reinforcement process with more document data. We expect the quality of ranking to be improved with a stronger connection network.

Acknowledgements

This work was partly supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

[1] R. Agrawal, T. Imilienski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, 1993.

[2] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 207–216, 1993.

[3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 255–264, 1997.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Net-*

work and ISDN Systems, 30(1-7):107–117, April 1998.

[5] L. Chen and K. Sycara. Webmate: A personal agent for browsing and searching. In *Autonomous Agents*, Minneapolis, May 1998.

[6] E. Cohen, M. Datar, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Trans. on Knowledge and Data Engineering*, 13(1):64–78, 2001.

[7] Semio Corp. *Connecting to Your Knowledge Nuggets.* Semio Corp. White Paper, 2001.

[8] D. Fragoudis and S. D. Likothanassis. Retriever: A self-training agent for intelligent information discovery. In *IEEE Symposium on Information and Intelligent Agents*, 1999.

[9] R. J. Bayardo Jr. and R. Agrawal. Mining the most interesting rules. In *Proc. of the fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 145–154, 1999.

[10] M. Junker, R. Hoch, and A. Dengel. On the evaluation of document analysis components by recall, precision, and accuracy. In *International Conference on Document Analysis and Recognition*, pages 713–16, Los Alamitos, CA, USA, 1999.

[11] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. S. Tomkins, and E. Upfal. The web as a graph. In *Symposium on Principles of Database Systems*, 2000.

[12] M. L. Mauldin. Lycos: Hunting WWW information. Technical report, CMU, 1994.

[13] O. A. McBryan. GENVL and WWWW: Tools for taming the web. In *First International Conference on the World Wide Web*, CERN, Geneva(Switzerland), 1994.

[14] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer, 1992.

[15] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database. *Int'l Journal of Lexicography*, 3(4):235–244, 1990.

[16] N.J. Nilsson. *Artificial Intelligence : A New Synthesis.* Morgan Kaufmann Publishers, Inc, 1998.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[18] B. Pinkerton. Finding what people want: Experiences with the WebCrawler. In *The Second International WWW Conference*, Chicago, USA, 1994.

[19] G. Salton. *Automatic Text Processing; the Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley Publishing Company, 1989.

[20] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.

[21] T. Taketa and H. Nukokawa. An efficient information retrieval method in WWW using genetic algorithms. In *Parallel Execution on Reconfigurable Hardware (PERH). IEEE.*, 1999.

[22] C.C. Yang, J. Yen, and H. Chen. Intelligent Internet searching engine based on hybrid simulated annealing. In *International Conference on System Sciences*, pages 415–22, Hawaii, USA, 1998.

Antenna Design Using Genetic Algorithms

Derek S. Linden

Linden Innovation Research
PO Box 1601
Ashburn VA 20146-1601
dlinden@lindenir.com

Abstract

Antennas are an important component in any wireless system, as they transform a signal that flows through wires into a signal that propagates through space and back again. How well it does this job is a determining factor in how well a wireless system will operate. Two real-world problems are described in this paper that are solved by antennas optimized by genetic algorithm. The antennas show the ability of the genetic algorithm to allow the designer to optimize an antenna for several different criteria at once, and to create new antennas with very little information from the designer other than general constraints and the desired performance characteristics.

1 INTRODUCTION

Communication, radar and remote sensing systems employ thousands of different types of antennas, and there is an increasing need for them to be high-performance and customized. Traditional methods of designing and optimizing antennas by hand using simulation or analysis are time- and labor-intensive, and limit complexity. Local search techniques are helpful, but because the search spaces of even simple antennas are highly multimodal, the initial guess must be close to the final design, and therefore these methods have limited usefulness.

Evolutionary computation methods like the genetic algorithm (GA) are able not only to optimize performance of existing antenna designs, but also to create new kinds of antennas with highly counterintuitive designs. Using a GA, it is possible to prescribe the desired performance of an antenna and allow the computer to find the parameters for the design.

GAs are being applied to many different antenna designs by many different researchers [1]. GAs and other evolutionary computation techniques are very useful in this field for several reasons, including:

- Antenna principles, which are a subset of electromagnetics and founded on Maxwell's equations, are extremely difficult to understand and grasp intuitively.
- There are many fast antenna simulators, requiring only seconds to produce accurate results.
- Search spaces are highly multimodal and resistant to other forms of numerical and hands-on optimization, yet finding good designs is important to industry.
- Since the GA is naturally robust to local optima, and does not even require an initial guess, the amount of design information the engineer must supply to get a good result is minimal.

The GA has the ability to find new solutions when no known conventional antenna designed with conventional techniques is able to approach the requirements of a particular problem, or when such an antenna is expensive and/or difficult to manufacture.

Most antenna optimizations begin with a conventional design and the GA finds the optimal parameters based on desired conventional characteristics. For instance, an inherently high-directivity design like the Yagi-Uda antenna may be optimized for maximum gain (these terms are defined in the next section). This approach is certainly useful, since even conventional problems are difficult to optimize with most other methods, and the resulting optimized designs will often be better than any found previously.

However, of greater interest is to apply conventional designs to unconventional applications, where the GA has enough degrees of freedom to significantly change the mode of operation of the antenna to suit the new application, and to create new antennas when the amount of engineering constraint is minimal.

As the world goes increasingly wireless, there is a growing number of antenna problems without good solutions. The tracking of hospital patients, biomedical research, wideband data communications, remote sensing, integration of antennas within electronic devices, and many others, are all demanding antennas that meet their needs. Meeting them rapidly and effectively will require new approaches to antenna design, because the traditional method is too limited to keep pace with the rising demand. What the GA provides is a means to explore areas of antenna design previously unsearchable and solve antenna design problems unhindered by the limits of human intuition and experience.

2 ANTENNA BASICS

This section provides a brief tutorial on antenna design concepts that will help the reader to understand the designs described in later sections.

There are many antenna classes, such as reflector antennas (e.g., dish antennas), phased array antennas (consisting of multiple regularly spaced elements), wire antennas, horn antennas, and microstrip and patch antennas. Each of these classes use different structures and exploit different properties of electromagnetic waves. Wire antennas will be the focus of this paper. An antenna is a wire antenna if it is constructed from conductors that are much longer than their width.

A *ground plane*—at its simplest a large, flat piece of metal underneath the antenna—is often used in conjunction with a wire antenna. It acts as a mirror for the antenna above it, and therefore changes the antenna gain pattern. A ground plane can decrease the required height and/or simplify the construction of the wire antenna. The hood or roof of a car acts as a ground plane, and antennas that will be affixed to such places need to be designed for use with one.

Directivity and *gain* are two related qualities in antenna design. Directivity is the ratio of power density being transmitted by an antenna in a particular direction to the average power density being transmitted in all directions. The gain is the directivity multiplied by the ratio of power radiated to power input. Gain takes into account all losses such as loss due to resistance in the antenna, which converts some of the input power into heat, and loss due to mismatch between the transmitter/receiver and the antenna. When the losses are considered to be zero, as in this chapter, the directivity and gain are equal.

Gain is usually expressed in decibels (dB), which relates to a ratio of power or power densities by the following expression: $dB = 10\log_{10}(P_1/P_2)$. In the case of gain, P_2 is the power density of an isotropic radiator that transmits power equally in all directions. The abbreviation dBi refers to gain compared with an isotropic radiator. However, the "i" is sometimes left off, and is understood from context.

A *gain pattern* or *antenna pattern* plots gain magnitude versus angle, showing the proportion of power an antenna transmits in a particular direction. For 2-D antennas, or antennas symmetric in the third dimension, this angle is simply the elevation angle θ. In 3-D, there are two angles that specify a direction: θ and the azimuth ϕ. Figure 1 shows these angles on a set of axes. An antenna is considered to be *directive* if its gain pattern is heavily weighted in one direction. The greater the desired directivity, the larger the antenna must be relative to a wavelength, which is commonly labeled λ. The wavelength is the speed of light divided by the frequency, so a 300 MHz signal has a wavelength of about 1 m.

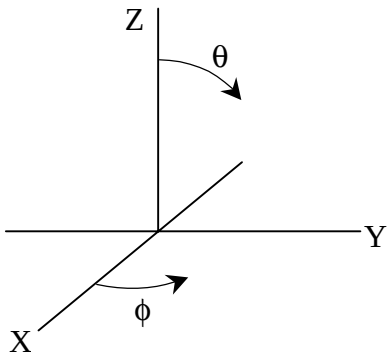


Figure 1: θ and ϕ on a 3-D Axis System

An antenna's *beamwidth* refers to the useful angle span of the so-called main lobe or beam. This lobe usually has the highest gain in the pattern, and is what is of interest to optimize. In a uniform gain pattern, there is only one lobe, but in a directive pattern where the beamwidth is desired to cover only a certain angle span, there can exist other lobes. These other lobes are called *sidelobes*, and usually the designer seeks to minimize them.

Voltage Standing Wave Ratio, or *VSWR*, is a way to quantify the match between an antenna and a device connected to it. A standing wave is created when there is a mismatch in this connection, which prevents power from flowing to and from the antenna. If the standing wave is large, implying a high VSWR, there is a significant mismatch. If it is low, the match is good. A VSWR of 3.0 or less is considered adequate for many low-power applications, while a VSWR less than 1.5 or 2.0 is desired if power considerations are important. A VSWR of 1.0 is a perfect match, and it can never be less than 1. VSWR is easy to measure, and since it is a common parameter specified by antenna designers, it is often an important quantity to optimize.

Bandwidth is the useful range of frequencies for an antenna, and is usually desired to be as large as possible. It is given in percent, which is the ratio of the useful frequency span over the nominal operating frequency. For an antenna operating at 2GHz, a bandwidth of 3% would

mean it would operate over a 60MHz range, from 1.97GHz to 2.03GHz.

Polarization refers to the orientation of electromagnetic waves. Electromagnetic waves are composed of two components: an E-field (electric field) component, which is a sinusoidal wave that exists in one plane, and an H-field (magnetic field) component that exists at right-angles to the E-field. Thus, the wave is asymmetric, and has a definite orientation. Since the H field is constrained by the E-field in a propagating wave, we will discuss just the E-field. In a wave of constant overall magnitude, the E-field magnitude can actually be a time-varying quantity. If one looks at a wave propagating past a fixed point in space, the E-field can oscillate back and forth in a single orientation, giving *linear polarization*, or it can actually be rotating in a circle as its x and y components oscillate back and forth out of phase, giving *circular polarization*. It can also take an orientation in between, giving *elliptical polarization*. Antennas are polarization-sensitive: an antenna that is optimal for picking up linear polarized signals will miss half of the energy of a wave that is circularly polarized and all of a wave that is cross-polarized (linearly polarized at a right angle to the antenna). An antenna that is left-hand circularly polarized (the E-field moves in a circle to the left) will miss a right-hand circularly polarized wave completely. In addition, for ground-to-satellite communications, circular polarization is very helpful because it minimizes the effect of polarization distortion that occurs as a signal travels through the ionosphere.

The next section discusses a conventional design optimized for an unconventional application.

3 A CONVENTIONAL DESIGN AND AN UNCONVENTIONAL APPLICATION: THE YAGI-UDA ANTENNA

As shown in the figure below, the Yagi-Uda antenna is a series of parallel wires, first proposed by Prof. Yagi and his student S. Uda in the late 1920s. One element is driven, one element is behind the driven element and is called the reflector, and all other elements are called directors. The highest gain can be achieved along the axis and on the side with the directors. The reflector acts like a small ground plane, allowing power that would otherwise be sent backward to be reflected forward.

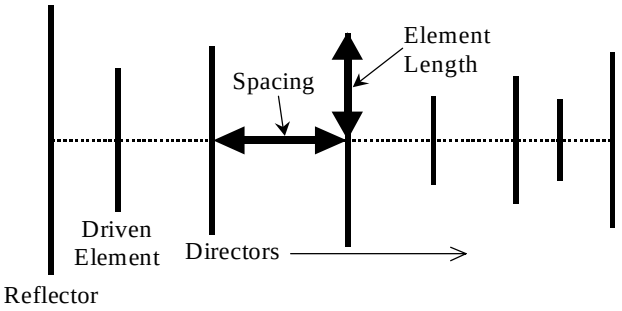


Figure 2: The Yagi-Uda Antenna

The conventional Yagi design includes geometry variables of length for each element, spacing between elements, and the diameter of the wire. Thus, with N elements, there are N length variables, N-1 spacing variables, and one wire diameter variable, giving 2N variables total. The center of the driven element is where the feed is connected.

An unconventional application of this antenna is described in [3], and involves designing a special antenna for the Arecibo 305-meter spherical reflector in Puerto Rico [4]. The antenna was to be used to search for primeval hydrogen having a redshift of approximately 5. Neutral hydrogen line emission is at a frequency of 1420 MHz; thus the frequency region of interest was about 235 MHz. Preliminary studies indicated that the band from 219 to 251 MHz was of the greatest interest, particularly from 223 to 243 MHz. The most important design goal was for the feed to have sidelobes at least 25 dB down from the main beam gain in the region from $70° < \phi < 290°$, due to the interference which came from surrounding radio and TV towers. Of lesser importance was that the E-plane (the plane parallel to the plane of the antenna) and H-plane (perpendicular to the E-plane) beamwidths be about 50°. The VSWR was desired to be under 3.0 and the gain was to be as high as possible, limited by the wide beamwidth. The feed would be mounted over a 1.17 meter square ground plane—that is, a ground plane only 0.92λ in size. The antenna also needed to have a single polarization so that two of them operated at cross-polarity (i.e., each would have a polarization exactly opposite the other), and thus could be used to discriminate between the randomly polarized hydrogen signal and the deliberately polarized signals from the surrounding radio and TV towers. This arrangement would work best if the antennas were 2-dimensional, so that they could be collocated at the same position at right angles.

Since there did not seem to be any conventional antenna that would meet the above specifications, it was decided to use a GA to optimize a Yagi type structure for this unconventional application. Yagi antennas are usually used for high-gain, narrowband applications. The desired bandwidth and beamwidth were very large for this kind of antenna, and the sidelobe requirements were very difficult

to meet. However, a standard Yagi antenna is 2-dimensional and therefore able to meet the polarization and collocation requirements.

The number of wires was specified to be 14. The variables were: the length of each element (14 were required), each constrained to be symmetric and between 0 and 1.5λ, the spacing between each set of two elements (13 were required), constrained to be between 0.05λ and 0.75λ (the total boomlength was allowed to vary), and the diameter of the wire, constrained to be 2, 3, 4, 5 or 6 mm. This wide latitude in parameters allowed the GA to explore very unconventional areas of the Yagi search space.

Note that of the total 28 variables, 27 of them were continuous, real-numbered parameters, making this a natural problem for a real-valued chromosome. The discrete variable—wire diameter—used a real-valued gene, but it was discretized so the GA would only use one of the allowed values. Doing so is usually not recommended for the type of crossover techniques used, but the problem was insensitive to this parameter and it did not affect results adversely.

As expected, the GA produced an antenna that approached the above requirements, though its configuration was quite unconventional for a Yagi antenna. It differed from a conventional Yagi in that the director elements were very closely spaced, its overall length was much less than a typical Yagi with the same number of elements, and its wire lengths varied haphazardly. The genetic Yagi had 13 elements (plus the ground plane) with a boom length of only 1.11λ. The directors varied in length in a seemingly random fashion from about 0.25λ to 0.4λ with an average spacing of less than 0.1λ, as shown in Figure 4. A conventional 14 element Yagi has a boom length about 3 times as long, with directors that are about 0.4λ in length and 0.35λ in spacing, and the lengths become slightly shorter and the spacings slightly larger the greater the distance from the driven element.

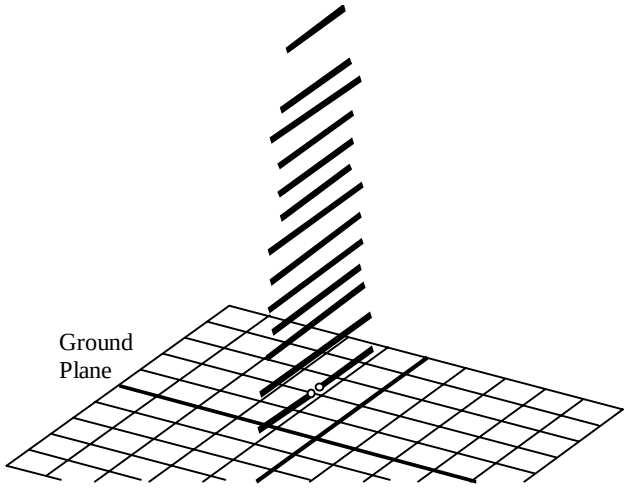


Figure 3: Genetic Yagi Feed for the Arecibo Radio Telescope. From [3].

The performance of this Yagi was computed at 2 MHz increments over the band from 219 to 251 MHz, a bandwidth of 13.6%. The figure below shows the E-plane patterns and H-plane patterns for the genetic Yagi over a finite ground plane at the same frequencies. It is seen that the sidelobe levels for both planes are more than 25 dB lower than the gain at 0° from 223 to 243 MHz, the most important part of the band, and are more than 20 dB lower over the rest of the frequency band. The E- and H-plane half-power beamwidths range from 51 to 55° and 64 to 69° respectively, slightly larger than desired but certainly acceptable. The VSWRs are less than 3.0 from 227 to 245 MHz, though they are higher at the ends of the frequency band. The antenna gain ranged from 10.4 to 11.0 dB over the frequency band. This gain is approximately 1 dB lower than that for a Yagi that is optimized for maximum gain.

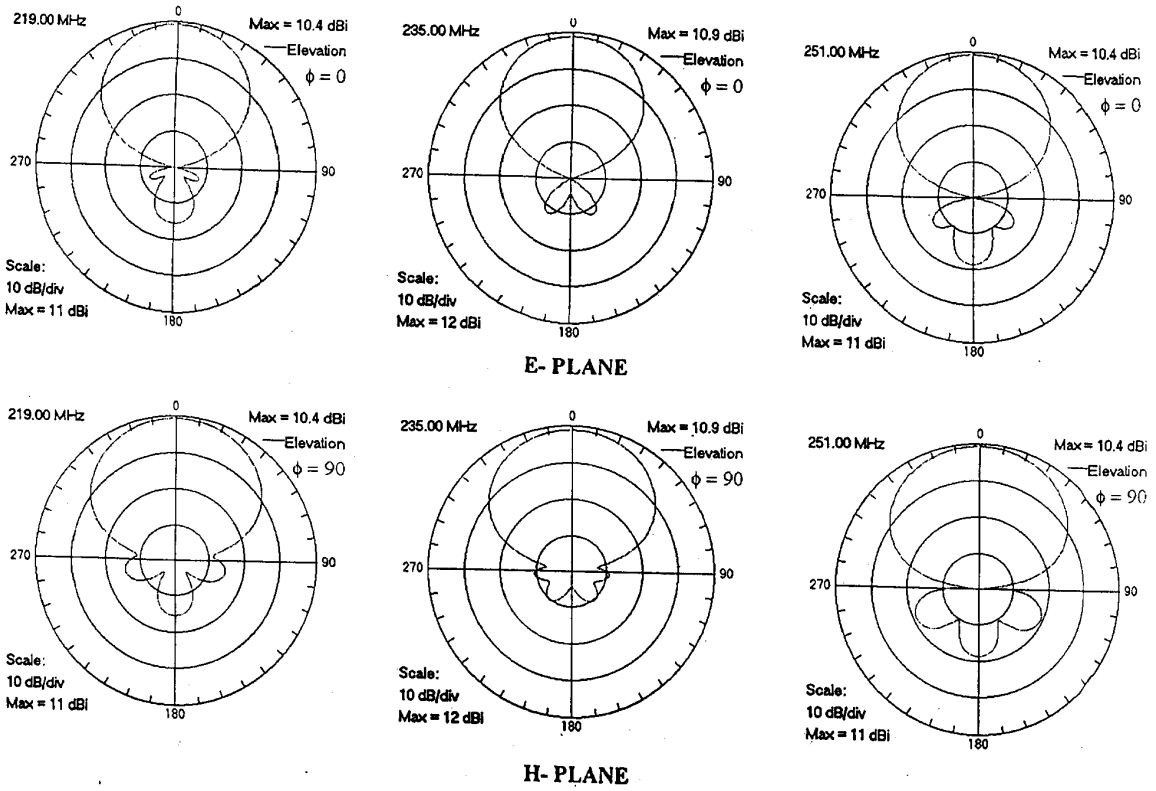


Figure 4: Computed Gain Patterns of Yagi Over a Ground Plane at 219, 235 and 251 MHz. From [3].

The antenna was fabricated to a $1/6^{th}$ scale and the E-plane patterns and VSWR were measured. The computed and measured patterns had reasonable agreement. The measured VSWRs were less than 3.0 over most of the band and had a maximum value of 3.7 near the ends. The measured gains were slightly less than 10 dB; however, if the reflection losses are taken into account, the corrected values for a matched antenna approach the computed gains. For more information about this antenna, see [3] and [8].

This antenna shows the power of the GA to mold conventional designs into new form to solve difficult problems. Naturally, it is important to allow the GA sufficient latitude in the design parameters to change the design from its traditional form to something new.

However, it is not always necessary to specify a design at all, in which case the GA is truly the inventor of an antenna, as the next section will show.

4 UNCONVENTIONAL DESIGN: THE CROOKED-WIRE GENETIC ANTENNA

The application of interest in this section is fairly conventional: ground-to-satellite communications using omnidirectional antennas. Antennas for this application, intended for use on cars or handsets, must be cheap, robust, and have as uniform a gain pattern across the hemisphere as possible for a right-hand circularly-polarized signal, excluding low elevations less than 10° above the horizon, where multipath problems will arise. (*Multipath* refers to the reception of a signal from more than one path, such as receiving a signal from direct line-of-sight and from a reflection off the ground. When the multipath signals do not arrive in phase and at the same time, as generally happens, problems arise such as the ghosting seen on televisions.) This antenna is not trivial to design, and several conventional designs, such as the quadrafilar helix, have emerged to solve this problem to varying degrees. These antennas tend to be expensive and narrowband, and they often require a signal to be fed to the antenna in two places with a precise difference in phase to set up the circular wave. There is thus considerable room for improvement in the state of the art.

There are several qualities that one might desire such an antenna to have, which can be turned into general constraints on the design. For instance, one might desire a single feed point at the base of the antenna for simplicity and low-cost. In addition, it is helpful to have such an antenna over a ground plane. The antenna is expected to be relatively small because near-hemispherical coverage is desired, so it would make sense to constrain the search space to a fairly small volume, for instance, a cube half a wavelength on a side, with the antenna's base located in the center of the bottom face. Doing so will increase the

average speed of simulation while having little impact on results.

It would also be convenient if the antenna were a connected series of straight wires so that there are no precision bends, floating parasitics, or branch points required, for ease in fabrication. 5, 6, 7 and 8 straight wire segments connected in series were thus chosen for investigation. (Preliminary results showed the 7-wire antenna performed slightly better than the others, so 7 wires will be used from here on.) A depiction of the search space that incorporates these constraints is shown in the figure below.

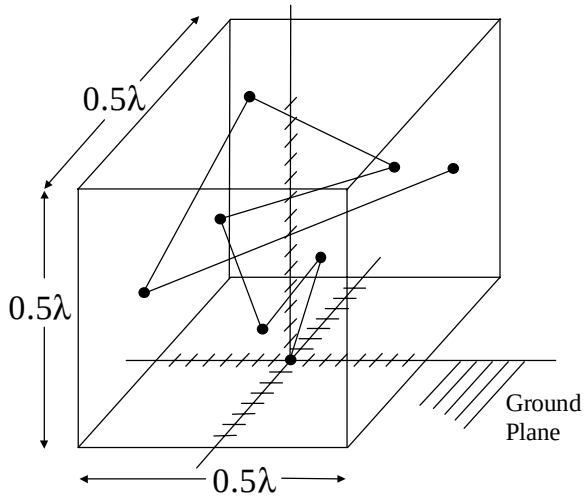


Figure 5: Crooked-Wire Genetic Antenna Search Space for 7 Wires

The GA has performed well up to this point, but it has always had a pre-existing design to use as a template for its work. Can a GA really produce a good antenna with its only engineering knowledge coming from constraints based on convenience?

To begin encoding this problem for a GA, the location of the nodes defining the start and endpoints of the wire segments were mapped into a chromosome. Since each node requires three coordinates, the 21 parameters were placed into a chromosome, i.e., Point1-X, Point1-Y, Point1-Z, Point2-X…, each value encoded into 5 binary bits. 5 bits, corresponding to 32 levels, was chosen because the accuracy of fabrication was not expected to be better than this resolution (3mm at 1600 MHz). Thus, the whole chromosome required 105 bits.

The cost function was then determined for this antenna. The goal was to obtain right hand circular polarization 10° above the horizon at a frequency of 1600 MHz. A good measure of that desired performance can be found in the sum of the squares of the deviation of all calculated gains from the mean. In equation form:

$$\text{Fitness} = \Sigma_{\text{over all } \theta,\phi}(\text{Gain}(\theta,\phi) - \text{Avg. Gain})^2 .$$

The GA's goal was to minimize this fitness. For its first attempt at finding a 7-wire antenna, the steady-state GA had a population of 500 chromosomes, 50% overlap from generation to generation, and a 1% mutation rate. It also used one-point crossover, which was allowed to occur between any two bits in the chromosome with equal probability.

After several hours, the GA converged on a 7-wire configuration with a highly unusual shape, as shown in the inset and the photograph in the figures below. This shape was so unusual and its simulated performance so good that great care was taken to ensure its validity, including building and testing it [9].

The computed radiation patterns of the antenna over an infinite ground plane are shown below for azimuth angles of 0º, 45º, 90º and 135º at a frequency of 1600 MHz. This pattern varies by less than 4 dB for angles over 10° above the horizon—excellent performance, especially since the antenna is so inexpensive and simple to build. (Simple is a relative term, of course, for it does not look all that simple in the figure. However, it was possible to fabricate this antenna by hand using very simple tools, while a conventional design would be tremendously difficult to manufacture by hand.)

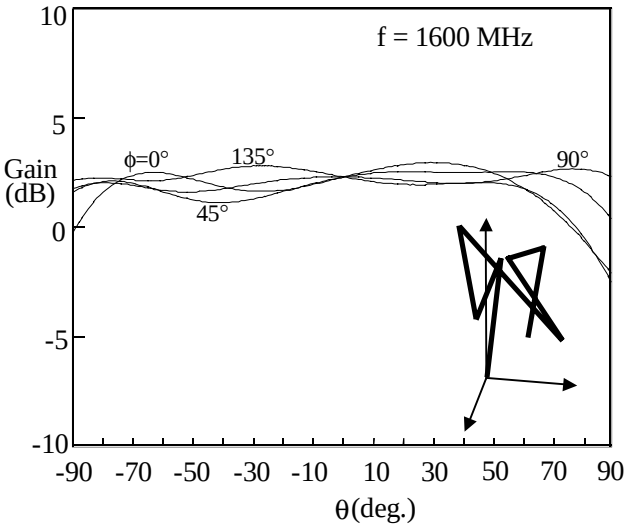


Figure 6: Crooked-Wire Genetic Antenna Radiation Pattern and Diagram. From [6].

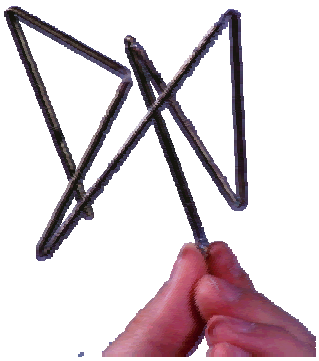


Figure 7: Photograph of the Actual Crooked-Wire Genetic Antenna. From [6].

Although this antenna was only designed to operate at a single frequency, its performance was also investigated for the range of 1300 to 1900 MHz, and it was found to have bandwidth of over 30%, which is excellent for a circularly polarized antenna having near hemispherical coverage.

The antenna was built and measured for its radiation properties. There was about a 6 dB variation in the field above an elevation angle of 10º as compared to the computed variation of about 4 dB. This small discrepancy exists because the measurements were made over a 1.2 m x 1.2 m ground plane, while the computations were performed for an infinite ground plane. Patterns measured over the frequency range from 1300 to 1900 MHz also compared well with the computed patterns.

After this spectacular result, many other GAs, both binary and real-valued, were run for these requirements, and the results were never the same [6]. Two more antennas optimized with the same constraints, chromosome and fitness function are shown below. Though they have nearly the same performance, they are quite dissimilar in shape from each other and from the antenna above. From these and other runs it is apparent that this search space is highly multi-modal, with many minima that give similar performance [6].

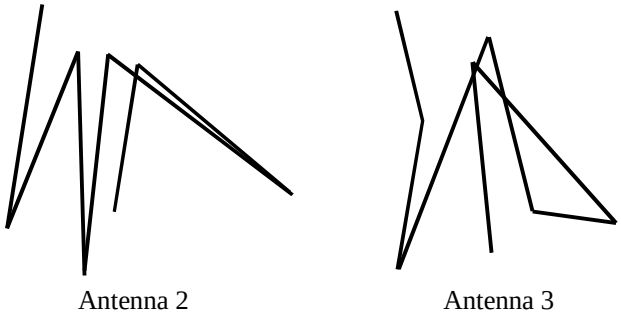


Figure 8: Two More Crooked-Wire Antennas with Nearly Identical Performance

Because of the revolutionary nature of this antenna design process, a patent has been awarded [14]. This patent, which appears to be the first of its kind [15], covers the process of creating a new antenna using no previously known underlying theory of operation and the antennas created by this process.

Though patented by people, the antennas are the innovation of the GA, for it does not have much useful configuration input from the designer, and any constraints which it does have are made for convenience rather than for antenna design soundness.

The design in Figure 7 was the first genetic antenna, and though it has been in existence since 1995, it is an excellent demonstration of the power of the GA. Ongoing research has greatly advanced the state of the art of the genetic antenna, however. It has been applied with great success to several other problems, to include very small antennas [10], uniform gain for low elevation angles over a lossy ground [11], and adaptation of an antenna to its environment for omnidirectional and high-gain applications [12]. For more information about this antenna and other antennas optimized by GA, see also [1, 3, 8, 9,13].

5 CONCLUSION

Each of the antennas described above demonstrated a different quality of GAs as applied to wire antenna design. The Yagi antenna optimized for the Arecibo Feed problem shows how the GA can change conventional designs, using them with unusual parameters, to solve unconventional problems. The crooked-wire genetic antenna shows the raw power of the GA to find not just an optimized design for an application, but to create a new design with minimal help from the engineer.

It may seem extremely surprising that a GA can autonomously find such amazing antennas. However, consider that many people have optimized antennas without any knowledge of electromagnetic theory through the adjustment of their TV's "rabbit ears" antenna. What is used is a rather stochastic local-search technique, based on feedback from the quality of reception, sometimes

even involving haphazard pieces of aluminum foil, to find the best reception. It is usually unknown if the television viewer has found the best possible reception, but the process is stopped once reception has been found that is "good enough" or appears unlikely to improve. The antenna configuration and characteristics are usually quite different from the original V-shaped design, but it works, at least while the surrounding conditions remain constant. In addition, many different configurations will often give the same performance.

Similarly, the GA uses feedback from the antenna simulator to search, somewhat more effectively, the large search space of antenna configurations to find one that is acceptable. As with most complex engineering problems, it is very difficult to tell if the GA has found the best antenna, but often that is not as important as having an acceptable solution. And as in the case of the television antenna, many different antenna configurations may give similar performance, depending on the problem. So while the results shown in this chapter are indeed remarkable, they are not unreasonable, given the nature of antenna design.

In summary, then, GAs are able to optimize wire antennas for diverse and difficult applications. The inherent power of the GA to not only optimize conventional designs, but to create them virtually on its own, makes it an ideal method of automated design for antennas.

6 REFERENCES

[1] *Electromagnetic Optimization by Genetic Algorithms.* Y. Rahmat-Samii and E. Michielssen, eds., Wiley, 1999.

[2] G.J. Burke and A.J. Poggio. "Numerical Electromagnetics Code (NEC)-Method of moments." Rep. UCID18834, Lawrence Livermore Laboratory, January 1981.

[3] E.E. Altshuler and D.S. Linden. "Wire Antenna Designs using a Genetic Algorithm." IEEE Antenna & Propagation Society Magazine, Vol. 39, pp. 33-43, April 1997.

[4] I.M. Avruch, et al., "A Spectroscopic Search for Protoclusters at High Redshift." Bulletin of the American Astron. Society, Vol. 27, No. 4, 1995.

[5] A. Adewuya, "New Methods in Genetic Search with Real-valued Chromosomes," Master's Thesis, Mech. Engr. Dept., MIT, 1996.

[6] D.S. Linden. "Automated Design and Optimization of Wire Antennas using Genetic Algorithms." Ph.D. Thesis, MIT, September 1997.

[7] D.S. Linden. "Using a Real Chromosome in a Genetic Algorithm for Wire Antenna Optimization." Presented at and published in the Proceedings of the IEEE APS International Symposium, Montreal, Canada, 13-18 July 1997.

[8] E.E. Altshuler and D.S. Linden. "Design of Wire Antennas using Genetic Algorithms." *Electromagnetic Optimization by Genetic Algorithms*, Y. Rahmat-Samii and E. Michielssen, eds., Wiley, 1999.

[9] D.S. Linden and E.E. Altshuler. "Automating Wire Antenna Design using Genetic Algorithms." Microwave Journal, Vol. 39, No. 3, March 1996.

[10] E.E. Altshuler. "Small Wire Antenna Design using a Genetic Algorithm." Presented at and published in the Proceedings of the URSI General Assembly, Toronto, Ontario, Canada, 13-21 August 1999.

[11] B.S. Sandlin. "A Wire Antenna Designed for Space Wave Radiation Over the Earth Using a Genetic Algorithm." Ph.D. Thesis, AFIT, 1997.

[12] D.S. Linden. "Wire Antennas Optimized in the Presence of Satellite Structures using Genetic Algorithms." To be presented at and published in the Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 18-25 March 2000.

[13] D.S. Linden. "Creative Antenna Design using Evolutionary Computation." Creative Evolutionary Systems, Peter Bentley and Dave Corne, eds., 2002.

[14] E.E. Altshuler and D.S. Linden. "A Process for the Design of Antennas using Genetic Algorithms." Patent #5,719,794, 17 February 1998.

[15] P. Weiss. "On the Origin of Circuits," Science News, Volume 156, pp. 156-158, September 4, 1999.

Adaptive Reconfiguration of Data Networks Using Genetic Algorithms

David Montana, Talib Hussain and Tushar Saxena
BBN Technologies
10 Moulton Street, Cambridge, MA 02138
{dmontana,thussain,tsaxena}@bbn.com

Abstract

Genetic algorithms are applied to an important, but little-investigated, network design problem, that of reconfiguring the topology and link capacities of an operational network to adapt to changes in its operating conditions. These conditions include: which nodes and links are unavailable; the traffic patterns; and the quality of service (QoS) requirements and priorities of different users and applications. Dynamic reconfiguration is possible in networks that contain links whose endpoints can be easily changed, such as satellite channels or terrestrial wireless connections. We report results that demonstrate the feasibility of performing genetic search quickly enough for online adaptation.

1 INTRODUCTION

There is a growing need for networks to adapt to their operating conditions in order to maintain acceptable levels of performance. Networks must increasingly be able to continue to function effectively despite obstacles such as the disabling of portions of the network by cyberattacks or large fluctuations in the traffic patterns and service requirements. Network adaptation potentially enables not just fine-tuning in response to normal variations but also survivability of the network and its critical applications in the face of catastrophic failures and large-scale shifts in operating conditions.

Although dynamic routing solutions (e.g., [2]) to some of these problems exist, routing has natural limitations. For example, a routing algorithm cannot transmit data between nodes for which cyberattacks have disabled all connecting paths, nor can it transmit a high bandwidth of data between nodes which have only a low-bandwidth path between them. Robust network adaptation requires changes to the underlying network infrastructure (i.e. topology and link capacities) in response to changes in operating conditions.

Despite this need, the problem of automatic, dynamic redesign of functioning networks has received little attention. One reason for this is that network links were traditionally cables and hence not dynamically reconfigurable like satellite or wireless links. Second, the optimization algorithms and computers of the past were not capable of finding a new network configuration fast enough to support adaptive reconfiguration.

In this paper, we investigate the use of a genetic algorithm to dynamically redesign a network with reconfigurable links. Before discussing our work, we provide a brief review of some of the previous work on the use of genetic algorithms for (static) network design.

1.1 PREVIOUS WORK

There is not just one problem in network design but rather a whole family. There are three different components of a network architecture: the topology, the link capacities, and the routing policies. Different problems work with different subsets of these components. There are also three different basic criteria on which to judge a network: cost, reliability, and quality of service (QoS). Different problems use different subsets of these criteria, different measures of these criteria, and combine the criteria they do use in different ways.

A major focus has been minimal spanning tree problems (e.g., [16, 1, 4]). The only network component considered is the topology, and the topology is always a tree. There have been some novel chromosome representations used for these problems, including Prüfer encoding [16, 1] and Huffman trees [8].

When considering factors other than cost, the best topology is generally a graph rather than a tree. Dif-

ferent problems in optimizing non-tree topologies arise from different definitions of the evaluation criteria. For example, [13] and [7] use a probabilistic measure of reliability, while [9] and [17] use a measure of reliability based on redundancy. Given a numbering of all possible links between all pairs of nodes, graph topologies have been genetically represented as fixed length binary strings [13, 17] and as variable-length strings of unique integers [7].

With knowledge of the network traffic patterns, it is also possible to optimize the link capacities and routing policies. In early work, [5] used genetic algorithms to select a set of link capacities given a fixed topology. More recently, with the benefit of greater computational power, researchers have investigated using genetic algorithms to simultaneously optimize topology and link capacities [18, 9] or all three components of the network (topology, link capacity, and routing policies) [12, 11]. It is possible to use a single chromosome that represents all the required information about a network (e.g., [12]) or to use separate representations and solve for the different components in separate (nested) optimizations [11].

2 ADAPTIVE REDESIGN

2.1 PROBLEM STATEMENT

The adaptive network redesign problem is inherently a dynamic problem, since network traffic patterns, requirements and priorities, and available resources (primarily links and nodes) all change with time. In our current work, we consider a snapshot of the problem at a particular time, performing the adaptation by solving for each snapshot independently.

Let us consider a network that contains both fixed (wired) links and reconfigurable links. We use a model for the reconfigurable links that is based upon satellites using a frequency-division multiplexing allocation scheme. There is a fixed amount of total reconfigurable bandwidth available. This bandwidth is unidirectional and is divided into identically sized chunks called *channels*. Reconfigurable links consist of one or more channels configured to have the same source node and destination node. The bandwidth of the channels of a reconfigurable link add, but the bandwidths of a reconfigurable link and a fixed link do not add. Instead, the link with the higher bandwidth is used and the other ignored. Each node has a limit on the number of channels it can send and receive, which is a type of node-degree constraint [4].

The givens of the problem include:

- **available nodes** - This is the set of all nodes not currently disabled by an attack or failure.
- **available fixed links** - This is the set of all fixed links not currently disabled by an attack or failure. Associated with each fixed link is a source node, destination node, capacity, and inherent transmission delay (which is the delay associated with the medium and does not include the delays due to queueing). Note that, for the purposes of our model, all fixed links are unidirectional; bidirectional links are decomposed into two unidirectional ones.
- **available channels** - For a given problem, the total number of channels, bandwidth per channel, and inherent channel transmission delay are fixed.
- **data flows** - Each data flow has associated with it the following information: source node, destination node, priority rating (a positive integer with smaller meaning higher priority), protocol (TCP or UDP), required transmission delay, required dropped packets, and the statistics of the generated traffic. We model the traffic as bursts of data of random number of bytes at random intervals, with Gaussian distributions for the number of bytes and the size of the interval. The mean and standard deviation are the required parameters for each of these distributions. Note that the quality of service (QoS) metrics (i.e., dropped packets and transmission delay) refer to the service as perceived by the application, not the network. In particular, a packet that is initially dropped but successfully retransmitted does not count as dropped but does register a long transmission delay. Hence, the dropped packets metric only applies to UDP flows, since TCP resends all dropped packets.

The variables over which to optimize are:

- **configuration of each channel** - Zero to all available channels may be added to the network topology. The source and destination nodes of each added channel must be specified.

The constraints to obey are:

- **send and receive limits** - The number of channels with a particular node as its source (destination) cannot exceed the send (receive) limit for that node.

The optimization criteria are:

1. **connectivity** - The measure of the degree to which flows are totally disabled due to lack of connectivity is the sum over all disconnected flows of $\frac{1}{\rho_i}$, where ρ_i is the priority rating of the flow (recalling that a lower ρ_i means a higher priority). Note that TCP flows will be disabled if there does not exist a path in both directions between the source and destination (to allow acknowledgements), while UDP flows only

require a path in the forward direction.

2. **meeting transmission delay requirements** - The measure of the degree to which the network does not meet the transmission delay requirements is the sum over all connected flows for which the requirement is not met of $\frac{1}{\rho_i}(D_i - d_i)$, where D_i is the average measured delay (in seconds), d_i is the required delay (in seconds), and ρ_i is the priority rating of the flow.
3. **meeting dropped packets requirements** - The measure for the dropped packets requirements is the sum over all connected flows for which the requirement is not met of $\frac{1}{\rho_i}(P_i - p_i)$, where P_i is the average measured percent of packets dropped, p_i is the required percent of packets dropped, and ρ_i is the priority rating of the flow.

The three optimization criteria are combined into a single score using a weighted sum, $w_1 S_1 + w_2 S_2 + w_3 S_3$, where S_i is the score for the i^{th} criterion. The goal is to minimize this combined score. For our experiments, we used $w_1 = 100$, $w_2 = 1$, and $w_3 = 1$.

2.2 GENETIC ALGORITHM

Representation - Each chromosome is a variable-length list of reconfigurable link *allocations*, where each allocation is a 3-tuple (S, D, C) containing the source node (S), destination node (D), and the number of channels (C) connecting the source to the destination. Only allocations with a non-zero number of channels are included in the list. For example, the chromosome [(6 3 1) (12 2 2)] indicates a reconfigurable link with 1 channel from node 6 to node 3, and a reconfigurable link with 2 channels from node 12 to node 2.

Genetic Operators - We use three operators:

- Crossover - Combine all the allocations from both parents into a single randomly sorted list. Proceed through this list including each allocation in the child chromosome if adding it does not violate any constraints and if no allocation with the same source and destination nodes has already been added.
- Local Mutation - Randomly select one allocation in the parent and randomly choose to either increase the number of channels by one or decrease it by one. If the choice was an increase and if this violates constraints, then attempt to assign the entire reconfigurable link allocation to a different source node or destination node; if none of these produces a legal chromosome, then discard the child.
- Global Mutation - Randomly select a number between half and all-but-one of the allocations in the parent. Randomly select this number of allocations from the parent and add them to the new child. Complete the child by randomly specifying the remaining available channels using the same algorithm as the initialization procedure, described below.

Initialization - The initialization procedure fills the initial population with randomly generated chromosomes. To generate a random chromosome, it specifies one channel at a time until some resource (total channels, node send limits, or node receive limits) has been fully exhausted. For each new channel, it randomly selects source and destination nodes that have not yet exhausted their send and receive limits, respectively. If there is an existing reconfigurable link allocation between the pair of nodes, it adds an additional channel to that allocation; otherwise, it creates a new allocation between the nodes containing one channel.

Evaluation Function - We have modified NS, version 2 [15], a packet-level network simulator, to compute the percentage of dropped packets and the average transmission delay exhibitted, on a per-flow basis, by a network during simulation. The evaluation function first converts the chromosome into a description of the represented network in the format expected by NS. It then starts the modified NS and sends NS the network data. NS performs the simulation and returns the QoS statistics. Finally, the evaluation function uses these statistics to compute the score given in Section 2.1. We use a packet-level network simulator rather than a computationally less expensive approach in order to compute network statistics with greater realism. However, as we discuss below, it is inefficient to restart NS from scratch for every evaluation (particularly because this means restarting its TCL interpreter), and this is something we need to change.

Population Management - The genetic algorithm uses steady-state, worst-one-out replacement. The population allows no duplicate members. Parents are selected probabilistically using *roulette-wheel* selection. Probabilities are distributed exponentially based upon rank. The search terminates when the number of evaluations reached a threshold.

3 EXPERIMENTAL RESULTS

We had two goals for our experiments and two corresponding sets of experiments. The first was to provide concrete examples of the types of problems that adaptive network reconfiguration can solve. The second goal was to investigate the performance of the genetic algorithm, particularly concentrating on scaling with problem size and the tradeoff between the execution time and the quality of the solution found.

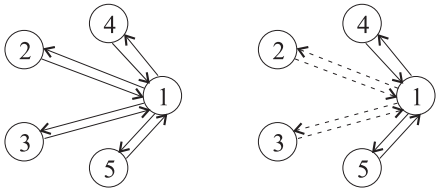


Figure 1: The first network in the sequence. Note that the dotted lines in the topology are the reconfigurable links and the solid lines are the fixed links.

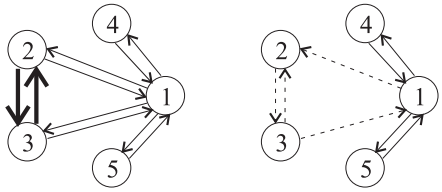


Figure 2: The second network in the sequence. Note that the heavier lines in the flows indicate higher priority traffic.

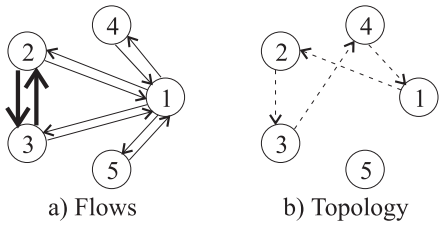


Figure 3: The third network in the sequence.

We approximate the search space size as

$$(M(M-1))^N/N! \quad (1)$$

where M is the number of nodes and N is the maximum number of channels. (There are $M(M-1)$ possible ways to assign a source and destination node to each channel, and hence $(M(M-1))^N$ ways to assign sources and destinations to each of N channels. However, the networks formed are not unique. For any network with no two channels sharing the same source and destination, there are $N!$ different ways to form this network; for other networks, there are less. Hence, Equation 1 is an underestimate but is a good approximation when $N \leq M/2$.)

All of our timing results were performed on a single 850-MHz Pentium. All times are divided into two components: the number of total evaluations, which measures the effectiveness of the genetic algorithm search, and the average time per evaluation, which measures the efficiency of the evaluation function.

In our experiments, all fixed links have a capacity of 1000 kbits/sec (except in the random network experiment) and transmission delay of 10 msecs. Likewise, all channels have a capacity of 1000 kbits/sec and transmission delay of 10 msecs.

3.1 ILLUSTRATIVE EXAMPLES

A Sample Adaptation Sequence - We start by examining a sequence of networks that could be snapshots of a single network as its operating conditions change with time. They illustrate, in a simple-to-understand scenario, the power of adaptive network reconfiguration.

There are three networks in the sequence, each with a maximum total of four channels. The first network in the sequence has five nodes. The traffic flows, pictured in Figure 1a, are typical of a server (node 1) with multiple clients (nodes 2-5). The clients communicate only with the server and not with each other. The server sends 400 kbits/sec to each client, while each client sends 40 kbits/sec to the server, all using the TCP protocol. The priorities are all 5, and the required latency is 10 msecs (since it is using TCP, dropped packets are not a criterion). There exist bidirectional fixed links between nodes 1 and 4 and between nodes 1 and 5. Each node has a limit of 3 send channels and 3 receive channels.

Clearly, the best solution is the one shown in Figure 1b, with four reconfigurable links, each containing 1 channel, that effectively form bidirectional links between nodes 1 and 4 and between nodes 1 and 5. The solution thus provides a single-hop path for all flows.

The second network in the sequence is the same as the first except for the addition of two high-priority (priority 1) flows, one from node 2 to node 3 and the other from node 3 to node 2 (e.g., a teleconference between two CEOs, or communication between two units in battle). The flows are shown in Figure 2a. The new optimal configuration is that shown in Figure 2b, since it provides full connectivity, a one-hop connection for all high priority flows, and a maximum delay of two hops for the lower priority flows.

The third network in the sequence is shown in Figure 3a and is the same as the second except that all the fixed links have been disabled (e.g., due to a coordinated cyberattack). It is now impossible to fully connect all the nodes (4 unidirectional links can connect at most 4 nodes), so there is a choice about which node to leave out of the network. The best configuration is to use the channels to form a ring network between the four of the nodes, three of which must be

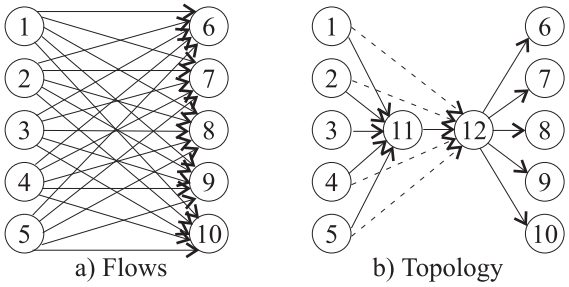


Figure 4: The first bottleneck network.

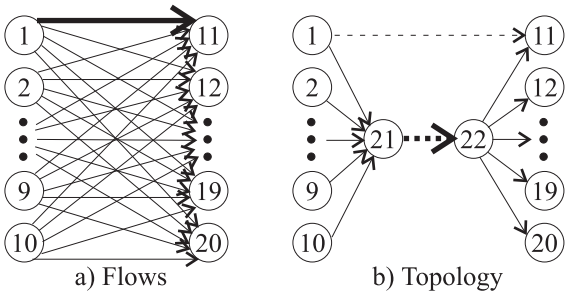


Figure 5: The second bottleneck network. Note that five channels form a single high-capacity link between nodes 21 and 22, replacing the original lower-capacity fixed link.

nodes 1, 2 and 3. An example of such a network is shown in Figure 3b.

The genetic algorithm finds all solutions always in well under 500 evaluations. To perform 500 evaluations requires 4 minutes, or 0.48 secs per evaluation. Almost all of this time (between 0.4 and 0.45 seconds) is spent restarting NS; by eliminating this restart, we could get the runtime down to under 30 seconds.

Bottleneck Networks - We consider two more example networks, larger than the previous ones. Both have paths of fixed links with sufficient capacity to handle the traffic for any individual flow, but there exists a bandwidth bottleneck when considering the flows in aggregate. Reconfigurable links are used to relieve the bottleneck.

The first "bottleneck" network is shown in Figure 4. Each of nodes 1-5 sends data to each of nodes 6-10. All 25 flows are identical: transmitting an average of 200 kbits/sec, using the UDP protocol, having priority 2, and requiring 0% dropped packets (with no requirement on latency). There are four available channels. Without the benefit of the reconfigurable links, all 5000 kbits/sec of the aggregate traffic would travel across the central link between nodes 11 and 12 (which has capacity of only 1000 kbits/sec). An optimal solution is shown in Figure 4, using the reconfigurable links (dotted lines) to relieve this bottleneck by bypassing the central link. (There are five equivalent solutions.)

The second bottleneck network is shown in Figure 5. Each of nodes 1-10 sends data to each of nodes 11-20. All but one of the 100 flows are identical: sending an average of 50 kbits/sec, using the UDP protocol, having priority 100 and required dropped packets 0% (with no requirement on latency). The flow between nodes 1 and 11 differs from the other flows in that it has priority 1, which is much higher than the others, and that it has a required latency of 10 msecs. There are six available channels. As with the first bottleneck network, without reconfigurable links, all 5000 kbits/sec of traffic would travel across the central link between nodes 21 and 22 (which has capacity of only 1000 kbits/sec). The solution is pictured in Figure-5b: use five channels to relieve the bottleneck by replacing the central link with a higher-capacity reconfigurable link, and use the sixth channel to directly connect nodes 1 and 11 and thereby provide the required latency. Note that in going from the first to the second network the optimal strategy changes from bypassing the central link to building up the central link.

The genetic algorithm consistently finds an optimal solution to the first bottleneck problem in under 1000 evaluations. These 1000 evaluations required 20.5 minutes, an average of 1.23 seconds per evaluation. According to Equation 1, the search space size is 1.3×10^7. The genetic algorithm consistently finds the solution to the second problem in under 10,000 evaluations, requiring 394 minutes (2.36 seconds per evaluation). The search space size is 1.4×10^{13}.

3.2 PERFORMANCE INVESTIGATIONS

We investigate the performance of the genetic algorithm on families of networks, where all the networks in a family have the same basic statistical properties but different sizes. This permits us to investigate the scaling properties of the genetic algorithm as a function of the size of the network. We used five different families, one family of "ring" networks plus four families of random networks.

For each network used to explore performance, we performed the same set of experiments, running the genetic algorithm ten times with each of the following sets of parameters:

- $popsize = 20$, $probdecay = 0.7$, $maxevals = 100$
- $popsize = 40$, $probdecay = 0.8$, $maxevals = 300$
- $popsize = 100$, $probdecay = 0.9$, $maxevals = 1000$
- $popsize = 300$, $probdecay = 0.967$, $maxevals = 3000$

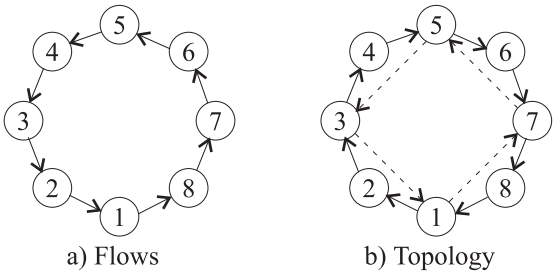


Figure 6: The 8-node, 4-channel ring network.

• $popsize = 1000$, $probdecay = 0.99$, $maxevals = 10000$
Here, *popsize* is the population size, *probdecay* is the parameter that determines the exponential distribution of parent selection probabilities, and *maxevals* is the number of evaluations before terminating the run. A small population size and high selection pressure (small *probdecay*) mean that the genetic algorithm will converge (through loss of diversity) quickly, and are hence appropriate for a short run.

Ring Networks - We start by examining performance on a family of highly contrived networks, which we call "ring" networks. This family of networks has three important properties. First, it contains networks with arbitrarily large and small numbers of nodes and available channels, hence allowing an investigation of how the algorithm scales with network size. Second, each network has a known best solution and hence allows comparison with this known optimum. Third, the optimization problems are especially difficult for a genetic algorithm and hence provide worst-case scenarios.

A member of this family has N channels and a network with $M = kN$ nodes, where k and N are positive integers. There are M identical flows, with one flow from node i to node $(i-1)$ for each $i = 2, ..., M$ and one flow from node 1 to node M. Each flow uses the TCP protocol, has a required latency of 10 msecs, and transmits $\frac{800}{k}$ kbits/sec. There are M fixed links, one from node $(i-1)$ to node i for each $i = 2, ..., M$ and one from node M to node 1. The fixed topology requires packets to travel $(M-1)$ hops. An optimal placement of reconfigurable links connects every k^{th} node in reverse order from the fixed links and reduces the number of hops to k. Figure 6 shows this network when $M = 8$ and $N = 4$, along with an optimal solution. (The other optimal solution is obtained by rotating each reconfigurable link one node clockwise.)

This problem is very difficult for a genetic algorithm because of the existence of multiple completely distinct solutions (i.e. solutions that have no reconfigurable link in common). The genetic algorithm has trouble keeping the building blocks from these different solutions separate. This is generally not a problem in less contrived networks.

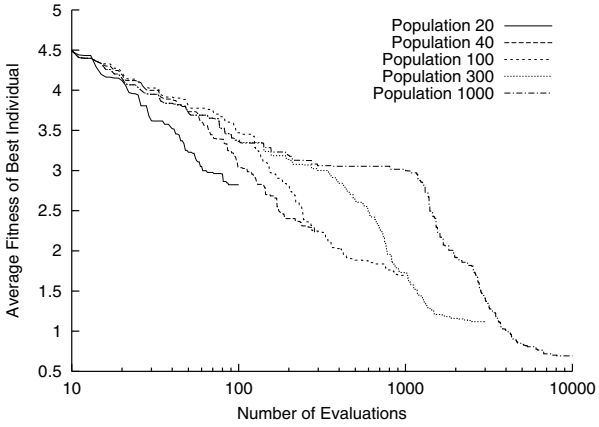


Figure 7: Average progress of the genetic algorithm for the five different parameter sets for ring network 16/8.

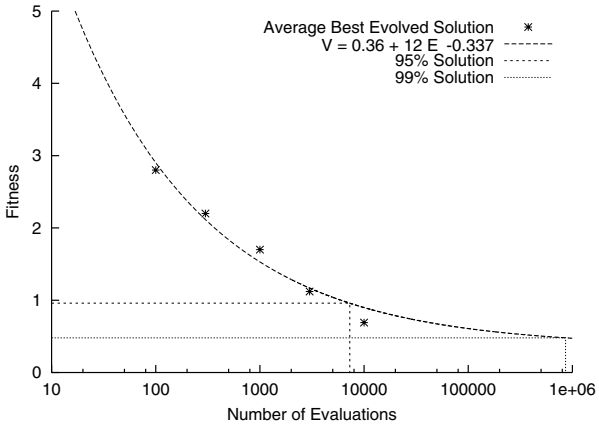


Figure 8: Modeling the tradeoff between solution quality and number of evaluations for the 16/8 ring network.

We ran experiments on ring networks with six different node/channel (i.e., M/N) configurations: 8/4, 12/6, 16/8, 20/5, 20/10, and 40/10.

For the 16/8 network and for each of the genetic algorithm parameter sets, Figure 7 shows the progress of a run (averaged over ten independent runs) plotted as the value of the best individual versus the number of evaluations (i.e., the number of configurations tried so far). Note how the smaller population with greater selection pressure starts out better but quickly stops making progress due to convergence. A larger population and smaller selection pressure requires longer to converge but eventually does better by exploring more of the space.

Network	Search Space	100 Evals	300 Evals	1000 Evals	3000 Evals	10000 Evals	A	B	C	E_{95}	E_{99}	Secs/ Eval
8/4	4.1E5	.75	.27	.20	.18	.18	.18	1100	1.65	6	16	0.69
12/6	7.3E9	1.90	1.14	.80	.48	.31	.27	22	.559	210	3800	0.93
20/5	6.6E10	5.01	3.30	1.97	1.48	1.27	.97	57	.572	190	3100	1.28
16/8	2.7E14	2.8	2.2	1.70	1.12	.69	.36	12	.337	7300	8.6E5	1.24
20/10	1.7E19	4.5	3.3	2.6	1.88	1.15	.44	17	.315	1.3E4	2.2E6	1.55
40/10	2.4E25	13.2	11.0	7.5	5.6	3.8	1.9	54	.332	8300	1.1E6	3.20

Table 1: Results for the ring networks

Table 1 provides a summary of the results for the different ring networks. Each row contains the results from one network. Column 1 contains the network name, and column 2 has the search space size as given by Equation 1. Columns 3-7 contain for each of the five parameter sets the value of the best individual at the end of a run averaged over the ten runs. The values in columns 3-7 provide five data points for the map between the number of evaluations performed and the quality of the solution.

As more evaluations are performed, the expected value of the best solution asymptotically approaches the optimal value. This leads us to a model for this relationship of the form

$$V = A + BE^{-C} \tag{2}$$

where V is the expected value of the best individual, E is the number of evaluations, and A, B and C are constants determined by the data. The constant A is the value of the best possible solution, which is known for the ring networks. We use the five data points to do a least-squares regression analysis to find B and C. We report A, B and C for each network in columns 8-10 of Table 1. Figure 8 shows an example graph of this curve for the 16/8 ring network.

The constant C measures on average how quickly the search approaches the optimal solution. After E evaluations, the search has roughly proceeded $1 - E^{-C}$ of the way from a random solution to the best solution. To find a solution that is a fraction f of the way to the optimal solution therefore requires roughly $(1-f)^{\frac{-1}{C}}$ evaluations. In columns 11 and 12 of Table 1, we report the number of evaluations required to achieve 95% and 99% of the optimal solution, given by

$$E_{95} = 20^{\frac{1}{C}}, E_{99} = 100^{\frac{1}{C}} \tag{3}$$

Figure 8 shows these values for the 16/8 ring network.

Random Networks - We next examine optimization performance on a set of randomly generated networks. While ring networks provide a worst-case optimization problem, we also would like results for more typical networks. While we do not know the best solution for these networks a priori, we can still use a regression analysis similar to (although less accurate than) that used for the ring networks to estimate how performance varies with the number of evaluations.

Net	A	B	C	E_{95}	E_{99}	S/E
8/4	.064	.91	.697	74	740	0.69
12/6	.038	150	1.63	6	17	0.90
20/5	.201	2.8	.599	150	2200	1.43
16/8	.078	5.7	.821	38	270	1.10
20/10	.092	2.5	.523	300	6700	1.49
40/10	.30	11	.512	350	8100	3.50

Table 2: Results for sparse/light random networks

Net	A	B	C	E_{95}	E_{99}	S/E
8/4	.102	13	1.10	15	66	1.49
12/6	1.53	11	.381	2600	1.8E5	1.86
20/5	2.74	7.5	.364	3800	3.1E5	3.25
16/8	.99	8.6	.434	990	4.1E4	2.70
20/10	2.07	12	.440	910	3.5E4	3.34
40/10	6.35	21	.330	8800	1.1E6	7.04

Table 3: Results for sparse/heavy random networks

Given a specified number of (i) nodes, (ii) available channels, (iii) bidirectional fixed links, and (iv) traffic flows, our software randomly generates a network with these dimensions. The random components include: (i) the fixed topology, (ii) the fixed link capacities (1000, 2000 or 3000 kbits/sec), and (iii) the source, destination, priority (1, 10 or 100), protocol (UDP or TCP), and bandwidth (100, 400 or 1000 kbits/sec) of each flow. Required latency and dropped packets were always 0.

For the experiments, we have used families of six networks. For each family, the number of nodes (M) and satellite channels (N) are the same six pairs of values as for the ring networks, hence permitting comparisons

Net	A	B	C	E_{95}	E_{99}	S/E
8/4	.045	1.2	.520	320	7000	0.75
12/6	.024	.65	.760	52	430	1.02
20/5	.089	1.2	.640	110	1300	1.76
16/8	.049	1.6	.773	48	390	1.35
20/10	.066	1.0	.540	260	5100	1.77
40/10	.159	1.9	.382	2500	1.7E5	4.82

Table 4: Results for dense/light random networks

Net	A	B	C	E_{95}	E_{99}	S/E
8/4	.103	150	1.48	8	22	1.62
12/6	.55	120	1.16	13	53	2.19
20/5	1.04	23.6	.823	38	270	4.16
16/8	.33	4.1	.576	180	3000	3.14
20/10	0.67	3.7	.368	3400	2.7E5	4.04
40/10	1.67	42.3	.696	74	750	9.59

Table 5: Results for dense/heavy random networks

of optimization performance between networks with the same search space size. The number of fixed links is pM and number of flows is qM, where q and p are constant for a family. We have used $q = 1$ and $q = 2$, referred to as "sparse" and "dense" respectively, and $p = 1$ and $p = 4$, referred to as "light" and "heavy" respectively, leading to four families of random networks: sparse/light, sparse/heavy, dense/light, and dense/heavy. For each of these families of random networks, we have done the same experiments and analysis as for the ring networks, except that we do not know apriori the optimal solution and hence the value to use for the A term. We instead estimate the optimal solution as the best solution found in any of the ten runs for any of the genetic algorithm parameters. The results are shown in Tables 2-5.

Analysis of Results - The central question is whether the optimization algorithm will support online adaptation by producing good enough configurations fast enough. While there is no clear threshold defining good enough or fast enough, we take 95% of the optimal solution in ten minutes to be our standard.

For small networks (≤ 20 nodes and ≤ 5 channels), the optimization algorithm will support online adaptation. It will consistently find the 95% solution in under 10 minutes. Once we fix the NS restart problem with the evaluations, it will do even better, potentially reaching the 98% or 99% solution in the given time.

For mid-sized networks (≤ 40 nodes and ≤ 10 channels), the optimization algorithm will be sufficient for online adaptation only with the help of additional hardware to speed the optimization. Genetic algorithms are inherently parallelizable, with a near linear speedup as a function of the number of processors up to a large number of processors [3]. Assuming a 100-processor cluster providing a factor of 100 speedup, all of the reported networks would reach their 95% solution within ten minutes.

For larger networks, the highly superlinear (potentially exponential) scaling of the algorithm means that more hardware will not address the scaling problem. Instead, fundamental improvements to the algorithm are required.

One potential source of improvements to the genetic algorithm is to use the fact that network adaptation is a continuous process. A solution that was good a few minutes earlier is still most likely a good solution. Particularly for big networks, the current optimal configuration is likely only at most a small perturbation from the previously optimal configuration. By including the previous best configuration in the initial population of the genetic algorithm to determine the current configuration, the algorithm gets a big head start and can find a good solution in far less time [14].

Another approach to improving the genetic algorithm is to incorporate heuristics, such as some of those in [10] into the algorithm. These heuristics can be used both when generating the initial population and as part of the genetic operators, and will often improve the search by a large amount [6].

A second question is how search time varies with the network. The size of the search space is the single biggest factor influencing search difficulty. It grows very quickly with the number of nodes and channels, resulting in a rapid growth in the number of evaluations required to find a good solution. (This growth is shown in Tables 1-5 as a general increase in E_{95} and E_{99} with search space size.) However, when we examine the random networks, we see that there are some networks with large search spaces that are much easier to solve than others with smaller seach spaces (e.g., see Table 4). Also, the random networks with heavy traffic tend to be more difficult to solve than networks with the same size search space but with light traffic. This is likely because more flows mean more tradeoffs and hence more difficult decisions. However, as both the ring and random networks show, even networks with light traffic can present difficulties.

4 CONCLUSION

We have introduced an important, little-investigated problem, that of determining at any given time the op-

timal configuration of a reconfigurable data network. Finding a good configuration is a critical part of the process of adaptively reconfiguring a network online. Adaptive network reconfiguration offers the benefits of survivability in the face of major changes in network operating conditions and performance fine-tuning in response to minor changes in operating conditions.

We have developed an algorithm for solving the problem using a genetic algorithm. In its current form, it is too slow for online adaptation. However, the simple step of distributing the evaluations of the genetic algorithm across many machines would make it fast enough for small and mid-sized networks. Improvements to the core algorithms of the genetic algorithm could potentially make it fast enough for networks with large numbers of nodes and reconfigurable links.

Future work should focus on making adaptive network reconfiguration a reality rather than just a possibility through (i) speeding the optimization by tuning the algorithm and (ii) integration with actual networks.

Acknowledgments

This work was supported by DARPA contract N66001-00-C-8042 as part of the SWWIM program. Thanks to Ken Theriault and John Lowry for their guidance and for their recommending this problem.

References

[1] Abuali, A., R. Wainwright, and D. Schoenfeld: 1995, 'Determinant Factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem'. *Proc. Fifth Intl. Conf. on Genetic Algorithms*. pp. 470–477.

[2] Ash, G.: 1995, 'Dynamic network evolution, with examples from AT&T's evolving dynamic network'. *IEEE Communications Magazine* **33**(7), 26–39.

[3] Cantu-Paz, E.: 2000, *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer.

[4] Chou, H., G. Premkumar, and C.-H. Chu: 2001, 'Genetic Algorithms for Communications Network Design - An Empirical Study of the Factors that Influence Performance'. *IEEE Trans. on Evolutionary Computation* **5**(3), 236–249.

[5] Coombs, S. and L. Davis: 1987, 'Genetic algorithms and communication link speed design: constraints and operators'. *Proc. Second Intl. Conf. on Genetic Algorithms*. pp. 257–260.

[6] Davis, L.: 1991, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

[7] Dengiz, B., F. Altiparmak, and A. Smith: 1997, 'Local Search Genetic Algorithm for Optimal Design of Reliable Networks'. *IEEE Trans. on Evolutionary Computation* **1**(3), 179–188.

[8] Elbaum, R. and M. Sidi: 1996, 'Topological design of local-area networks using genetic algorithms'. *IEEE/ACM Trans. on Networking* **4**(5), 766–778.

[9] Huang, R., J. Ma, and D. Hsu: 1997, 'A genetic algorithm for optimal 3-connected telecommunication network designs'. *Proc. Third Intl. Symp. on Parallel Architectures, Algorithms and Networks*. pp. 344–350.

[10] Kershenbaum, A.: 1993, *Telecommunications Network Design Algorithms*. McGraw-Hill.

[11] Ko, K.-T., K.-S. Tang, C.-Y. Chan, K.-F. Man, and S. Kwong: 1997, 'Using genetic algorithms to design mesh networks'. *Computer* **30**(8), 56–61.

[12] Konak, A. and A. Smith: 1999, 'A hybrid genetic algorithm approach for backbone design of communication networks'. *Proc. 1999 Congress on Evolutionary Computation*. pp. 1817–1823.

[13] Kumar, A., R. Pathak, M. Gupta, and Y. Gupta: 1993, 'Genetic algorithm based approach for designing computer network topology'. *Proc. 1993 ACM Conf. on Computer Science*. pp. 358–365.

[14] Montana, D., M. Brinn, S. Moore, and G. Bidwell: 1998, 'Genetic Algorithms for Complex, Real-Time Scheduling'. *Proc. IEEE Intl. Conf. on Systems, Man, and Cybernetics*. pp. 2213–2218.

[15] NS: 2001, 'The network simulator - ns-2'. http://www.isi.edu/nsnam/ns/.

[16] Palmer, C. and A. Kershenbaum: 1995, 'An Approach to a Problem in Network Design Using Genetic Algorithms'. *Networks* **26**(3), 151–163.

[17] Pierre, S. and G. Legault: 1998, 'A genetic algorithm for designing distributed computer network topologies'. *IEEE Trans. on Systems, Man and Cybernetics, Part B* **28**(2), 249–258.

[18] Sayoud, H. and K. Takahashi: 2001, 'Designing communication network topologies using steady-state genetic algorithms'. *IEEE Communications Letters* **5**(3), 113–115.

Application of Genetic Algorithms to the Discovery of Complex Models for Simulation Studies in Human Genetics

Jason H. Moore, Lance W. Hahn, Marylyn D. Ritchie, Tricia A. Thornton, Bill C. White

Program in Human Genetics
Department of Molecular Physiology and Biophysics
519 Light Hall
Vanderbilt University
Nashville, TN 37232-0700
{moore, hahn, ritchie, thornton, bwhite}@phg.mc.vanderbilt.edu

Abstract

Simulation studies are useful in various disciplines for a number of reasons including the development and evaluation of new computational and statistical methods. This is particularly true in human genetics and genetic epidemiology where new analytical methods are needed for the detection and characterization of disease susceptibility genes whose effects are complex, nonlinear, and partially or solely dependent on the effects of other genes. Despite this need, the development of complex genetic models that can be used to simulate data is not always intuitive. In fact, only a few such models have been published. In this paper, we present a strategy for identifying complex genetic models for simulation studies that utilizes genetic algorithms. The genetic models used in this study are penetrance functions that define the probability of disease given a specific DNA sequence variation has been inherited. We demonstrate that the genetic algorithm approach routinely identifies interesting and useful penetrance functions in a human-competitve manner.

1 INTRODUCTION

One goal of human genetics is to identify genes that confer an increased risk of disease in certain individuals. The identification of disease susceptibility genes has the potential to improve human health through the development of new prevention, diagnosis, and treatment strategies. Although achieving this goal is an important public health endeavor, it is not easily accomplished for common diseases, such as essential hypertension, due to the complex multifactorial nature of the disease (Kardia, 2000; Moore and Williams, 2002). That is, in such cases, risk of disease is due to a complex interplay between multiple genes and multiple environmental factors. The identification of genes that influence risk of disease only through complex interactions with other genes (i.e. gene-gene interactions) and/or environmental factors (i.e. gene-environment interactions) remains a statistical and computational challenge (Templeton, 2000; Moore and Williams, 2002). The statistical challenge is to consider high-dimensional interactions without loss of degrees of freedom while the computational challenge lies in the size and complexity of the search space. Gene-gene interactions are examples of attribute interactions, a major challenge for data mining (Freitas, 2001).

Several new methods have been developed in an attempt to address the statistical and computational challenges of detecting and characterizing complex disease susceptibility genes. These methods can be classified as either data reduction approaches or pattern recognition approaches. Data reduction methods such as the multifactor dimensionality reduction or MDR approach (Ritchie et al., 2001) seek to reduce the dimensionality of the problem in order to facilitate exploratory data analysis and hypothesis testing. MDR reduces multiple predictor variables to a single variable, thereby reducing the dimensionality of the problem. In contrast, pattern recognition and machine learning strategies such as neural networks (Lucek et al., 1998; Saccone et al., 1999) and cellular automata (Moore and Hahn, 2002) consider the full dimensionality of the data by considering patterns of DNA sequence variations. Although these methods are promising, the power of these approaches for identifying gene-gene and gene-environment interactions has not been fully evaluated. The evaluation of power is best accomplished using simulated data.

The goal of this study was to develop a genetic algorithm (GA) strategy for discovering complex genetic models in the form of penetrance functions that can be used to simulate data for the evaluation of new statistical and computational methods. Penetrance functions define the probability of disease given a particular combination of DNA sequence variations has been inherited. Penetrance functions of interest in this study exhibit gene-gene or attribute interactions in the absence of independent main effects. We begin in Section 2 with an overview of genetic models in terms of penetrance functions. In Section 3, we describe our GA approach to discovering

complex genetic models. A summary and discussion of the results are presented in Sections 4 and 5 respectively. The conclusions are presented in Section 6. The results presented in this paper demonstrate a GA strategy is capable of routinely identifying interesting and useful genetic models in a human-competitve manner.

2 PENETRANCE FUNCTIONS AS GENETIC MODELS

Penetrance functions represent one approach to modeling the relationship between genetic variations (i.e. variation in the DNA sequence of a gene) and risk of disease. Penetrance is simply the probability of disease given a particular combination of genotypes. A single genotype is determined by one allele (i.e. a specific DNA sequence state) inherited from the mother and one allele inherited from the father. For most genetic variations, only two alleles (*A* or *a*) exist in the biological population. Therefore, because the order of the alleles is unimportant, a genotype can have one of three values: *AA*, *Aa* or *aa*. Penetrance functions define the probability of disease for all genotypes for one or more genetic variations. Once the penetrance functions are specified, genetic data can easily be simulated for people with the disease and for people without the disease. For example, the penetrance function for an autosomal recessive disease (i.e. a disease that requires two copies of the same allele) such as cystic fibrosis in which only one of the three genotypes leads to disease might look like Table 1. Here, individuals who inherit the *AA* or *Aa* genotypes have zero probability of disease while individuals who inherit the *aa* genotype are certain to have the disease. From this simple recessive Mendelian model, data can simply be simulated by giving affected individuals *aa* genotypes and unaffected individuals *AA* or *Aa* genotypes, in proportion to their defined population frequencies.

Table 1. Penetrance values for three genotypes from a gene acting under an autosomal recessive disease model.

AA	*Aa*	*aa*
0	0	1

More complex genetic models can be developed by assigning disease risk to more than one genotype from one or more genetic variations. Table 2 illustrates a penetrance function that relates two genetic variations, each with two alleles and three genotypes, to risk of disease. In this example, the alleles each have a biological population frequency of $p = q = 0.5$ with genotype frequencies of p^2 for *AA* and *BB*, $2pq$ for *Aa* and *Bb*, and q^2 for *aa* and *bb*, consistent with Hardy-Weinberg equilibrium (Hartl and Clark 1997). Thus, assuming the frequency of the *AA* genotype is 0.25, the frequency of *Aa* is 0.5, and the frequency of *aa* is 0.25, then the marginal penetrance of *BB* (i.e. the effect of just the *BB* genotype on disease risk) can be calculated as (0.25 * 0) + (0.5 * 0) + (0.25 * 1) = 0.25. This means that the probability of disease given the *BB* genotype is 0.25, regardless of the genotype at the other genetic variation. Similarly, the marginal penetrance of *Bb* can be calculated as (0.25 * 0) + (0.5 * 0.5) + (0.25 * 0) = 0.25. Note that for this model, all of the marginal penetrance values (i.e. the probability of disease given a single genotype, independent of the others) are equal, which indicates the absence of main effects (i.e. the genetic variations do not independently affect disease risk). This is true despite the table penetrance values not being equal. Here, risk of disease is greatly increased by inheriting exactly two high-risk alleles (e.g. *a* and *b* are defined as high risk). Thus, *aa*/*BB*, *Aa*/*Bb*, and *AA*/*bb* are the high-risk genotype combinations. This model was first described by Frankel and Schork (1996). What makes this model complex is the absence of a main effect for either of the genetic variations. Thus, each genetic variation only has an effect on disease risk in the context of the other genetic variation. Such gene-gene interactions are believed to play an important role in determining an individual's risk for developing common diseases (Moore and Williams, 2002; Templeton, 2000).

Table 2. Penetrance values for combinations of genotypes from two genes exhibiting interactions but not main effects.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	0	0	1	.25
Bb (.50)	0	.50	0	.25
bb (.25)	1	0	0	.25
Margin penetrance values	.25	.25	.25	

Table 3. Penetrance values for combinations of genotypes from two genes exhibiting interactions but not main effects.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	0	1	0	.50
Bb (.50)	1	0	1	.50
bb (.25)	0	1	0	.50
Margin penetrance values	.50	.50	.50	

The gene-gene interaction model described in Table 2 was developed by trial and error. That is, a human derived this model by substituting various allele frequencies and

penetrance functions until a model was found that had attribute or gene-gene interaction effects without independent main effects. This is one of only a few complex genetic models that have been described in the literature. The scarcity of complex genetic models in the literature is primarily due to the extraordinary combinatorial complexity of the problem, as has been discussed by Culverhouse et al. (2002). Effectively, there are an infinite number of possible penetrance functions that could be developed for just two genetic variations. Only some of these models would exhibit a complex relationship with disease risk. The size of the search space precludes the human-based trial and error approach as well as exhaustive computational searches without specific restrictions and assumptions about the allele frequency and penetrance function values. For example, Li and Reich (2000) enumerated every possible penetrance function using probability values restricted to zero and one. This yielded a manageable 2^9 total models. Only one of these models exhibits interaction effects in the absence of main effects (see Table 3). Culverhouse et al. (2002) have also enumerated a restricted set of models. The goal of the present study was to develop a machine intelligence approach to discovering complex genetic models in the form of penetrance functions. The next section describes the GA approach we used.

3 THE GENETIC ALGORITHM

3.1 OVERVIEW OF GENETIC ALGORITHMS

Genetic algorithms have been shown to be a very effective strategy for implementing beam searches of rugged fitness landscapes (Goldberg, 1989). Briefly, this is accomplished by generating a random population of models or solutions, evaluating their ability to solve a particular problem, selecting the best models or solutions, and generating variability in these models by exchanging model components among different models. The process of selecting models and introducing variability is iterated until an optimal model is identified or some termination criteria are satisfied. This general procedure was inspired by the problem solving abilities of evolution by natural selection in biological populations. Using similar language, GAs operate using populations of chromosomes (models) that undergo selection according to fitness, reproduction, recombination, and mutation.

3.2 DESCRIPTION OF OUR GENETIC ALGORITHM

3.2.1 SOLUTION REPRESENTATION

A solution or model consists of a set of nine penetrance values or probabilities on the interval from zero to one in increments of 0.001. Thus, the entire search space consisted of 10^{27} possible models. Each penetrance value represents the probability of disease given a particular combination of two genotypes. Each of the nine real-valued probabilities was encoded as 32 bits for a total GA chromosome length of 288 bits.

3.2.2 FITNESS FUNCTION

Fitness was determined by maximizing the variance of the table penetrance values (V_t) and minimizing the variance of the marginal penetrance values (V_m). Maximizing V_t ensures that we identify interesting patterns of genotypes while minimizing V_m ensures the size of the main effect of each genotype is small. We stopped the GA when a model satisfied both $V_t \geq 0.1$ and $V_m \leq 0.0001$. These values were selected to ensure interaction effects without main effects of each genetic variation.

3.2.3 GA PARAMETERS

Table 4 summarizes the GA parameters used in this study. We ran the GA a total of 100,000 times with each run consisting of a maximum of 10,000 generations.

Table 4. GA parameters.

Objective	Discover complex models
Fitness function	V_t - V_m
Number of runs	100,000
Stopping criteria	$V_t \geq 0.1$ and $V_m \leq 0.0001$
Population size	200
Generations	10,000
Selection	Stochastic uniform sampling
Crossover	Uniform, by variable
Crossover probability	0.60
Mutation	Gaussian
Mutation probability	0.01

3.2.4 SOFTWARE AND HARDWARE

Our GA implementation used GAlib, a C++ class library for UNIX, Windows and Mac operating systems (http://lancet.mit.edu/ga/). Coarse-grained parallelism, utilizing 10 processors to perform 10 sets of 10,000 runs, for a total of 100,000 runs, used the MPICH parallel programming library on a 110-node Beowulf-style parallel computing cluster running Linux.

4 RESULTS

The GA was run for a total of 100,000 times, and the best model was saved from each. Of the 100,000 best models discovered by the GA, there were no duplicates. Thus, each model was unique. We first wanted to know whether penetrance function models that have been previously described in the literature were discovered by

the GA. The GA-generated model illustrated in Table 5 (V_t = .154764 and V_m = .000044) is very similar to the model shown in Table 2 while the model illustrated in Table 6 (V_t = .21157 and V_m = .000082) is very similar to the model shown in Table 3. Subtle variations of the models shown in Tables 2 and 5 were discovered in 13 out of the 100,000 GA runs. Similarly, subtle variations of the models shown in Tables 3 and 6 were discovered in three out of the 100,000 GA runs. Thus, the GA routinely discovered models that have been described previously.

Table 5. GA-generated model similar to the previously described model in Table 2.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.083	.076	.964	.29
Bb (.50)	.056	.508	.085	.30
bb (.25)	.977	.098	.062	.30
Margin penetrance values	.30	.29	.31	

Table 6. GA-generated model similar to the previously described model in Table 3.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.094	.905	.097	.51
Bb (.50)	.967	.097	.937	.52
bb (.25)	.027	.990	.080	.51
Margin penetrance values	.50	.52	.52	

Table 7. A GA-generated model.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.967	.314	.137	.43
Bb (.50)	.313	.312	.742	.43
bb (.25)	.129	.779	.075	.42
Margin penetrance values	.43	.42	.44	

Our second question was whether the GA routinely generated new and interesting models. All of the models identified by the GA exhibited gene-gene interactions with minimal or no main effects. In fact, other than the class of models illustrated in Tables 5 and 6, none have been described previously in the literature. Thus, approximately 99,987 models are unique. Tables 7-10 illustrate four of the new models discovered by the GA.

Table 8. A GA-generated model.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.967	.139	.799	.51
Bb (.50)	.057	.655	.627	.50
bb (.25)	.974	.544	.019	.52
Margin penetrance values	.51	.50	.52	

Table 9. A GA-generated model.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.017	.451	.711	.42
Bb (.50)	.520	.571	.039	.41
bb (.25)	.640	.053	.949	.43
Margin penetrance values	.41	.43	.42	

Table 10. A GA-generated model.

	Table penetrance values			Margin penetrance values
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	.954	.256	.360	.44
Bb (.50)	.010	.731	.300	.45
bb (.25)	.801	.093	.808	.44
Margin penetrance values	.46	.44	.45	

The model summarized in Table 7 (V_t = .106238 and V_m = .000052) indicates that individuals with genotype

combinations of *AA/BB*, *Aa/bb*, and *aa/Bb* are at highest risk of disease while those with *AA/Bb*, *Aa/BB*, and *Aa/Bb* are at intermediate risk. The remaining individuals are at relatively low risk. This nonlinear pattern of high-risk and low-risk genotype combinations is indicative of gene-gene interactions. Risk of disease is not significantly different between single genotypes (represented by margin penetrance values), confirming an absence of main effects.

The models summarized in Table 8 (V_t = .140427 and V_m = .000091), Table 9 (V_t = .110712 and V_m = .000098), and Table 10 (V_t = .120743 and V_m = .000035) have different nonlinear combinations of genotypes associated with varying risk of disease. Again, none of the genotypes in these models is associated with disease risk independent of the other genotypes. This indicates gene-gene or attribute interaction in the absence of main effects.

5 DISCUSSION

In the present work, we focused on genetic models with just two genetic variations. However, we anticipate that genetic models incorporating more than just two genetic variations will be useful in simulation studies since most common diseases are likely to be influenced by many genes. This is evident in the study by Ritchie et al. (2001) that identified a combination of four genetic variations that is associated with risk of sporadic breast cancer in a complex nonlinear manner. Our future studies will focus on expanding the GA to search for combinations of three or more genetic variations that exhibit attribute interaction in the absence of main effects. Further, it will be important to explore a range of allele frequencies as well as methods for categorizing models into similar classes.

Human genetics is undergoing an information explosion and a comprehension implosion. In fact, our ability to measure genetic information, and biological information in general, is far outpacing our ability to interpret it. As demonstrated in this study, machine intelligence strategies such as GAs hold promise for dealing with genetic data that is high-dimensional and complex. However, the present study is not the first to apply evolutionary algorithms to a genetics problem. In fact, evolutonary algorithms have been used to optimize data analysis approaches in genetic epidemiology studies (Congdon et al., 1993; Carlborg et al., 2000; Tapadar et al., 2000; Moore and Hahn, 2002), gene expression studies (Moore and Parker, 2001; Moore et al., 2001; Parker and Moore, 2001), and studies of gene networks (Koza et al., 2001). We anticipate an increase in applications of GAs in the field of human genetics as more investigations begin to focus on the challenge of simulating and analyzing complex, high-dimensional genetic data.

6 CONCLUSIONS

The results of this study document the utility of GAs for the discovery of complex genetic models that can be used for simulation studies in human genetics. In fact, our GA discovered approximately 99,987 models that have not been previously described in the literature. Thus, the results are human-competitive and routine. To our knowledge, this is the first application of a machine intelligence approach to the discovery of complex genetic models such as penetrance functions.

Acknowledgments

This work was funded by National Institutes of Health grants HL65234, HL65962, GM31304, AG19085, and AG20135.

References

Carlborg O, Andersson L, Kinghorn B (2000): The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. Genetics 155:2003-10.

Congdon CB, Sing CF, Reilly SL (1993): Genetic algorithms for identifying combinations of genes and other risk factors associated with coronary artery disease. In: Proceedings of the Workshop on Artificial Intelligence and the Genome. Chambery.

Culverhouse R, Suarez BK, Lin J, Reich T (2002): A perspective on epistasis: limits of models displaying no main effect. American Journal of Human Genetics 70:461-71.

Frankel WN, Schork NJ (1996): Who's afraid of epistasis? Nature Genetics 14:371-373.

Freitas AA (2001): Understanding the crucial role of attribute interaction in data mining. Artificial Intelligence Reviews 16:177-199.

Goldberg DE (1989): "Genetic Algorithms in Search, Optimization, and Machine Learning." Reading: Addison-Wesley.

Hartl DL, Clark AG (1997): "Principles of Population Genetics" Mass: Sinauer Assoc.

Kardia SLR (2000): Context-dependent genetic effects in hypertension. Current Hypertension Reports 2:32-38.

Koza JR, Mydlowec W, Lanza G, Yu J, Keane MA (2001): Reverse engineering of metabolic pathways from observed data using genetic programming. Pacific Symposium on Biocomputing 2001:434-445.

Li W, Reich J (2000): A complete enumeration and classification of two-locus disease models. Human Heredity 50:334-349.

Lucek P, Hanke J, Reich J, Solla SA, Ott J (1998): Multi-locus nonparametric linkage analysis of complex trait loci with neural networks. Human Heredity 48:275-284.

Moore JH, Hahn LW (2002): A cellular automata approach to detecting interactions among single-nucleotide polymorphisms in complex multifactorial diseases. Pacific Symposium on Biocomputing 2002:53-64.

Moore JH, Parker JS, Hahn LW (2001): Symbolic discriminant analysis for mining gene expression patterns. In De Raedt, L. and Flach, P. (eds), " Machine Learning: ECML 2001" Lecture Notes in Artificial Intelligence 2167:372-381, Berlin: Springer-Verlag.

Moore JH, Parker JS (2001): Evolutionary computation in microarray data analysis. In Lin, S. and Johnson, K. (eds), " Methods of Microarray Data Analysis" Boston: Kluwer Academic Publishers.

Moore JH, Williams SW (2002): New strategies for identifying gene-gene interactions in hypertension. Annals of Medicine 34:1-8.

Parker JS, Moore JH (2001): Dynamics based pattern recognition and parallel genetic algorithms for the analysis of multivariate gene expression data. Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco, pp 433-436.

Ritchie MD, Hahn LW, Roodi N, Bailey LR, Dupont WD, Parl FF, Moore JH (2001): Multifactor dimensionality reduction reveals high-order interactions among estrogen metabolism genes in sporadic breast cancer. American Journal of Human Genetics 69:138-147.

Saccone NL, Downey Jr. TJ, Meyer DJ, Neuman RJ, Rice JP (1999): Mapping genotype to phenotype for linkage analysis. Genetic Epidemiology. S17: S703-8.

Tapadar P, Ghosh S, Majumder PP (2000): Haplotyping in pedigrees via a genetic algorithm. Human Heredity 50:43-56

Templeton AR (2000): Epistasis and complex traits. In: Wade M, Brodie III B, Wolf J (eds) " Epistasis and Evolutionary Process" Oxford: Oxford University Press.

Multi Objective Airfoil Design using Single Parent Populations

Boris Naujoks
Computer Science Dept. XI
University of Dortmund
D-44221 Dortmund

Werner Haase
EADS Military Aircraft
Dept. MT63 / Build. 70.N
D-81663 Munich

Jörg Ziegenhirt
NuTech Solutions GmbH
Martin-Schmeißer-Weg 15
D-44227 Dortmund

Thomas Bäck
NuTech Solutions GmbH
Martin-Schmeißer-Weg 15
D-44227 Dortmund

Abstract

A new approach for multi criteria design optimization is presented in the paper. The problem tackled with this approach is the 2-dimensional design of an aircraft wing. To carry the derandomized step size control operator for evolution strategies also to multi criteria applications, three different selection schemes are proposed. Two of them fail to obtain a certain quality of outcome, but the third one leads to promising results. This third selection scheme more emphasizes the diversity among individuals than the other selection schemes.

1 INTRODUCTION

Aviation in general is not only one of the most important fields in industry, but also in science. Due to the many potential savings that are possible in this area many researchers from scientific organizations as well as from industry work here on production cost minimization, flight behavior improvement, etc. This results in a considerable impact from aviation on science itself and makes it that important.

One of the most referenced applications from aviation is aircraft wing design. Due to the development of computational fluid dynamic (CFD) methods, wing design is nowadays mostly done using computers, which provide a scalable preciseness for different design tasks. Nevertheless computers are nowadays not able to offer the computational power to calculate all needful properties of a whole aircraft with the mandatory precision in a reasonable amount of time. Therefore, the aircraft is divided into logical parts that are often designed independently. After receiving good and reasonable results on the several parts, these parts are put together again.

The insufficient computational power of computers is also the reason for working with two-dimensional problems to get basic results, before changing to the three-dimensional applications.

Different methods from optimization have been carried out on the current design problem, which is presented in detail in section 2. On a simplified problem considering only one flow condition resulting a one dimensional optimization problem, evolutionary algorithms (EA) [1] have been applied very successfully. Especially the derandomized step size control mechanism for evolution strategies (ES) [2] yielded the best results here. This special mutation operator is presented in section 3.1.

To carry this operator to multi objective tasks and, moreover, use step size adaptation in multi objective applications in general, three approaches are tackled in section 4.1.

First results can be found in section 5, where also first conclusions are drawn. One of the selection schemes is recognized to be a very promising approach, that should be further investigated on other test cases.

2 AIRFOIL DESIGN TEST CASE

In the current investigation a two-dimensional airfoil design problem for viscous flow is considered. The problem described is on of the test cases from the European research project AEROSHAPE. Here, all modeling issues concerning CFD, e.g. mesh size, mesh generation, used models, pressure calculation, etc. have been fixed and two regimes of flow conditions have been chosen. These regimes vary in the flow parameter settings and a suitable airfoil as a compromise for both conditions is to be designed.

In contrast to the airfoil design problem using only

one regime of flow conditions (single point), the current task requires the application of multi criteria decision making methods. Therefore, genetic algorithms as well as evolution strategies have been used in conjunction with Pareto optimization techniques. In contrast to results already presented on the current test case [3, 4], the parameterization of the airfoil using Bezier points for determining the design has been improved. Here, some x-components of these points have been involved in the optimization process in addition to the y-components of all points.

The software and technical support for the fitness function calculation was provided by the European Aeronautic Defence and Space Company – Military Aircraft (EADS-M), one of the partners in the AEROSHAPE project. The two flow conditions are calculated using different models, namely the Johnson-King model for the subsonic flow (high-lift test case) and the Johnson-Coakley model for the transonic flow (low-drag test case).

The parameter settings describing the flow conditions in use are given in table 1.

Property	Case	high lift	low drag
M_∞	[−]	0.20	0.77
Re_c	[−]	$5 \cdot 10^6$	10^7
$X_{transition}$ (upper / lower)	[c]	3% / 3%	3% / 3%
α	[^{0}]	10.8	1.0

Table 1: Summarized design conditions (c=chord length)

For the fitness function calculation two target airfoil designs are given, one for each flow condition. The fitness function reads as follows:

$$F(\alpha_1, \alpha_2, x(s), y(s)) = \sum_{n=1}^{2} \left[W_n \int_0^1 (C_p(s) - C^n_{p,target}(s))^2 ds \right]$$

with s being the airfoil arc-length measured around the airfoil and W_n weighting factors. C_p is the pressure coefficient distribution of the current and $C^n_{p,target}$ the pressure coefficient distribution of the target airfoils, respectively.

Due to the large calculation times for the Navier-Stokes simulations, only the restricted number of 1000 fitness function evaluations, being already a lot, is allowed.

For first investigations, only one flow condition has been studied. This leads to a *normal* single dimensional test case and a reduced calculation time of about half of the calculation time for both flow conditions.

3 EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms are nowadays a widely spread stochastic optimization method. They have proven their practicability and efficiency in many optimization tasks. Furthermore, EAs are build on theoretical fundamentals today, reemphasizing their appropriateness. Nevertheless detailed knowledge about parameterizing these methods accordingly is essential.

This class of algorithms is subdivided into different methods according to the representation, genetic operators and selection methods used. In the present application genetic algorithms (GA) [5] have been tested next to evolution strategies. The genetic algorithm stems from a commercial design optimization toolbox called FRONTIER[1]. The algorithms included in FRONTIER are parameterized to solve design problems in general. The user is able to change basic strategy parameters in the graphical user interface of the FRONTIER tool, to receive different performances for the problem under investigation. For GAs these parameters are

- population size,
- mutation probability, and
- recombination type

among others, which are not that influential.

The best result for the single-objective low-drag test case has been a value of $1.44 \cdot 10^{-3}$. This result was computed using the new airfoil parameterization mentioned above (compare 2) and could be improved using evolution strategies. Best results could be obtained with ES using the derandomized step size control mechanism from Ostermeier et al. [6].

3.1 DERANDOMIZED STEP SIZE CONTROL

The mechanism of derandomized step size control in evolution strategies (DES) has been proven to be very successful, particularly in industrial applications with

[1]FRONTIER, the Open System for Collaborative Design Optimization using Pareto Frontiers, is a product of ES.TEC.O, a branch of ENGIN SOFT, Tecnologie per l'Ottimizzazione, AREA Science Park, Padriciano 99, 34012 Trieste, Italy (www.enginsoft.it/frontier).

only a restricted number of fitness function evaluations.

In contrast to the standard step size adaptation technique from ES, the derandomized mutational step size control accumulates information about the selected individual's mutation vector $\vec{z}$ over the course of evolution by adding up the successful mutations. The authors claim that the method enables a reliable adaptation of individual step sizes (i.e., n different standard deviations σ_i) even in small populations, namely, in (1,λ)-strategies with $\lambda = 10$ in the experiments reported. The proposed method utilizes a vector $\vec{z}^{\,g}$ of accumulated mutations as well as individual step sizes σ_i and a global step size σ according to [6]:

$$\vec{z}^{\,g} = (1-c)\vec{z}^{\,g-1} + c\vec{z}^{\,*}\ ,\ \vec{z}^{\,0} = \vec{0} \quad (1)$$

$$\sigma' = \sigma \cdot \left(\exp\left(\frac{|\vec{z}^{\,g}|}{\sqrt{n}\sqrt{\frac{c}{2-c}}} - 1 + \frac{1}{5n} \right) \right)^{\beta} \quad (2)$$

$$\sigma_i' = \sigma_i \cdot \left(\frac{|z_i^g|}{\sqrt{\frac{c}{2-c}}} + 0.35 \right)^{\beta'} \quad (3)$$

$$x_i' = x_i^* + \sigma' \cdot \sigma_i' \cdot N_i(0,1) \quad (4)$$

Essentially, equation (1) captures the history of successful mutations by a weighted sum of the mutations selected in preceding generations (i.e., $\vec{z}^{\,g-1}$) and the mutation vector $\vec{z}^{\,*}$ of the selected parent individual (notice that the method applies to (1,λ)-strategies, i.e., $\vec{z}^{\,*}$ is the mutation vector of the single best offspring individual produced in generation $g-1$). The vector $\vec{z}^{\,g}$ is then used to update both a global step size σ and individual step sizes σ_i according to equations (2) and (3).

Equation (4) then denotes the generation of offspring individuals from the single parent (with components x_i^*) in a way similar to the standard ES mutation mechanism using σ' and σ_i'. Concerning the choice of the new learning rates c, β, and β', both theoretical and empirical arguments are given in [6] for the settings $c = 1/\sqrt{n}$, $\beta = 1/\sqrt{n}$, $\beta' = 1/n$.

Figure 1 shows 4 runs of $(1+10)$-ES using this kind of step size control after the change of the airfoil parameterization already mentioned in section 2 (compare [3, 4]). The best solution so far could be improved in all 4 runs. This shows, that the derandomized step size control mechanism is a very successful method for the given application. Therefore, the usage of this method should also be continued or at least tried on the multi objective test case.

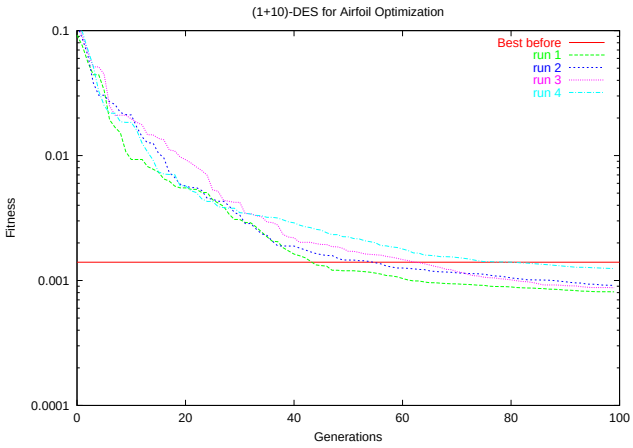


Figure 1: 4 runs of a $(1+\lambda)$-DES in comparison to the best result so far (horizontal line)

4 MULTI OBJECTIVE EVOLUTIONARY ALGORITHMS

Due to their population-based approach, EAs are very promising methods if more than one value to describe the quality of an individual is required. The first overview on multi criteria decision making using EAs was given by Fonseca and Fleming in 1995 [7]. In the following text, their definitions concerning dominance, Pareto-optimality, Pareto-sets, and -fronts are used. The term *Pareto-front* is additionally defined as the set of non-dominated individuals.

Next to the definition of Pareto-dominance etc. an archive to store the individuals, e.g. non-dominated ones, over a number of generations plays another important role in Multi Objective Evolutionary Algorithms (MOEAs). This archive gives the algorithm the chance to compare current solutions to older non-dominated ones and select these older individuals, if the chosen selection mechanism does not find a better one in the current population. Roughly, this archive implements a kind of plus strategy from evolution strategies nomenclature, taking not only the current offspring population into account for the selection step, but also individuals from preceding generations. In the case of using archives, not only parents are taken into account, but all individuals from the whole history of the current evolution process qualified for the incorporation in the archive, e.g. by being non-dominated.

Problems that arise from using archives are the number of individuals stored in this archive. Here, new methods for selecting individuals to be stored and also to be deleted from the archive are in discussion [8]. Another problem arises when using step size adapta-

tion. Falling back to older individuals from the archive could lead to replacing good step sizes in the current individuals and moreover to forget useful information gained from the evolution process. This is the main reason, why there normally is no step size adaptation in MOEAs.

Our approach to deal with the airfoil optimization problem therefore does not use an archive. No information, except for the parent individual is carried on into the next generation.

4.1 SINGLE PARENT MULTI OBJECTIVE DES

Since the single objective version of the DES approach has been successfully applied to the single point airfoil design problem, a transformation using this approach for multi-objective design seemed quite natural. Therefore, a lot of different subjectives have to be taken into account, especially:

- One single parent has to be chosen from a population respecting all different and sometimes conflicting fitness function values.
- The strategy must not focus in approaching one single global optimum but the whole Pareto-front of non-dominated individuals.

Selecting one individual from a set of individuals each having more than one fitness function value is in the end only a special case of selecting a new population in MOEAs. The first step in the selection scheme is almost clear, if the Pareto concepts are applied. If there is one and only one non-dominated individual, select this one for becoming the parent of the next generation.

If there is not only one non-dominated individual in the population, there must be more than one non-dominated individuals. Due to some logical considerations the case that there is no non-dominated individual is not possible.

For the case with more than one non-dominated individuals, further approaches are possible. One possibility, which was tried in the current investigations, is to choose the individual, which dominates most of the other ones. Another selection scheme would be to select the individual next to the origin of the fitness function space, the global optimum for both fitness function values in this special redesign test case. In fact this global optimum remains unreachable if different target airfoils are considered. Another scheme tested is the selection of the individual with the greatest distance to the other individuals in the fitness function space. Again, different strategies can be applied here: comparing the fitness to all other individuals or to the individuals on the same level, e.g. all non-dominated individuals, all individuals dominating the same number of other individuals and so on.

Up to now all the formulations and specifications for the new method of multi objective DES hold for two cases: A comma strategy as well as the elitist plus strategy. Due to hard restrictions concerning the allowed number of fitness function evaluations, the elitist (1+10)-DES is used. Combined with the choice of this selection scheme is the hope, that it performs better than the comma strategy, because only improvements with respect to the selection mechanism in use are possible.

One drawback of the applied techniques is that the sum of different fitness function values can become worse in following generations. This might happen, if one non-dominated individual is selected due to other selection schemes, e.g. the distance to the other non-dominated individuals.

In the current investigation, the following selection schemes have been compared:

Scheme 1: If more than one non-dominated individual is in the population, the number of individuals dominated is taken into account. If there is one individual with a maximum number of dominated individuals, this one is chosen to become the parent of the next generation. If there are more than one with the same maximum number of individuals dominated, the distance to the origin of the fitness function space is taken into account. In this special case, the individual with the smallest distance to the origin, which is the global optimum due to fitness function formulation, is selected to become the parent of the next generation.

Scheme 2: This selection scheme is similar to Scheme 1 presented above, but instead of the distance to the origin, the distance to other individuals from the population has been taken into account. More precisely, the individuals with the same number of dominated individuals are compared to each other. Therefore the distance of one individual to the other ones is calculated and added. The one with the greatest sum, thus the one with the greatest sum of distances to the other individuals, is selected and becomes the parent of the next generation.

Scheme 3: In the third selection scheme investigated here, the second criteria from scheme 2, the number of individuals dominated, is omitted. If more than one non-dominated individual is in the population, the one with the greatest distance among all non-dominated solutions is selected to become the parent of the next generation. Therefore, the distance of each non-dominated individual to the other ones is calculated and added similarly as in selection scheme 2.

5 RESULTS

For each of the presented selection schemes, different optimization runs have been performed. Each of these runs can be compared to one another by the results achieved. Most important for the comparison of results is the obtained Pareto-front.

Prior to these ES results, figure 2 presents the geometry obtained with the multi objective GA in the FRONTIER tool. The thick solid line denotes the best engineering geometry. The symbols present the Bezier control points by which this airfoil geometry is achieved. It can be well recognized, that the resulting geometry is a compromise between the two target (subsonic and transonic) geometries (narrow lines shown in figure 2).

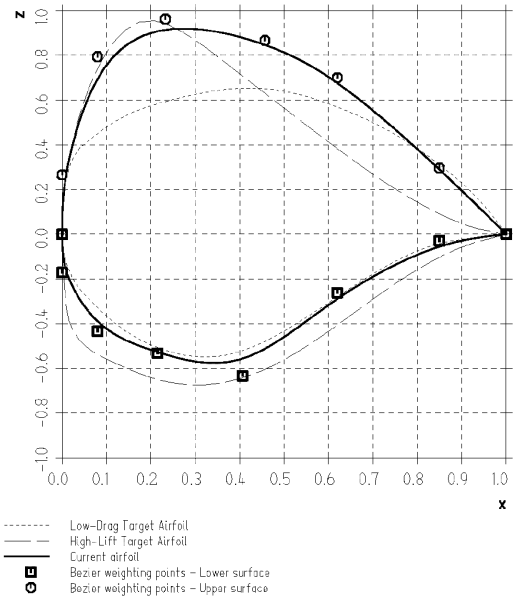


Figure 2: Airfoil geometry (thick) and two target airfoils (thin)

Additionally, figure 3 exhibits the pressure distribution obtained for the best airfoil in subsonic and transonic flow. Again, the results indicate a compromise between both targets and, furthermore, that the optimized shape tends to come closer to the subsonic shape in the front part of the airfoil and follows better the transonic shape in the rear part.

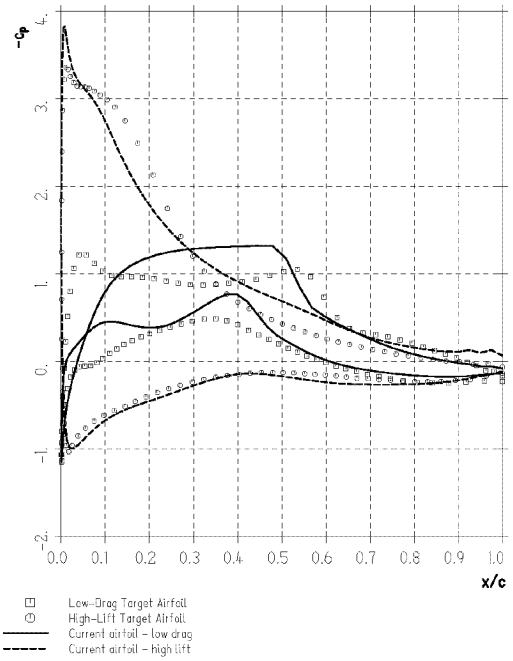


Figure 3: Pressure distributions on upper an lower surface from the airfoil design in figure 2 (lines) and two target airfoils (dots)

In addition to the Pareto-front, conclusions can be drawn from the typical fitness over generations plots. Different curves can be investigated like the fitness of each objective against the generation number or the sum of both objectives against the generation number. But this will not be done here. Instead, the path of the parent individual in the fitness function space is observed as another method for comparison.

5.1 DIFFERENT SELECTION SCHEMES

Comparing the different selection schemes presented in this paper, the Pareto-fronts show the most obvious and remarkable results. A typical Pareto-front obtained by selection scheme 1 is presented in figure 4. Here, as well as in the following figures of the same type all 1000 individuals of one optimization run are presented and the Pareto-front members are marked more dark. The search focuses on the path to the origin of the fitness function space. This behavior can also be observed in the way of the parent individual in the fitness function space drawn in figure 5. It is

expected that this behavior originates from the third selection step, taking this distance to the origin of the fitness function space into consideration.

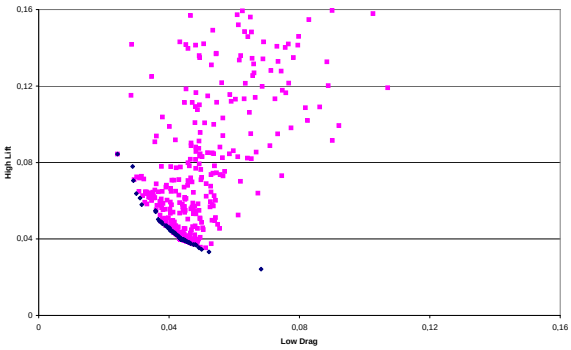


Figure 4: Pareto-front, selection scheme 1

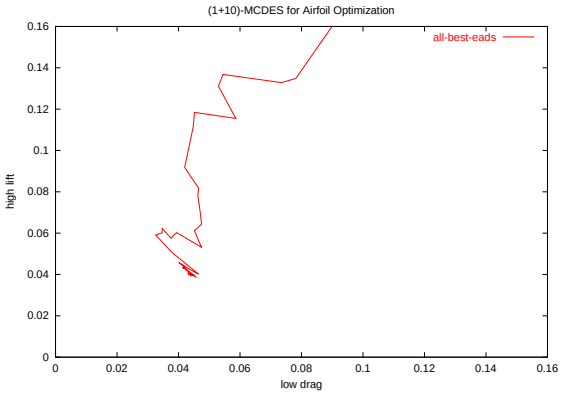


Figure 5: Path of parent individual, selection scheme 1

Surprisingly, the behavior does not change significantly, if the third selection step is changed. In figure 6 the Pareto-front obtained by a typical run using selection scheme 2 is shown. The same behavior like before, the focus of the search towards the origin of the fitness function space, can be observed. This behavior changes not until the second selection step, selection considering the number of dominated solutions, is omitted from the selection procedure.

5.2 SELECTION SCHEME 3

Figure 7 presents a typical Pareto-front from a run using selection scheme 3. The Pareto-front is covered satisfactorily, showing a wide range of different alternative solutions. This gives the user the possibility to compare many alternative solutions featuring different

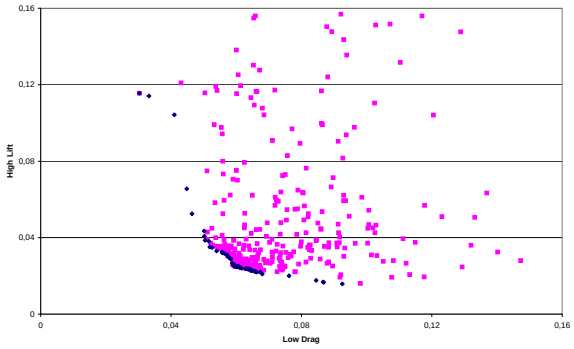


Figure 6: Pareto-front, selection scheme 2

design aspects.

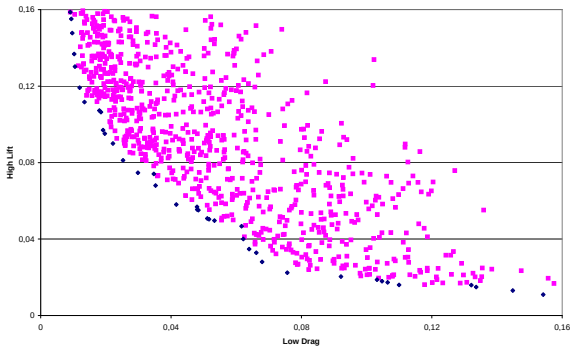


Figure 7: Pareto-front, selection scheme 3

Furthermore, even the extreme specifications of the fitness function space are explored, where the solutions focus on only one objective. This behavior may be of special interest, if one objective plays an accentuated role among the others.

It should be emphasized again, that all results have been obtained within the small number of 1000 fitness function evaluations. This is small in comparison to other MOEAs on similar problems (compare to Obayashi [9] or Quagliarella et al. [10]) and especially compared to authors working on theoretical test function, usually using more than 10000 fitness function evaluations [11, 12].

In the following, the Pareto-front from figure 7 is compared to another one obtained using selection scheme 3. This one is presented in figure 8. Here, a focus towards the low-drag area of the fitness function space

can be observed next to yielding a similar distributed Pareto-front. In contrast to this run focusing more on the low-drag area, an emphasis on the high-lift area can be detected in figure 7.

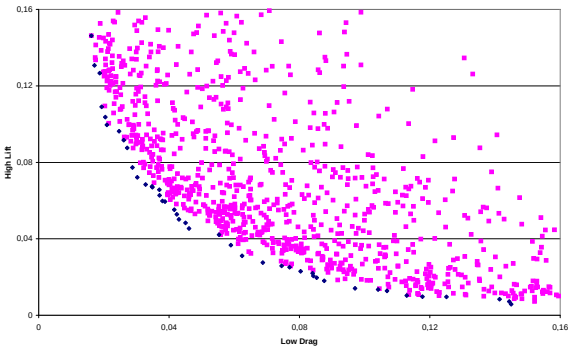


Figure 8: Pareto-front, selection scheme 3, further run

For further investigations, the path of the parent individual in the fitness function space is considered. The path of the parent individual from the run presented in figure 7 using selection scheme 3 is presented in figure 9. It can be directly compared to the path of the parent individual obtained using selection scheme 1 from figure 5. It can be observed, that the path of the parent individual in figure 9 is much more distributed in the fitness function space. In conjunction with the selection pressure this distribution is moved towards the Pareto-front. The result is, that a much larger part of the Pareto-front is covered here.

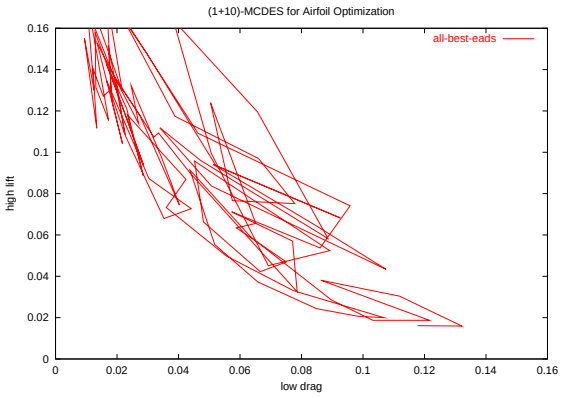


Figure 9: Path of parent individual, selection scheme 3

In figure 10 the path of the parent individual from the run presented in figure 8 is shown. The distribution of the Pareto-front is as wide as in figure 9 sharing the same selection scheme. This shows the reliability of the method used. Otherwise, some differences between the runs can be observed again. This kind of plots emphasizes the preferred search direction. While figure 9 focuses much more on the high-lift quality of the airfoil, figure 10 more tends towards the low-drag part. It can be observed, that this preferred search directions are more driven towards the absolute minimum, the target airfoil respectively. This results in a shorter distance to the corresponding axes. Based on this results, it can be assumed, that more fitness function evaluations would lead to better results, driving both qualities of the airfoil to much better solutions.

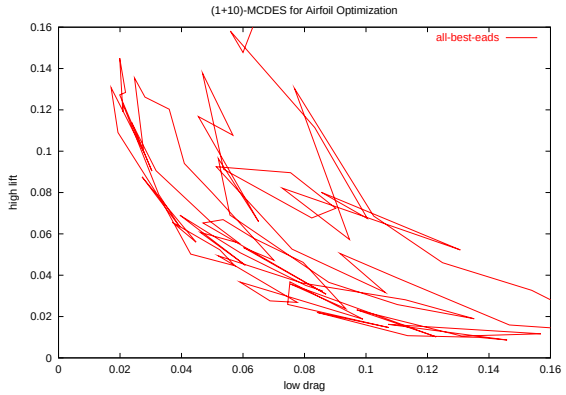


Figure 10: Path of parent individual, selection scheme 3, further run

6 CONCLUSIONS

Three different selections schemes have been presented, each able to select exactly one individual from a set of candidates using a multi objective fitness function. This was necessary for carrying the derandomized step size control mechanism from single point airfoil design, where it was applied very successfully, to multi point the design case.

Two of the presented selection schemes lead to insufficient results, not yielding Pareto-fronts of an expected quality, e.g. distribution over the fitness function space. This was due to considering the number of individuals dominated for the selection scheme and therefore focusing too much on Pareto techniques instead of the diversity of individuals.

Much better results could be obtained by focusing more on the distribution of individuals in the selection scheme, omitting the part considering dominated individuals in between. Using the new selection scheme 3 different results of comparable qualities can be obtained focusing on different regions of the fitness func-

tion space. The high quality Pareto-fronts have been computed within the very small number of 1000 fitness function evaluations and much better solutions, focusing on more regions of the fitness function space can be assumed from more fitness function evaluations.

Acknowledgments

This research was supported by the Deutsche Forschungsgemeinschaft as part of the collaborative research center "Computational Intelligence" (531).

The AEROSHAPE project (Multi-Point Aerodynamic Shape Optimisation) is a collaboration between Aerospatiale Matra Airbus, Alenia Aeronautica (Coordinator), Daimler Chrysler Airbus, EADS-M, Dassault Aviation, SAAB, SENER, SYNAPS, CIRA, DERA, DLR, FFA, INRIA, HCSA, NLR, ONERA, and NuTech Solutions. The project is funded by the European Commission, DG Research, under the GROWTH initiative (Project Ref: GRD1-1999-10752).

References

[1] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[2] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[3] Boris Naujoks, Lars Willmes, Werner Haase, Thomas Bäck, and Martin Schütz. Multipoint airfoil optimization using evolution strategies. In *Proc. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS'00) (CD-Rom and Book of Abstracts)*, page 948 (Book of Abstracts), Barcelona, Spanien, September 11–14, 2000 2000. Center for Numerical Methods in Engineering (CIMNE).

[4] Thomas Bäck, Werner Haase, Boris Naujoks, Luca Onesti, and Alessio Turchet. Evolutionary algorithms applied to academic and industrial test cases. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 383–397. Wiley, Chichester, 1999.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[6] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. Step-size adaptation based on non-local use of selection information. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature — PPSN III International Conference on Evolutionary Computation*, volume 866 of *Lecture Notes in Computer Science*, pages 189–198. Springer, Berlin, 1994.

[7] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[8] David W. Corne, Joshua D. Knowles, and Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[9] Shigeru Obayashi, Takanori Tsukahara, and Takashi Nakamura. Multiobjective Evolutionary Computation for Supersonic Wing-Shape Optimization. *IEEE Transactions on Evolutionary Computation*, 4(2):182–187, July 2000.

[10] Domenico Quagliarella and Alessandro Vicini. Designing High-Lift Airfoils Using Genetic Algorithms. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Proceedings of EUROGEN'99*, Jyväskyl, Finland, 1999. University of Jyváskylä.

[11] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[12] Carlos A. Coello Coello. An Updated Survey of Evolutionary Multiobjective Optimization Techniques : State of the Art and Future Trends. In *1999 Congress on Evolutionary Computation*, pages 3–13, Washington, D.C., July 1999. IEEE Service Center.

Multi-Objective Optimisation of Rolling Rod Product Design using Meta-Modelling Approach

V. Oduguwa and R. Roy

Department of Enterprise Integration, School of Industrial and Manufacturing Science,
Cranfield University, Cranfield, Bedford, MK43 0AL, UK
Email: {v.oduguwa and r.roy}@cranfield.ac.uk
Tel: +44 (0) 1234 754072, Fax: +44 (0) 1234 750852

Abstract

Traditional solution methods such as search and sort for optimising complex real life engineering problems can be very expensive in terms of computational time. The considerable execution time tends to inhibit elaborate exploration of the design space and often results to sub-optimal solutions. This paper reports on an engineering optimisation approach designed to bridge the gap between traditional solution methods in the industry and state-of-the-art techniques from the research community. A modelling and optimisation technique has been developed using Design of Experiment (DoE) and meta-modelling approach to approximate expensive finite element (FE) runs. An evolutionary computational technique (NSGAII) is used for solving the optimisation problem. This solution technique was applied for multi-objective optimisation of a rod rolling design problem. The results showed NSGAII converge to the Pareto optimal front. The multiple optimal solutions help the designer in delivering a variety of optimal designs.

1 INTRODUCTION

Finite element analysis (FEA) and genetic algorithm (GA) often used as an integrated optimisation process paradigm is an increasingly important component of engineering research and product development. Finite element solver is used as the fitness function within GA in order to exploit GA's global searching capability and the modelling strength of the FE solvers. Since GA requires a large number of function evaluations, it follows that large number FE runs are also required. This can be computationally expensive for solving complex engineering problems.

In rod rolling design optimisation problems, conventional methods such as search and sort are often used to solve complex optimisation problems. This approach relies on the use of the analyst's qualitative knowledge to explore the design space (Roy, 1997; Oduguwa and Roy, 2001). Expensive FE analyses are often invoked repeatedly during the process making multi-objective optimisation and concept exploration time consuming. This search method can inhibit elaborate exploration of the design space and often results to sub-optimal solutions. The use of evolutionary multi-objective optimisation techniques for improving the search for this class of real life engineering problems is proposed in this paper. Even though this approach can be an improvement from the conventional method, literature reveals that integrating FE and GA incurs quite an expensive computational cost.

Cerrolaza and Annicchiarico, (1999) solved a bi-dimensional shape optimisation problem using GA and FEA as the fitness function. In the test results presented in their paper, the optimisation process stopped after 5000 FE evaluations and took about 150 minutes. If the same number of evaluations were used in rod rolling optimisation problem (such as the case presented in this paper, where one FE run last about 17 minutes) the process would be completed after 52 days. Clearly this time scale is not acceptable for engineering applications.

Statistical meta-modelling approach is proposed to address expensive FE runs in the context of multi-objective optimisation for rod rolling problems. Statistical techniques are becoming widely used in engineering design to construct approximations of meta-models- 'a model of a model' of these analysis codes; these serve as surrogate models of the analysis codes (Myers and Montgomery, 1995; Kleijnen and Sargent, 2000). An evolutionary multi-objective optimisation technique is also proposed as above, for improving the search for this class of real life engineering problem.

This paper reports on the application of design of experiment (DoE) to create meta-models for FE models and evolutionary computational techniques (NSGAII) for the multi-objective optimisation of a rod rolling design problem.

The remainder of the paper is organised as follows. Section 2 states the formal definition on multi-objective optimisation. Section 3 reviews the literature on approaches to address the computational cost of FE runs and also the recent multi-objective techniques. Section 4

presents the rod rolling design problem. Section 5 covers the meta-modelling approach consisting of 6 main steps. Section 6 and 7 presents the application of the meta-modelling approach to the rod rolling design problem. Section 8 contains future research activities and finally, section 9 concludes.

2 MULTI-OBJECTIVE OPTIMISATION

Most real world problems are characterised by several non-commensurable, conflicting objectives. Multi-objective optimisation seeks to minimise the n components $f(x) = (f_1(x), ..., f_n(x))$, of a possibly non-linear vector function f of a decision variable x in the search space. Each of these objectives has a different optimal solution. There is no unique, (Utopian) solution to a multi-objective problem but a set of non-dominated solutions referred to as Pareto-optimal set. A solution to this class of problem is Pareto-optimal if from a point in the design space, the value of any other solution cannot be improved without deteriorating at least one of the others. The objective for a complex multi-objective optimisation problem is to find different solutions close and well distributed on the true Pareto-optimal front. The conditions for a solution to become dominated with respect to another solution are described as follows.

For a problem having more than one objective function (say, f_j, where $j = 1, ..., M$ *and* $M > 1$), A solution $x^{(1)}$ is said to dominate solution $x^{(2)}$ if the following conditions are satisfied:

a) The solution $f_j(x^{(1)})$ is no worse than $f_j(x^{(2)})$ for all $j = 1, 2, ..., M$ objectives.

b) The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective.

3 LITERATURE REVIEW

In this section, related research in optimisation for solving real life problems is reviewed, with focus on the solution approaches to address the expensive computational cost of large FE runs. Recent solution techniques on multi-objective optimisation are also reviewed.

3.1 FEA AND GA COMPUTATIONAL COST

There are several approaches proposed to address computational cost of large FE runs. Deb and Gulati, (2001) in their work on design of truss-structures, introduced the concept of basic and non-basic node to emphasise creation of user-satisfactory trusses and reduce computational time by avoiding expensive FEA for unsatisfactory trusses. Quagliarella and Vicini, (2001) proposed a hierarchical approach for the fitness evaluation. This involves using several solvers with different levels accuracy, in order to use the more computationally expensive models only when needed. These approaches can be regarded as "good house keeping measures" that improves on the computational expense of large FE runs, however they fall short of alleviating the problem in the context that makes them applicable to complex real life problems. A second classification is the solution approximation approach. This occurs when numerical solution of the FE solver is approximated, using different techniques. Chen and Lin (2000) in optimisation of design space topology used artificial neural network as an approximation to replace the structural analyses of the FE. Although this gives quick results, the approach still requires substantial data to train and validate the neural network Chen, (2001) applied design of experiment to approximate FE analysis and created a response surface for single objective optimisation of impact structure and crashworthiness problem. The author used classical full factorial experimental designs. This is considered expensive. Sacks et al, (1989) argued that since deterministic computer experiment lacks random error, classical experimental designs are not suitable for sampling them. This implies that computer experiments can be run with less sample points. Greiner et. al. (2001) also reported, using least square approximation for FE runs in optimising frame structures. Approximate models even though are not as accuracy as the actual numerical solutions, can give a reasonable representation of the design landscape, and speed up the search procedure. They can achieve significant savings in computational cost and can be used for solving complex real-life optimisation problems.

3.2 MULTI-OBJECTIVE METHODS

The challenge facing most solution methods is to ensure convergence of well-dispersed solutions close to the true optimal front. Some of the most recent evolutionary search algorithms for multi-objective optimisations are reviewed as follows.

3.2.1 Strength Pareto Evolutionary Algorithm (SPEA)

SPEA is an elitist evolutionary algorithm (Zitzler and Thiele, 1998). The algorithm maintains an external population for storing elite solutions from beginning of the initial population. At each generation, the external and current population is combined and fitness assigned. All non-dominated solutions are assigned fitness equal to the number of solutions they dominate and dominated solutions are assigned fitness worse than the worst solution of any non-dominated solution. Clustering technique is used to maintain diversity.

3.2.2 Pareto-Archived Evolutionary Strategy (PAES)

PAES is a multi-objective evolutionary algorithm (Knowles, Watson, et al., 2000) based on evolutionary strategy. Deb et al (2000) described PAES with one parent and one child. Both are compared, and if the child dominates the parent, it becomes the new parent and the iteration continues. If the parent dominates the child, the

child is discarded and a new child created by mutation. However if either of them dominates each other the choice is made by comparing them with the archived best solutions found so far. If the child dominates any member of the archive, it becomes the new parent and the dominated solution eliminated from the archive. If the child does not dominate any member of the archive, both parent and child are compared for their proximity, with archive solutions. If the child resides in the least crowded region in the parameter space among the archived member it becomes the parent and a copy added to the archive.

3.2.3 Elitist Non-Dominated Sorting Genetic Algorithm (NSGAII)

NSGAII(Deb, Agrawal et al., 2000) is a fast elitist solution algorithm that uses explicit–preservation strategy to maintain diversity among solutions in the non-dominated front. In the elitist strategy, the population is sorted into different non-domination levels and each solution assigned a fitness equal to its non-domination level (where 1 is the best level). Binary tournament selection, crossover and mutation operators are used to create offspring population. Other features of the algorithm include crowding distance assignment procedure (for estimating the distance between two points in the solution space) and the crowded tournament selection operator (guides the selection process towards a uniformly dispersed Pareto-optimal front). The algorithm has been shown to demonstrate better performance than most of other contemporary algorithms (Deb, Agrawal et al., 2000). NSGAII can generate some non-Pareto-optimal solutions if the first non-dominated set is larger than the population (Deb, 2001). This problem was experienced in the current study. It is referred to as "generational elitist problem".

3.2.4 Generalised Regression GA (GRGA)

GRGA is one of the most recent multi-objective GA developed by Tiwari et. al (2001) to handle complex multi-objective optimisation problems having high degrees of inseparable function interaction. An interaction occurs when the effect a variable has on the objective function depends on the values of other variables in the function. The author suggests in his paper that "inseparable function interaction in objective functions may augment one or more of the following features that obstruct convergence to the true (or global) Pareto-optimal front", multi-modality, deception, collateral noise and isolated optimum. GRGA works by attaching a non-linear multi-variable regression analysis module to other optimisation algorithm. The author used NSGAII in their paper, but other optimisation algorithm can be used. The algorithm use regression coefficient to guide the search towards the Pareto front and determine termination conditions for the algorithm. One of the main advantages of this algorithm is that it can be used with different multi-objective solution algorithm. GRGA demonstrates better performance than NSGAII in solving the inseparable function interaction problem present in most complex multi-objective optimisation problems. See (Tiwari, Roy et al., 2001) for more details.

4 ROD ROLLING DESIGN PROBLEM

The Rod rolling process considered is a continuous manufacturing process whereby a square billet (dimension ranging from 100mm to 150mm) referred to as the stock is deformed into a rod size ranging between 5mm to 12mm. The rolling operation is a high speed, high production process in which a pair of rolls rotates at the same peripheral speed in opposite directions. The stock is continuously deformed by passing it through a series of high rolling mill stands. During the rolling process, the stock undergoes changes in the mechanical and thermal characteristics and after final cooling the metallurgical properties. Design of the rolling system involves consideration of the mechanical, thermal and thermo-mechanical behaviour of the process (Sun, Yun , et al., 1998), and the optimisation of roll pass design (Farrugia, 2000). Modelling of the rolling process is used to predict mill parameters (roll separating force, torque) and deformation characteristics such as the lateral spread and the evolution of metallurgical properties. These predictions were obtained using design variables related to the rolls and stock such as geometrical and material characteristics: temperature, friction etc.

Ovality in rod rolling is a geometrical property defined as the percentage difference between the stock height and the width. Ovality is considered important because it helps in forming the rod during rolling process, however it is not desirable in the end product. In this study a different definition of ovality is adopted. Ovality is defined as the difference between the maximum and minimum radial distance of the rod profile. This definition is chosen to mimic its application in the plant. In this work, ovality and the load required for rod deformation is modelled using a meta-modelling technique, and the minimisation of both responses is treated as a multi-objective problem. The problem is considered multi-objective in nature because ovality tends to vary inversely with load. In practice a minimum rod ovality condition requires high contact of the stock with the roll, which results in high loads.

5 META-MODELLING

A meta-model is defined as a model of an underlying simulation model (Kleijnen, 1975; Friedman, 1996). It is an approximation of the simulation program's input/output transformation referred to as a response surface. A typical meta-model approach is the design of experiment (DoE) using regression analysis, also known as analysis of variance (ANOVA). DoE involves making several designs at once and investigating the joint effects of these changes on a response variable. Meta-models offer the following benefits: (1) Insight into the

relationship between output responses y, and the input design variables, x. (2) Fast analysis tools for optimisation and design space exploration since the surrogate models are used in lieu of the expensive computer, and (3) the integration of discipline dependent analysis codes.

The basic meta-model framework adopted in this research is shown in figure 1. A brief discussion of some of the main steps is given below.

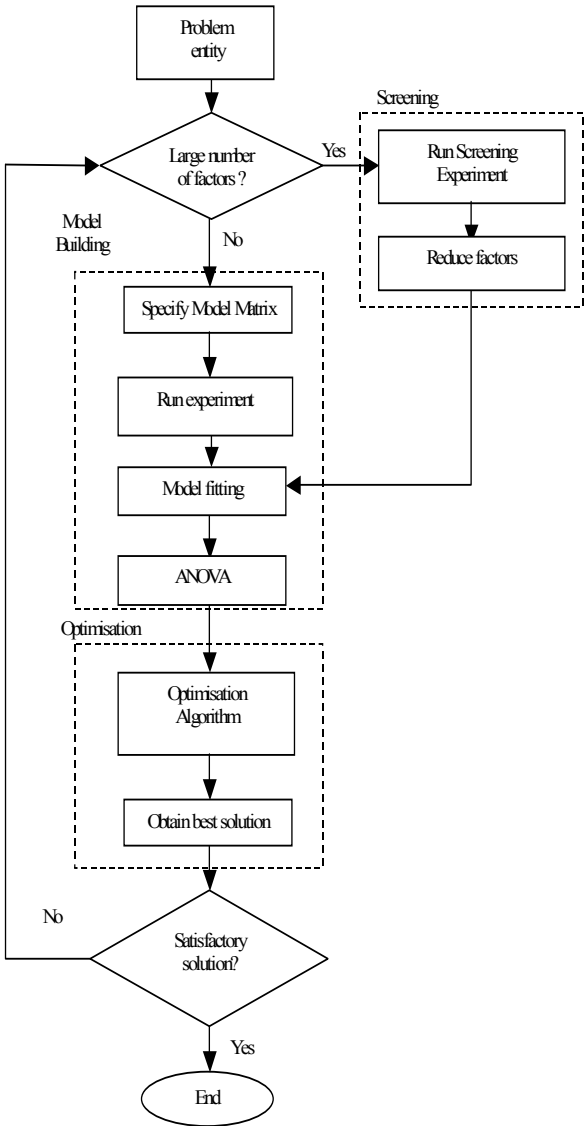


Figure 1: Meta-model approach

Step 1: Problem formulation

This is the first stage of the simulation effort where the problem is defined. The aim at this stage is to understand the nature of the problem, and to define the experimental region (Zeigler, 1976). This is achieved by identifying the candidate parameters and the boundaries that characterise the design space. Existing knowledge is required to identify all the possible parameters involved in the problem space. The output of this stage is a list of inputs and responses with their respective range.

Step 2: Definition of Objective

Defining the objective indicates the question to be answered by the simulation study. The options available in this methodology are screening and optimisation. Screening is based on the '*principle of parsimony'* or *Occam's razor*(Banks, 1998). The aim is to derive a short list of the most important factors from a large number of potentially important factors. In optimisation, the meta-model can be used to determine the set of problem entity input values that optimises a specific objective function.

Step 3: Specification of model matrix

Model matrix implies the type of DoE design (for example 2^{k-p}). The choice of design type is dependent on the objective and the number of factors. This decision is simplified by using existing designs.

Step 4: Fitting meta-model

The simulation run (is define as a single path with fixed values for all its inputs and parameters) is performed to obtain the input and output. This data set is used to estimate the parameter values of the meta-model using least squares. Typically a regression meta-model belongs to one of the following three classes:

Main effects model: (a first-order polynomial):

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Main effects + interaction effects (a first-order polynomial augmented with two factor interactions)

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_k x_k + \beta_{12} x_1 x_2 + \ldots + \beta_{k-1,k} x_{k-1} x_k$$

Quadratic model with quantitative factors (a second-order polynomial, which includes purely quadratic effects)

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \beta_{12} x_1 x_2 + \ldots + \beta_{k-1,k} x_{k-1} x_k + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \ldots + \beta_{kk} x_k^2$$

Step 5: Validation

The data set is validated by carrying out the statistical tests using the Analysis of Variance (ANOVA) table. This tests the hypothesis that each parameter significantly influences the response.

Step 6: Post-processing

Post-processing implies the interpretation and display of the results. The following are options available for

displaying the results: Main effect plot, interaction plot, and half normal probability plot (Daniel plot).

6 APPLICATION OF META-MODELLING APPROACH FOR THE ROD ROLLING DESIGN PROBLEM

6.1 EXPERIMENTAL METHOD

The example described in this paper deals with the multi-objective optimisation (load and load) of oval to round pass. The factors affecting ovality in the rod rolling process can be categorised as; (a) geometrical parameters such as height, width, roll gap, roll radius. (b) Related metallurgical parameters such as strain values, stress components and bulk temperature, (c) process parameters such as friction, roll speed etc. The independent variables especially relevant to the present ovality simulation are height (he), width (w), roll gap (rg), arc radius, roll radius, rolling speed and the bulk temperature. It is important to understand the overall effects and interactions of these parameters on ovality. Roll designers can use this knowledge to design the optimum required ovality that satisfies the conflicting objectives of the process plant (e.g. minimum load) and the product specification (e.g. minimum ovality).

Existing knowledge was used to define region of interest, 5 variables were identified and their operating range specified. A two level fractional factorial DoE augmented with centre points (to test for curvature) was applied to the design problem. The meta-modelling approach was applied as described below:

Table 1:Factors and factor levels used in simulations

Level	Factors				
	Width (W)	Roll Gap (Rg)	Arc Radius (Ar)	Pass Depth (Pd)	Angle (Ar)
1	18	4	66	20	30
-1	16	2	64	18	28.5

Step 1: Fractional factorial (DoE) design

A low cost resolution V design for a two-level 5 factor, fractional factorial design is shown in Table 2. This was augmented with one centre point to test for curvature. Each factor was run at two levels. Resolution V designs are types of designs where no main effect or two-factor interaction is aliased with any other main effect or two factor-interactions (Montgomery, 1997).

FEA simulations were performed using the set-up in Table 2 and the settings in Table 3 as the input value for the FE runs. For each run, values of the measured ovality (O_v) and load (L) were recorded as shown in Table 3.

Table 2:A 2^{5-1} Design

Run	A	B	C	D	E
1	-1	-1	-1	-1	1
2	1	-1	-1	-1	-1
3	-1	1	-1	-1	-1
4	1	1	-1	-1	1
5	-1	-1	1	-1	-1
6	1	-1	1	-1	1
7	-1	1	1	-1	1
8	1	1	1	-1	-1
9	-1	-1	-1	1	-1
10	1	-1	-1	1	1
11	-1	1	-1	1	1
12	1	1	-1	1	-1
13	-1	-1	1	1	1
14	1	-1	1	1	-1
15	-1	1	1	1	-1
16	1	1	1	1	1
17	0	0	0	0	0

Table 3:Input settings and response values from simulation study

Run	W (A)	Rg (B)	Ar (C)	Pd (D)	An (E)	Ov	L
1	16	2	64	18	30	1.17	238.5
2	18	2	64	18	28.5	4.69	299.5
3	16	4	64	18	28.5	3.17	178.0
4	18	4	64	18	30	1.15	232.0
5	16	2	66	18	28.5	1.28	241.0
6	18	2	66	18	30	4.6	293.0
7	16	4	66	18	30	3.2	167.0
8	18	4	66	18	28.5	1.24	226.3
9	16	2	64	20	28.5	3.17	169.3
10	18	2	64	20	30	1.15	222.8
11	16	4	64	20	30	6.55	129.7
12	18	4	64	20	28.5	3.93	159.1
13	16	2	66	20	30	3.22	171.4
14	18	2	66	20	28.5	1.25	233.5
15	16	4	66	20	28.5	6.52	122.2
16	18	4	66	20	30	3.99	154.4
17	17	3	65	19	29.2	1.74	217.3

Step 2: Model Fitting

Regression models of both responses are generated by fitting the model types shown in section 5 step 4 (main effects and interaction effects). The fit with the lowest sum of squares error (highest R^2) was selected, this resulted in the following experimental model as predicted using ANOVA for ovality (O_v) and load (L) as functions of the inputs,

$$f(x_{Ov}) = 3.06 - 0.3925x_1 + 0.5762x_2 + 0.02x_3 + 0.58x_4 - 0.0138x_5 - 0.75x_1x_2 - 0.75x_1x_4 + 0.95x_2x_4 + 0.0175x_2x_5 + 0.604\,x_3x_5 + 0.019x_4x_5 \quad (1)$$

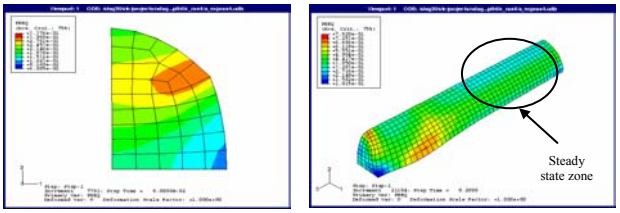
$f(x_L) = 203.28 + 25.21x_1 - 31.29x_2 - 1.25x_3 - 32.05x_4 - 1.25x_5 - 3.35x_1x_2 - 3.07x_1x_4 - 0.77x_1x_5 - 2.36x_2x_3 + 2.32x_2x_4 + 0.95x_2x_5 + 1.33x_3x_4 - 3.4x_3x_5$ (2)

With the inputs expressed in coded [-1,1] units (useful for comparing experimental models). The input was also expressed in engineering units as shown in equation 3 and 4. This can be useful for engineering decision making.

$O_v = 1314.263 + 16.1W - 4.7R_g - 23.53A_r + 9.75P_d - 52.82A_n - 0.75WR_g - 0.75WP_d + 0.95R_gP_d + 0.805A_rA_n + 0.025P_dA_n$ (3)

$L = -8246.94 + 123.7W + 97.74R_g + 113A_r - 73P_d + 306.1A_n - 3.35WR_g - 3.1WP_d - 1.03WA_n - 2.36R_g A_r + 2.32R_gP_d + 1.27R_gA_n + 1.33 A_r P_d - 4.52A_rA_n$ (4)

This model was used to perform the multi-objective optimisation problem.



(a) (b)

Figure 2: Finite Element contour plots of rod profile
(a) Transverse section (b) Full view showing SS zone

Table 4: Parameters used in simulation study

Geometrical parameters		Material specification
Height	30.6 mm	0.08% Carbon steel C 0.087, Si 0.003, Mn 0.34, P 0.025 S 0.02. Hot rolled and annealed Suzuki (4.22)
Roll Radius	250 mm	
Pass Radius	20 mm	
Width (W)	Factor	
Roll Gap (Rg)	Factor	
Arc Radius (Ar)	Factor	
Pass Depth (Pd)	Factor	Process Parameters
Arc Angle (An)	Factor	
Ovality (O_v)	Response	Temperature: 1000 ° C
Load (L)	Response	Roll Speed: 1000 m/s

6.2 Process conditions used in experiment

The choice of geometrical parameters and material properties is discussed in this section. The choice of parameters was driven by the need to mimic the real design problem experienced on the plant in the study. The results obtained can then be validated using existing domain knowledge.

Geometrical parameters:

Expert domain knowledge was used to select the geometrical parameters and the region of interest defined according to the ranges shown in Table 1. These five factors are varied in the simulation runs. Other parameters such as height roll radius and pass radius are kept constant to make the simulations comparable. A summary of these geometrical parameters is shown in Table 4.

Material specification

The material specification used in the study is shown in Table 4. The specification was identical for all runs.

Process parameters

The same loading conditions were applied in all the simulations so that the response could be obtained under similar conditions.

Finite Element Analysis and Data Extraction

The finite element runs were performed using Abaqus version 6.2.2. The mesh was generated using Patran software. A contour plot of PEEQ (equivalent plastic strain) for a typical run is shown in figure 2a. Results showing the deformation characteristics are taken in the steady state (SS) zone of the rod. The SS is defined as the region where the deformation characteristics is assumed to be uniform. This zone is identified by using a qualitative judgement to identify region along the rod (figure 2b) where the contour profiles are parallel.

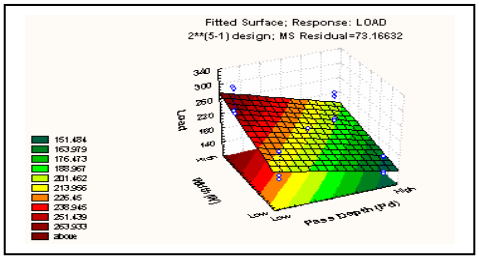


Figure 3: Interaction effects on Ovality Response

6.3 MULTI-OBJECTIVE OPTIMISATION OF ROD DESIGN PROBLEM USING NSGAII

NSGAII (Deb, Agrawal, et al., 2000) was considered suitable for optimising the response function described in section 6.1. (Equations 3 and 4). This is because NSGAII has been shown to perform well on equations with low-level inseparable function interaction (Deb, 2001). If the model were developed with higher order interaction terms, then GRGA would have being used. The models shown in section 6.1 (equations 3 and 4) were used as the fitness function in NSGAII. The parameters were represented using binary coding. The crowded tournament selector operator was used to select new offsprings. The experiment was run with a population of size 100 for 1000 generation with a crossover probability of 0.8 and a mutation probability of 0.05.

7 DISCUSSION AND RESULTS

7.1 METAMODELS

The response ovality and load from the simulation results were recorded as shown in Table 3 and the data used to perform the ANOVA shown in Table 5. The result suggests that for the ovality response, the most significant terms are A (W), B (Rg), D (Pd), AB, AD, BD and CE. The sum of square of these terms accounts for over 96% of the total variability in the response. Figure 3 shows interaction effect plots of pass depth and roll gap. These response surface have been generated whilst the third variable has been held constant. This plot indicates that pass depth has a much stronger effect on ovality when the roll gap is at high level. For minimum ovality, the roll gap should be at the low level and pass depth at high level. For the load response, factors A, B and D show the most significant effect on the load response. These three factors explain 98% of the variation in the load. Interaction effect of pass depth and width is plotted in figure 4. Again the plot indicates that pass depth has a much stronger effect on ovality where minimum ovality occurs at high pass depth level and low width level.

Table 5: Analysis of Variance (ANOVA) associated with regression model in equations 3 and 4

Ovality			Load		
Term	DoF	SSq	Term	DoF	SSq
A (W)	1	2.465	A (W)	1	10175.9
B (Rg)	1	5.313	B (Rg)	1	15664.4
C (Ar)	1	0.0064	C (Ar)	1	25.0
D (Pd)	1	5.382	D (Pd)	1	16432.0
E (An)	1	0.003	E (An)	1	24.9
AB	1	8.97	AB	1	179.1
AD	1	9.0	AD	1	151.2
BD	1	14.4	AE	1	9.5
AE	1	0.005	BC	1	89.0
CE	1	5.832	BD	1	86.4
DE	1	0.006	BE	1	14.4
Model	11	51.38	CD	1	28.2
Error	5	1.85	CE	1	184.3
Total	16	53.23	Model	13	43064.3
SSq: Sum of Squares			Error	3	220
DoF: Degree of Freedom			Total	16	43284.3

7.2 MULTI-OBJECTIVE OPTIMISATION (NSGAII)

The result in figure 5 shows the plots of solution results obtained by running the NSGAII algorithm. NSGAII was used to minimise both load and ovality using the GA parameters described in section 6.3 and equation 3 and 4 as the objective function. NSGAII was run ten times with different random number seeds. The best convergence is presented in figure 5. Seven out of ten runs obtained similar results. Therefore it is likely that NSGAII has converged to the global Pareto front. It can also be seen from figure 5 that NSGAII converges to the Pareto optimal front with a good spread of multiple optimal solutions. Table 6 shows decision variable values at two optimal solution points picked at one and two in figure 5, the extreme ends of the Pareto front. This demonstrates how multiple optimal solution can help produce a variety of optimal solutions.

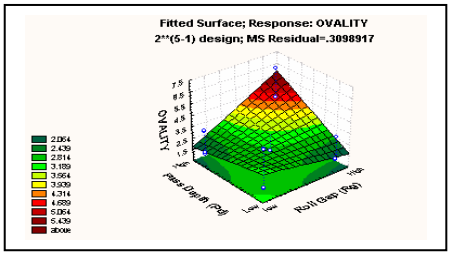


Figure 4: Interaction effects on Load Response

Table 6:Variable values for optimal solutions

Point	W	Rg	Ar	Pd	An	O_v	Load
1	16	2.4	64.7	18.7	30	2.2	216.9
2	16	4	66	20	30	7.7	130.5

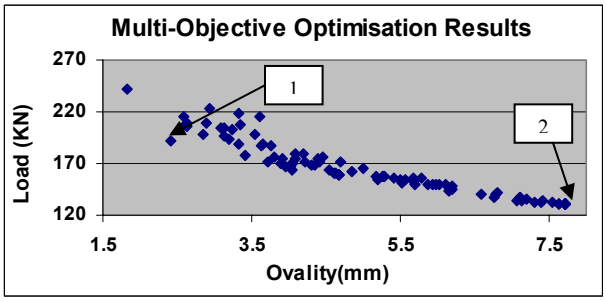


Figure 5: Multi-Objective Optimisation Solution Plot

8 FUTURE RESEARCH ACTIVITIES

The limitations in the current modelling and optimisation approach and the corresponding research activities are listed below.

- Qualitative (QL) knowledge cannot be used within the optimisation phase of the current methodology. It will be very useful to develop a framework explore the effect of QL variables on quantitative variables. This information can be used to guide the search in the optimisation process.
- The GA runs differ in results when different parameter settings and scaling for the decision variable space are used. The choice of the best parameter settings is difficult as it depends on the nature of the problem. Developing parameter-less GA's provides a challenging research area.

9 CONCLUSION

Traditional solution methods for optimising complex real life engineering problems can be very expensive and often results in sub-optimal solutions. A multi-objective optimisation approach is presented to address expensive computational cost of large FE runs using meta-models. This technique is effective in approximating FE runs and exploring complex search spaces for achieving multiple global optimal solutions. NSGAII was applied to a rod-rolling problem. NSGAII converged to the Pareto optimal front showing good results. Multiple optimal solutions give the opportunities to deliver variety of optimal designs in the presence of existing qualitative knowledge.

Acknowledgements

The authors wish to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) and Corus R, D and T, Swinden Technology Centre, UK.

References

Banks, J. (1998). *Experimental Design for Sensitivity Analysis, Optimization*. Handbook of Simulation. Georgia, John Wiley &Sons**:** 173-223.

Cerrolaza, M. and Annicchiarico, W. (1999). Finite elements, genetic algorithms and beta- splines: a combined technique for shape optimization. Finite Element in Analysis and Design **33**: 125-141.

Chen, S. (2001). “An approach for impact structure optimization using the robust genetic algorithm.” Finite Elements in Analysis and Design **37**: 431-446.

Chen, T. and Lin, C. (2000). *"Determination of optimum design spaces for topology optimization."* Finite Elements in Analysis and Design **36**: 1-16.

Deb, K. (2001). Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Ltd.

Deb, K.,Agrawal, S.,Pratap, A. and Meyarivan, T. (2000). *"A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II."* Parallel Problem Solving from Nature VI (PPSN-VI), Paris, 849-858.

Deb, K. and Gulati, S. (2001). *"Design of truss-structures for minimum weight using genetic algorithms."* Finite Elements in Analysis and Design **37**: 447 - 465.

Farrugia, D. (2000). *"Advanced analysis system for roll pass design."* 11th Abaqus User's group, Warrington, UK: 1-18.

Friedman, L. W. (1996). *The Simulation Metamodel*. Dordrecht, Netherlands, Kluwer.

Greiner, D.,Winter, G. and Emperador, J. M. (2001). *"Optimising frame structures by different strategies of genetic algorithms."* Finite Elements in Analysis and Design **37**: 381-402.

Kleijnen, J. P. C. (1975). "A comment on Blanning's meta-model for sensitivity analysis: the regression meta-model in simulation." Interfaces **5**(3): 21-23.

Kleijnen, J. P. C. and Sargent, R. G. (2000). *"A methodology for fitting and validating meta models in simulation."* European Journal of Operational Research **120**: 14-29.

Knowles, J. D., Watson, R.A., and Corne, D. W. (2000). *"Approximating the non-dominated front using the Pareto archived evolution strategy."* Evolutionary Computation Journal **8**(2): 149-172.

Montgomery, D. C. (1997). *Design and Analysis of Experiments*, John Wiley & Sons.

Myers, R. H. and Montgomery, D. C. (1995). *Response Surface Methodology*, John Wiley & Sons.

Oduguwa, V. and Roy, R., (2001). "*Qualitative and Quantitative Knowledge in Engineering System Design."* In: Advances in Manufacturing Technology XV, 17th National Conference on Manufacturing Research, Cardiff University, UK, edited by Pham, D. T., Dimov, S. S., O'Hagan, O., Professional Engineering Publishing: 81-86.

Quagliarella, D. and Vicini, A. (2001). *"Viscous single and multi-component airfoil design with genetic algorithms."* Finite Elements in Analysis and Design **37**: 365-380.

Roy, R. (1997). "Adaptive Search and the Preliminary Design of Gas Turbine Blade Cooling System." PhD Thesis, University of Plymouth, Plymouth.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989). *"Design and Analysis of Computer Experiments."* Statistical Science **4**(4): 409-435.

Sun, C. G., Yun, C. S., Chung, J. S. and Hwang, S. M., (1998). *"Investigation of Thermo-mechanical Behavior of Work Roll and a Roll Life in Hot Strip Rolling. "* Metallurgical and Materials Transactions A **29A**: 2407-2424.

Tiwari, A., Roy, R., Jared, G. and Munaux, O. (2001). *"Interaction and Multi-objective Optimisation."* Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), edited by Spector, L., Goodman, E., Wu, A., Langdon, W.B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. and Burke, E. 671-678, Morgan Kaufmann Publishers, San Francisco, USA.

Zeigler, B. P. (1976). *Theory of Modelling and Simulation*, John Wiley & Sons.

Zitzler, E. and Thiele, L. (1998). *"Multi-objective optimisation using evolutionary algorithms- A comparative case study."* Parallel Problem Solving from Nature V (PPSN-V), edited by Eiben, A.E., Back, T., Schoenauer, M. and Schwefel, H.P., 292-301, Springer, Berlin, Germany.

Genetic Search for Fixed Channel Assignment Problem with Limited Bandwidth

Eun-Jong Park, Yong-Hyuk Kim, and Byung-Ro Moon
School of Computer Science & Engineering, Seoul National University
Shillim-dong, Kwanak-gu, Seoul, 151-742 Korea
{silver,yhdfly,moon}@soar.snu.ac.kr

Abstract

We propose a hybrid genetic algorithm for the fixed channel assignment problem with limited bandwidth. A local optimization algorithm was devised to enhance the genetic algorithm's fine-tunning. In a matrix representation, the local optimization algorithm improves on a chromosome by horizontal and vertical moves of channels. The crossover is directly performed on two-dimensional matrices. Experimental results showed dramatic improvement against previous works.

1 Introduction

As the demand of mobile telecommunication sharply grows, the effective use of limited resources becomes more important. In a mobile system, the channel assignment problem is to assign channels optimally for the requests of cells. In this paper, a channel corresponds to a frequency band.

When we assign the channels to the cells, EMC (Electro-Magnetic Compatibility) constraints [14] must be satisfied. There are three types of constraints, namely, i) the cochannel constraint (CCC): the same channel cannot be assigned to certain pairs of radio cells simultaneously, ii) the adjacent channel constraint (ACC): channels adjacent in the frequency spectrum cannot be assigned to adjacent radio cells simultaneously, and iii) the cosite constraint (CSC): the channels assigned to the same radio cell need minimal frequency separation between cells. If these EMC constraints are not satisfied, interference occurs. Under these constraints, we can define the channel assignment problem as follows [4]:

Minimize Z

subject to

$$\sum_{p=1}^{Z} f_{ip} = d_i \qquad \text{for } 1 \le i \le N,$$

$$|p-q| \ge C_{ij} \qquad \text{for } 1 \le p,q \le Z \text{ and } 1 \le i,j \le N \text{ such that } f_{ip} = f_{jq} = 1, \text{ and}$$

$$f_{ip} = \begin{cases} 0 \\ 1 \end{cases} \qquad \text{if channel } p \text{ is } \begin{cases} \text{not assigned} \\ \text{assigned} \end{cases} \text{to cell } i$$

where

Z: the number of available channels,
N: the number of cells,
D: demand vector $D = (d_1, d_2, ..., d_N)$, where d_i is the demand of channels for cell i, and
C: compatibility matrix $C_{N \times N}$, where each element C_{ij} represents the required minimum distance of separations between two the channels assigned to cell i and cell j.

The status of channels allocated to each cell can be represented with an $N \times Z$ binary matrix F (see Figure 4). Each element of F has value 0 or 1; if the j^{th} channel is assigned to the i^{th} cell, f_{ij} has value 1, otherwise f_{ij} has value 0.

Consider a simple channel assignment problem proposed in [14]; there are 4 cells, and the compatibility matrix and the demand vector are given as

$$C = \begin{pmatrix} 5 & 4 & 0 & 0 \\ 4 & 5 & 0 & 1 \\ 0 & 0 & 5 & 2 \\ 0 & 1 & 2 & 5 \end{pmatrix}$$

and $D = (1,1,1,3)$. The diagonal elements $C_{ii} = 5$ mean that if any two channels are assigned to the same cell, interference occurs unless the frequencies are apart by at least 5. With the above compatibility matrix and the demand vector, an optimal assignment is shown in Figure 1.

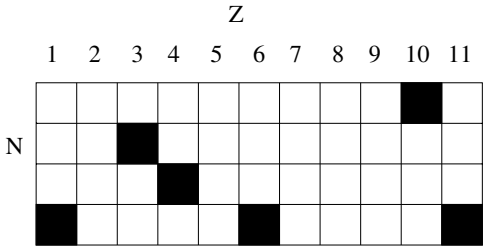


Figure 1: An interference-free assignment for the network with 4-cell and 11-channel.

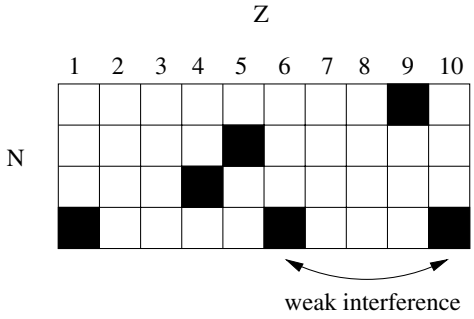


Figure 2: A weak-interference assignment for the network with 4-cell and 10-channel.

However, if there are only ten channels available, all the demands cannot be satisfied; we need to minimize the severity of interference. Figure 2 shows a solution with relatively weak cosite interference in cell 4 [2].

Generally, the fixed channel assignment problem (FCAP) is equivalent to the generalized graph-coloring problem [13][14][15]. We can obtain a graph with vertices and edges, where each vertex corresponds to a request of cells and the weight of the edge between two vertices is the required minimum separation for assigning two cells [15]. If all C_{ij}'s are 0 or 1, then there exists only the cochannel constraint and the FCAP reduces to the classical graph-coloring problem. Since the graph-coloring problem (either classical or generalized) is NP-complete [8], so is FCAP [14]. Hitherto a number of approximation algorithms have been proposed for FCAP. These include graph theoretic ordering approaches [14], neural network [6][15], simulated annealing [5], and genetic algorithms (GAs) [4][13].

The studies of FCAP can be classified into two approaches. One focuses on minimizing the total number of channels assigned to the cells with all constraints satisfied [14][16]. The other proposes appropriate cost functions and attempts to minimize the costs [4][13][6]. The cost should be zero when an assignment is conflict-free.

There have been a number of studies for FCAP using GAs. Ngo and Li [13] determined the number of available channels as lower bound using a graph theoretic method, and considered the interference cost as the fitness value. As Ngo and Li [13] suggested, Dirk and Ulrich [4] assumed the total number of available channels as lower bound according to the lower-bound rule of Gamst [7], used the blocking rate as the evaluation cost, and searched for the assignments with cost value zero. They searched for the conflict-free assignments with minimum channel span.

However, in practice, the available channels are limited, and the requests for channels often overflow beyond the capacity. In this situation, minimizing the total channel span is meaningless [9] and it is more useful to attempt to obtain the best assignments possible, given the number of channels [15]. Jin *et al.* [10] suggested a new formulation for a limited channel bandwidth below the lower bound. With the limited channel environment, all constraints cannot always be satisfied and we should allow blocked calls and/or the interference [10][9][15]. They consider the non-assigned requests and the violation of EMC constraints as blocking rate and interference cost, respectively. An optimal assignment maximally satisfies the demands of cells and minimizes the interference between cells. We use this model in this study.

There are two types of channel assignment problems: the fixed channel assignment and the dynamic channel assignment. In fixed channel assignment (FCA), one-time assignment is performed; in dynamic channel assignment (DCA), assignments are repeated with changing requests. FCA is more important in situation with heavy traffic loads [13]. Even in the situation in which DCA is used, the initial solution is usually provided by FCA, and DCA keeps modifying on it [15]. The running time is thus not very critical in FCA.

In this paper, we propose a hybrid genetic algorithm for FCA with limited channel bandwidth. Basically, we follow the formulation proposed in [10]. We designed a GA with two-dimensional chromosomes and an effective local optimization heuristic with horizontal/vertical searches.

The rest of the paper is organized as follows. In Section 2, we describe the formulation used in this paper. The proposed hybrid GA and local optimization algorithm are presented in Section 3. The experimental results are provided in Section 4. Finally, Section 5 summarizes the study.

2 Fixed Channel Assignment Problem

If a call cannot be assigned a channel, we say it is *blocked*. The *blocking rate* (the damage cost by blocked calls) is defined as,

$$\sum_{i=1}^{N} \sum_{j=DF_i+1}^{n_i} P(X_i = j)(j - DF_i)$$

where

F_i: the amount of channels assigned to cell i,
X_i: a random variable of required channels in cell i,
n_i: the expected number of requests in cell i, and
D: the number of channels provided by each frequency.

In assignments with the fixed channel condition, EMC constraints may be violated and the violation brings *interference*. EMC constraints are provided by the compatibility matrix C. In the matrix, each diagonal element C_{ii} represents the cosite constraint, and each non-diagonal element C_{ij} represents the minimum separation distance between any two frequencies assigned to cells i and j [13]. If $C_{ij} = 1$, it represents the cochannel constraint, and if $C_{ij} = 2$, it represents the adjacent channel constraint. Interference cost is defined as follows:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p=1}^{Z} \sum_{q=1}^{Z} f(i,j,p,q)$$

where

N: the number of cells,
Z: the number of available frequencies,

$$f(i,j,p,q) = \begin{cases} 0, & \text{if } |p-q| \geq C_{ij} \\ f_{ip} f_{jq} \psi_C(C_{ij} - |p-q|), & \text{if } |p-q| < C_{ij} \text{ and } i = j \\ f_{ip} f_{jq} \psi_A(C_{ij} - |p-q|), & \text{otherwise.} \end{cases}$$

ψ_C and ψ_A are some strictly increasing functions.

Since the number of available channels is limited, it is hard to satisfy both of the constraints. The total damage due to an assignment is formulated as

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p=1}^{Z} \sum_{q=1}^{Z} f(i,j,p,q) + \alpha \sum_{i=1}^{N} \sum_{j=DF_i+1}^{n_i} P(X_i = j)(j - DF_i)$$

where α is a weighting factor. In the formula, the first term represents the interference cost and the second term represents the blocking cost.

3 A Hybrid GA for the FCAP

In this section, we describe the proposed GA for FCAP. Figure 3 shows the template of a hybrid steady-state GA.

```
create initial population of a fixed size;
do {
        choose parent1 and parent2 from population;
        offspring ← crossover(parent1, parent2);
        mutation(offspring);
        local-optimization(offspring);
        replace(population, offspring);
} until (stopping condition);
return the best individual;
```

Figure 3: A typical hybrid steady-state GA

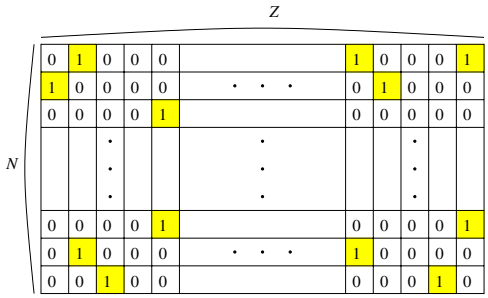


Figure 4: Two-dimensional chromosome for a solution

3.1 2D Representation

We represent a solution by a binary $N \times Z$ matrix. N is the number of cells and Z is the number of channels. If a gene $f_{ip} = 1$, then the p^{th} channel is assigned to the i^{th} cell (e.g., see Figure 4).

3.2 2D Crossover

A two-dimensional encoding/crossover pair can reflect more geographical linkages of genes than one-dimensional encoding/crossover pairs [12]. Cohoon and Paris [3] proposed a two-dimensional crossover which chooses a small rectangle from one parent and copies the genes in the rectangle into the offspring, with the rest of the genes copied from the other parent. Anderson *et al.* [1] suggested the block-uniform crossover on $n \times n$ grid. It divides the grid into $i \times j$ blocks at random; each block of one parent is interchanged randomly with the corresponding block of the other parent based on a pre-assigned percentage.

Although two-dimensional encoding can preserve more geographical relationships among the genes, when traditional straight-line-based cutting strategies are used, the power of new-schema creation is far below that of the crossovers on linear encodings [11].

Geographic crossover was suggested to resolve this problem [11] [12]. In the case of a two-dimensional encoding, it chooses a number of monotonic lines,

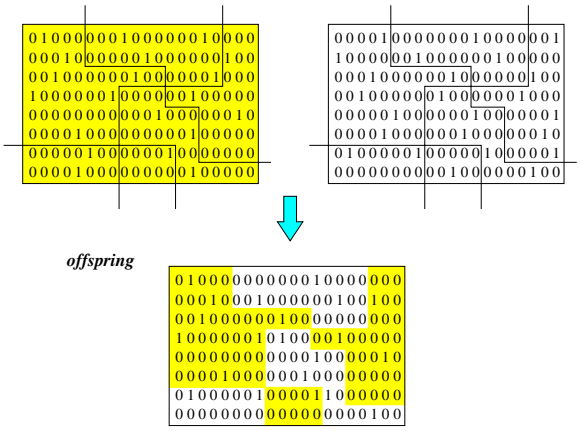


Figure 5: An example of geographic crossover

divides the chromosomal domain into two equivalence classes, and alternately copies the genes from the two parent chromosomes. We used geographic crossover in this work. Figure 5 shows an example geographic crossover operator for this problem. By combining two-dimensional representation and geographic crossover, we are pursuing both reduced information loss in the stage of encoding and the power of new-schema creation.

3.3 Mutation

On each cell i, we generate two random numbers. One is to get a channel number p and the other is a binary random number. We assign 0 or 1 to f_{ip} depending on the number. We control the total number of 1's in row i not to exceed d_i of the demand vector in any case.

3.4 HV-Move: the Local Optimization

We devised a local search heuristic to fine-tune around local optima. First, we fix the blocking rate and reduce the interference cost by a row-based search. Next, we reduce the interference and blocking cost simultaneously by a column-based search. The process of local search is performed by moving 1's to more attractive chromosomal positions.

Horizontal Search (Row-Based Search)

In the horizontal search, we redistribute the gene value 1's in each row so that the interference cost in the corresponding cell is minimized. In this search, the number of assigned channels in each cell does not change.

Vertical Search (Column-Based Search)

The vertical search is performed after the horizontal search and reduces the blocking rate and interference

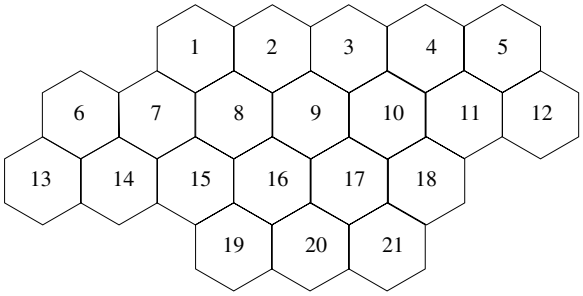


Figure 7: The 21-cell system

cost simultaneously. We redistribute the gene value 1's in each column so that the sum of blocking rate and interference cost is minimized. This search changes the number of channels in the cells.

Figure 6 shows the outline of the local optimization algorithm. For each gene with value 1, we tries to move 1 to another position in the same row. If the move gives some gain, the value 1 moves. Then we move the value 1 to another position in the same column if the move gives some gain. Mark the position to which the value 1 finally moved. After performing the above process for all genes with value 1, we repeat the above process as far as there is at least one marked position.

4 Experimental Results

4.1 Benchmark Problems

The 21-cell system is a useful benchmark for the channel assignment problem (see Figure 7). The compatibility matrix C_3, C_4, C_5, and the demand vector D of each cell are based on [10], [9], and [6]. Figure 9 shows the compatibility matrices and Figure 10 represents the demand vectors. The benchmark set is composed of 8 problems, originated from [10] and [9]. Table 1 shows the specification of the problems. In our experiments, we used the function $\psi_C(X) = 5^{x-1}$ and $\psi_A(X) = 5^{2x-1}$ to evaluate chromosomes and set the weight of blocking cost $\alpha = 1000$[1], as in [10] and [9].

4.2 Experimental Parameters

In our experiments, the population size was set to 50. In selection, the probability to select the best chromosome was given four times higher than the worst. Mutation rate was 0.01%. Our GA is a steady-state GA; each generation produces one offspring and re-

[1] The weight of blocking cost was mistakenly written as 10,000 in [10] and [9]. We corrected it to 1,000 by personal communication with them.

```
HV–Move
{
    count ← 0;
    mark 1 on all the positions of value 1;
    repeat
    {
        count ← count + 1;
        for each row
            for each gene with mark count in the row
            {
                if there exists gain by moving the value 1 to another position in the same row
                    then move the value 1;
                if there exists gain by moving the value 1 to another position in the same column
                    then move the value 1;
                if there was any move in the above two trials
                    then mark the finally moved position with count;
            }
    } until (there is nothing marked with count);
}
```

Figure 6: The outline of the local optimization algorithm

places with it one of the chromosomes in the population. The stopping condition is a fixed number of generations (we set the number to be 10,000). We performed the experiments on Pentium III 750 MHz.

4.3 Results

The experimental results are summarized in Table 2. The costs were dramatically reduced compared with [10] and [9]. In an extreme case (Problem 1), we found a solution of cost 0.385 while Horng *et al.* [9] reported 203.266 as the best solution cost.

Figure 8 shows the numbers of horizontal moves and vertical moves over the generations. The numbers were summed up every 200 generations. In the figure, one can observe that the horizontal moves occur more often than the vertical moves. Although not very often, the vertical moves steadily occurred over the generations.

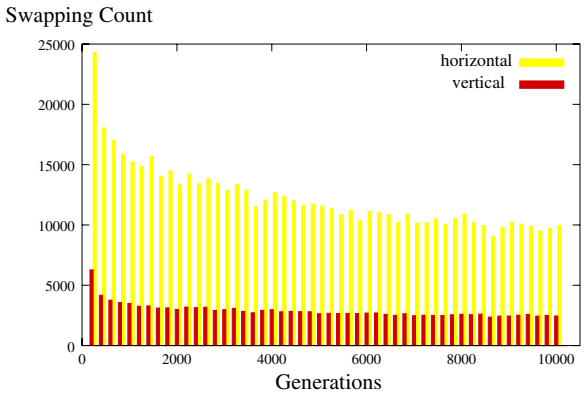


Figure 8: The swapping count in the local optimization

5 Conclusions

A hybrid GA was proposed to solve the fixed channel assignment problem that assign limited channels to the requests of cells. Solutions were represented by two-dimensional chromosomes and the geographic crossover was applied. To help the GA's fine-tunning, we devised a local optimization heuristic that performs row-based search and column-based search.

We may consider the parallelization of our method. Although the suggested GA showed dramatic improvement over the previous studies, we believe that there still remains room for further improvement, particularly in the local optimization part. We plan to apply our algorithm to other benchmark problems [15] and larger scale problems.

Acknowledgments

We thank Mr. Ming-Hui Jin for providing us the evaluation function procedure. This work was supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

Table 1: The Specification of the Problems

Problem	No. of cells(N)	No. of available channels(Z)	Compatibility matrix(C)	Communication load table(μ, δ)
1	21	60	C_3	$D_1^{\dagger}$
2	21	60	C_4	D_1
3	21	60	C_5	D_1
4	21	60	C_4	D_2
5	21	60	C_5	D_2
6	21	40	C_5	D_1
7	21	40	C_5	D_2
8	21	64	C_5	D_3

† The demand vector was mistakenly written as D_3 in [10] and [9].
We corrected it to D_1 by personal communication with them.

Table 2: Experimental Results

Problem	Previous works			Our GA results	
	Best [10]	CPU[1] [10]	Best [9]	Best(Average[3])	CPU[2]
1	217.947	34976	203.266	0.385(0.510)	65504
2	276.623	42807	271.366	27.945(30.881)	88692
3	2013.751	39226	1957.366	63.089(79.346)	89918
4	950.995	31465	906.299	675.849(684.134)	95585
5	4495.609	45712	4302.298	1064.090(1092.484)	87905
6	4857.711	27412	4835.366	1149.755(1227.302)	35790
7	21700.624	54426	20854.300	5636.684(5831.756)	37323
8	58089.148	42248	53151.570	41883.012(41967.549)	135224

1. CPU seconds on Pentium II 400 MHz.
2. CPU seconds on Pentium III 750 MHz.
3. Average over 30 runs.

References

[1] C. A. Anderson, K. F. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.

[2] G. Chakraborty. An efficient heuristic algorithm for channel assignment problem in cellular radio network. *IEEE Trans. on Vehicular Technology*, 50:1528–1539, November 2001.

[3] J. P. Cohoon and W. Paris. Genetic placement. In *IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.

[4] B. Dirk and K. Ulrich. A new strategy for the application of genetic algorithms to the channel-assignment problem. *IEEE Trans. on Vehicular Technology*, 48(4):1261–1269, 1999.

[5] M. Duque, D. Kunz, and B. Rueber. Channel assignment for cellular radio using simulated anealing. *IEEE Trans. on Vehicular Technology*, 42(1):14–21, 1993.

[6] N. Funabiki and Y. Takefuji. A neural network parallel algorithm for channel assignment problems in cellular radio networks. *IEEE Trans. on Vehicular Technology*, 41(4):430–437, 1992.

[7] A. Gamst. Some lower bounds for a class if frequency assignment problems. *IEEE Trans. on Vehicular Technology*, 35:9–14, 1986.

[8] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completen ess*. Freeman, San Francisco, 1979.

[9] J.Z. Horng, M.H. Jin, and C.Y. Kao. Solving fixed channel assignment problems by evolutionary approach. In *Genetic and Evolutionary Computation Conference*, pages 351–358, 2001.

[10] M.H. Jin, H.K. Wu, J.Z. Horng, and C.H. Tsai. An evolutionary approach to fixed channel assignment problems with limited bandwidth constraint. In *IEEE International Conference on Communications*, pages 398–412, 2001.

[11] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *Sixth International Conference on genetic Algorithms*, pages 96–103, 1995.

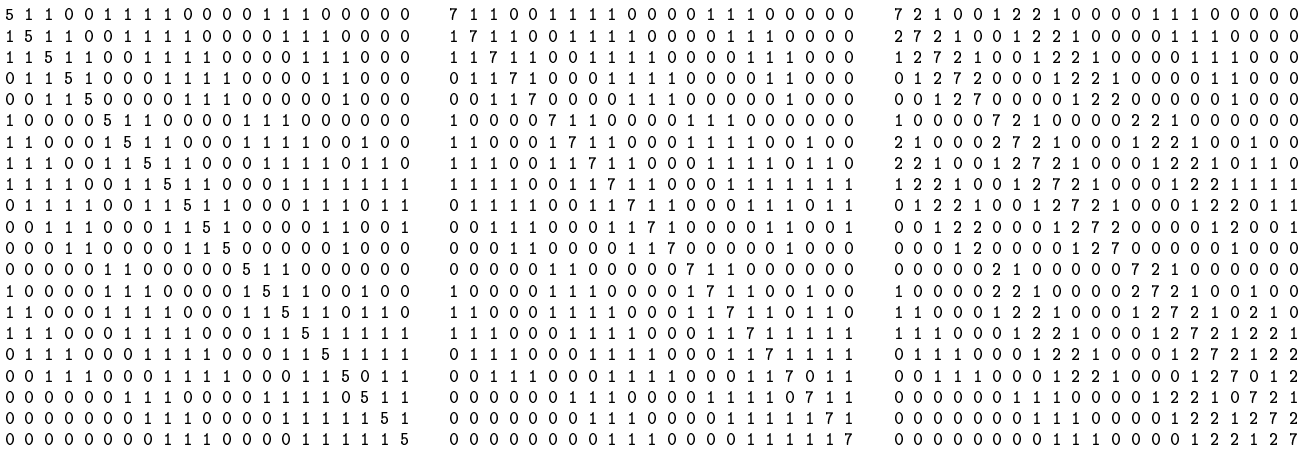
[12] B. R. Moon, Y. S. Lee, and C. K. Kim. GEORG: VLSI circuit partitioner with a new genetic algorithm framework. *Journal of Intelligent Manufacturing*, 9(5):401–412, 1998.

[13] C.Y. Ngo and V.O.K. Li. Fixed channel assignment in celluar radio networks using a modified genetic algorithm. *IEEE Trans. on Vehicular Technology*, 47(1):163–171, 1998.

[14] K.N. Sivarjan, R.J. McEliece, and J.W. Ketchun. Channel assignment in celluar radio. In *IEEE Vehicular Technology Conference*, pages 846–850, 1989.

[15] K. Smith and M. Palaniswami. Static and dynamic channel assignment using neural networks. *IEEE Journal on Selected Areas in Communications*, 15(2):238–249, 1997.

[16] W. Wang and C.K. Rushforth. A adaptive local-search algorithm for the channel-assignment problem(CAP). *IEEE Trans. on Vehicular Technology*, 45(3):459–466, 1996.



(a) The compatibility matrix C3 (b) The compatibility matrix C4 (c) The compatibility matrix C5

Figure 9: The compatibility matrices

cell	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
μ	5	5	5	8	12	25	30	25	30	40	40	45	20	30	25	15	15	30	20	20	25
σ	1.2	1.1	1.15	1.6	2.24	5.0	5.72	5.0	6.1	8.1	8.02	9.11	4.07	5.99	5.61	3.14	3.07	5.86	4.12	3.98	5.18

D_1

cell	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
μ	8	25	8	8	8	15	18	52	77	28	13	15	31	15	36	57	28	8	10	13	8
σ	1.61	4.88	1.52	1.49	1.61	3.11	3.52	9.73	11.62	5.1	2.15	2.66	4.72	2.77	4.93	8.64	3.92	1.25	1.72	2.14	1.27

D_2

cell	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
μ	59.38	105.56	95.79	92.83	61.35	191.81	39.53	62.84	42.58	34.73	36.02	41.59	80.7	35.67	61.91	42.22	65.61	52.45	85.78	33.9	80.62
σ	6.23	8.98	14.07	11.08	5.61	21.62	6.78	8.72	10.3	6.63	7.92	6.29	11.34	6.02	5.61	5.96	7.39	5.86	9.92	3.52	6.18

D_3

Figure 10: Communication load tables

Comparison of Methods for Using Reduced Models to Speed Up Design Optimization

Khaled Rasheed
Computer Science Department
The University of Georgia
Athens, GA 30605, USA
khaled@cs.uga.edu
Phone:(706)542-3444

Swaroop Vattam
Artificial Intelligence Center
The University of Georgia
Athens, GA 30605, USA
svattam@ai.uga.edu
Phone:(706)542-0358

xiao Ni
Artificial Intelligence Center
The University of Georgia
Athens, GA 30605, USA
xiaoni@ai.uga.edu
Phone:(706)542-0358

Abstract

In this paper we compare two methods for forming reduced models to speed up genetic-algorithm-based optimization. The methods work by forming functional approximations of the fitness function which are used to speed up the GA optimization. One method speeds up the optimization by making the genetic operators more informed. The other method speeds up the optimization by genetically engineering some individuals instead of using the regular Darwinian evolution approach. Empirical results in several engineering design domains are presented.

1 Introduction

This paper concerns the application of Genetic Algorithms (GAs) in realistic engineering design domains. In such domains a design is represented by a number of continuous design parameters, so that potential solutions are vectors (points) in a multidimensional vector space. Determining the quality ("fitness") of each point usually involves the use of a simulator or some analysis code that computes relevant physical properties of the artifact represented by the vector, and summarizes them into a single measure of merit and, often, information about the status of constraints. For example, the problem may be to design a supersonic aircraft capable of carrying 70 passengers from Chicago to Paris in 3 hours. The goal may be to minimize the takeoff mass of the aircraft. The constraints may include something like "the wings must be strong enough to hold the plane in all expected flight conditions".

Some of the problems faced in the application of GAs (or any optimization technique for that matter) to such problems are:

- Not all points in the space are legitimate designs — some points in the search space ("unevaluable points") cause the simulator to crash, and others ("infeasible points"), although evaluable by the simulator, do not correspond to physically realizable designs.
- The simulator will often take a non-negligible amount of time to evaluate a point. The simulation time ranges from a fraction of a second to, in some cases, many days.
- The fitness function may be highly non-linear. It may also have all sorts of numerical pathologies such as discontinuities in function and derivatives, multiple local optima, ..etc.

Fortunately, in many of these domains so-called "reduced models", which provide less-accurate but more efficient estimates of the merit of an artifact, are either readily available or can be learned online (i.e. in the course of the optimization) or off-line (i.e. by sampling and building a response surface before optimization). This paper compares methods for the modification of GAs specifically intended to improve performance in realistic engineering design domains in which no reduced models are available a priori. These methods form approximations of the fitnesses of the points encountered during the course of the GA optimization. The approximations are then used to speed up the GA. We compare two methods for improving the GA's performance. One is the idea of informed operators (IO) presented in [7] which uses the approximations to make the GA operators such as crossover and mutation more effective. The other is a variation of the genetic engineering (GE) idea presented in [8] which improves the efficiency of the GA optimization by replacing some of the regular Darwinian iterations, which generate new individuals using crossover and/or mutation, with iteration in which individuals are generated by running

a mini-optimization using the approximations and returning the best point found therein.

The use of reduced models to save time in evolutionary optimization dates all the way back to the sixties. Dunham et al. [4] worked with a two level problem in which they used an approximate model most of the time and only used the accurate/expensive model in the final stages of refinement. Numerous research efforts compute a response surface approximation and use it instead of the very expensive evaluation function with no looking back [9]. Other approaches rely on special relations between the approximate and accurate model to develop interesting multilevel search strategies. A notable class of such methods [10] focus on building variants of injection island genetic algorithms (iiGAs) for problems involving finite element analysis models. The approach was to have many islands using low accuracy/cheap evaluation models with small numbers of finite elements that progressively propagate individuals to fewer islands using more accurate/expensive evaluations. A recent approach [11] uses a functional approximation method to form reduced models. To the best of our knowledge, none of these approaches addressed the problem of unevaluable points.

We conducted our investigations in the context of GADO [3, 12], a GA that was designed with the goal of being suitable for use in engineering design. It uses new operators and search control strategies suitable for the domains that typically arise in such applications. GADO has been applied in a variety of optimization tasks that span many fields. It demonstrated a great deal of robustness and efficiency relative to competing methods.

In GADO, each individual in the GA population represents a parametric description of an artifact, such as an aircraft or a missile. All parameters take on values in known continuous ranges. The fitness of each individual is based on the sum of a proper measure of merit computed by a simulator or some analysis code (such as the takeoff mass of an aircraft), and a penalty function if relevant (such as to impose limits on the permissible size of an aircraft). The penalty function consists of an adaptive penalty coefficient multiplied by the sum of all constraint violations if any. A steady state GA model is used, in which operators are applied to two parents selected from the elements of the population via a rank based selection scheme, one offspring point is produced, then an existing point in the population is replaced by the newly generated point via a crowding replacement strategy. Floating point representation is used. Several crossover and mutation operators are used, most of which were designed specifically for the target domain type. GADO also uses a search-control method [12] that saves time by avoiding the evaluation of points that are unlikely to correspond to good designs.

The remainder of this paper first presents brief descriptions of the compared methods for reduced model use for speedup. The paper then presents brief descriptions of the approximation methods used to form the reduced models. We then present a number of experiments concerning the use of these approximation methods on one realistic engineering design task and several engineering design benchmarks. We conclude the paper with a discussion of the results and future work.

2 Reduced-model-based speedup methods

We compare two methods for improving the GA's performance. One is the idea of informed operators presented in [7] and the other is a variation of the genetic engineering idea presented in [8]. The remainder of this section describes these two approaches in more detail.

2.1 Informed operators

The main idea of Informed Operators (IO) is to replace pure randomness with decisions that are guided by the reduced model. We replace the conventional GA operators such as initialization, mutation and crossover with four types of informed operators:

- **Informed initialization:** For generating an individual in the initial population we generate a number of uniformly random individuals in the design space and take the best according to the reduced model. The number of random individuals is a parameter of the method with a default value of 20.
- **Informed mutation:** To do mutation several random mutations are generated of the base point. Each random mutation is generated according to the regular method used in GADO [3] by randomly choosing from among several different mutation operators and then randomly selecting the proper parameters for the mutation method. The mutation that appears best according to the reduced model is returned as the result of the mutation. The number of random mutations is a parameter of the method with a default value of five.

- **Informed crossover:** To do crossover two parents are selected at random according to the usual selection strategy in GADO. These two parents will not change in the course of the informed crossover operation. Several crossovers are conducted by randomly selecting a crossover method, randomly selecting its internal parameters and applying it to the two parents to generate a potential child. The internal parameters depend on the crossover method selected. For example to do point crossover the cut-and-paste point has to be selected. Informed mutation is applied to every potential child, and the best among the best mutations is the outcome of the informed crossover. The number of random crossovers is a parameter of the method with a default value of four. Thus each crossover-mutation combination uses 20 reduced model evaluations.

- **Informed guided crossover:** Guided crossover [6] is a novel operator used in GADO to replace some of the regular crossover-mutation operations to improve convergence towards the end of the optimization. Guided crossover does not involve mutation so we treat it differently. The way informed guided crossover works is a follows:

  - Several candidates are selected at random using the usual selection strategy of GADO to be the first parent for guided crossover. The number of such candidates is a parameter of the method with a default value of four.
  - For each potential first parent the second parent is selected in a fashion documented elsewhere [6]. Then several random points are generated from the guided crossover of the two parents and ranked using the reduced model. The number of such random points is a parameter of the method with a default value of five.
  - The best of the best of the random points generated is taken to be the result of the guided crossover.

  Thus the default total number of reduced model calls per informed guided crossover is 20.

2.2 Genetic Engineering

Our approach to Genetic Engineering (GE) attempts to improve the efficiency of the GA optimization by replacing some of the regular Darwinian iterations, which generate new individuals using crossover and/or mutation, with iteration in which individuals are generated by running a mini-optimization using the approximations and returning the best point found therein.

In our implementation of the GE approach, the non-gradient based Downhill-Simplex method is used - on a periodic basis - to generate new individuals instead of the traditional genetic operators. The Downhill-Simplex technique [13], is used because of its lesser commitment to the accuracy of the approximation function in comparison to the various gradient-based techniques. This method requires only function evaluations, not derivatives. Once in every four GA iterations we use the Genetic Engineering approach to create a new individual instead of the Darwinian genetic operators. Specifically, assuming the dimension of the parameter space is ndim, we select ndim+1 individuals from the current population using the regular selection strategy of GADO. We then use the cheap fitness function to get the approximate fitness values of these ndim+1 individuals. These ndim+1 individuals and their fitnesses serve as input to the Downhill-Simplex function. Based on the approximated fitnesses, new hypothesized minimum can be found by calling the Downhill-Simplex function. This minimum serves as the new born individual. Therefore, new individuals are created and evaluated by the true fitness function, so that the overall GA search proceeds.

It was hard to set the number of calls to the cheap fitness function with the GE as each mini-optimization could take a different number of iterations. We set the maximum number of calls during each mini-optimization to 500. However, we found that the GE approach ended up calling the cheap fitness function more than an order of magnitude more times than the IO approach. We did not repeat the experiments because the final conclusion was in favour the IO approach anyway.

3 Reduced model formation methods

In this section we briefly describe the methods used to create the approximate models which the speedup methods use. More detailed descriptions of these methods can be found in [14, 15].

3.1 General framework

We conducted our investigation in the context of the framework described in detail in [14]. We provide only a brief description here. The method is based on maintaining a large sample of the points encountered in the course of the optimization. Ideally, the sample should include all the points but a smaller sample may be used to keep the overhead reasonable.

Incremental approximate clustering We keep the sample divided into clusters. Starting with one cluster, we introduce one more cluster every specific number of iterations. The reason we introduce the clusters incrementally rather than from the beginning is that this way results in more uniform sized clusters. Every new point entering the sample, either becomes a new cluster (if it is time to introduce a cluster) or joins one of the existing clusters. A point belongs to the cluster whose center is closest in Euclidean distance to the point at the time in which the point joined the sample. We use clustering because it makes it possible to fit discontinuous and complicated surfaces with simpler surfaces such as quadratic approximations.

Approximate evaluation of new points The first step in evaluating the approximate fitness of a new point is to find to which cluster it belongs. If the point belongs to a cluster with cluster approximation functions, these are to be used, otherwise the global approximation functions are to be used. The evaluation method depends on the stage of the optimization. In the first half of the optimization the fitness is formed by using the approximate measure of merit and the approximate sum of constraints (which is forced to zero if negative). No attempt is made to guess at whether the point will be feasible, infeasible or unevaluable. In the second half of the optimization we use a two phase approach. First we use the nearest neighbors of the new point to guess whether the point is likely to be feasible, infeasible-evaluable or unevaluable. Based on this guess, and the point's cluster, we then use the proper approximation functions (for example, no approximation functions are used if the point is guessed to be unevaluable).

3.2 Quadratic Least Squares approximations

The first approach we used for forming the approximations was Quadratic Least Squares (LS). We distinguish between the approximation functions for the measure of merit and those for the sum of constraints.[1] The reason is that the constraints are only defined for infeasible designs. For feasible designs we have to put all the constraints at zero level as the simulators only return that they are satisfied. We form two types of approximations for measure of merit and for the sum of constraint violations:

- **Global approximation functions**

  We maintain two global approximation functions which are based on all the **evaluable** points in the sample.

  We use quadratic approximation functions of the form:

$$\hat{F}(\bar{X}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1,j=i}^{n,n} a_{ij} x_i x_j$$

  where n is the dimension of the search space and x_i is design variable number i. We use a least square fitting routine from [13] which works by using the normal equations method to fit the a_i values.

- **Cluster approximation functions**

  We use the same techniques for forming cluster approximation functions, except that we only form them for clusters which have a sufficient number of evaluable points.

3.3 Quickprop Neural Networks

Quickprop is a modification of the back-propagation learning algorithm (Backprop) that uses a second-order weight-update function, based on measurement of the error gradient at two successive points, to accelerate the convergence over simple first-order gradient descent [5]. Quickprop learns much faster than the standard back-propagation but also has more parameters that have to be fine-tuned. In this work, we used the C version [2] of Quickprop, translated by Terry Regier from Fahlman's original Lisp version.

There are two measures to be approximated, the measure of merit and the constraint violations, and therefore, the network structure could be either one-network-two-output, or two-network-one-output. After examining the two approaches, we have found that the two-network-one-output approach is better. We also introduced a mechanism for avoiding over-training in this network.

The Quickprop algorithm was implemented and integrated into GADO so as to generate both the global and the cluster approximation models. We form two types of approximations for measure of merit and constraint violations by maintaining both a global ANN and an ANN for each big enough cluster (i.e., cluster with a sufficient number of evaluable points) in a manner similar to that used for LS approximation.

4 Experimental results

To compare the performance of the different speedup methods we compared GADO with reduced-model-

[1] Since GADO only deals with the sum of all constraint violations rather than the individual constraints, we only form approximations for the sum of all constraint violations.

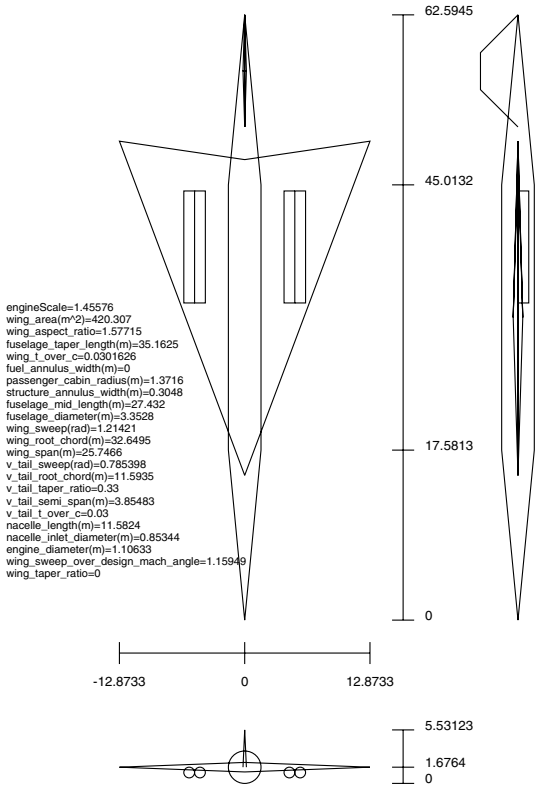


Figure 1: Supersonic transport aircraft designed by our system (dimensions in feet)

based informed operators to GADO with genetic engineering. We did the comparisons in the context of LS as well as QP based approximation methods. We also compare all of these to GADO without speedup altogether. We compared the five systems in several domains: one domain from real tasks in aerodynamic design, plus seven others from an existing set of engineering design benchmarks [1].

4.1 Application domain 1: Supersonic transport aircraft design domain

4.1.1 Domain description

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail in [16]. Figure 1 shows a diagram of a typical airplane automatically designed by our software system. The GA attempts to find a good design for a particular mission by varying twelve of the aircraft conceptual design parameters over a continuous range of values. An optimizer evaluates candidate designs using a multidisciplinary simulator. In our current implementation, the optimizer's goal is to minimize the takeoff mass of the aircraft, a

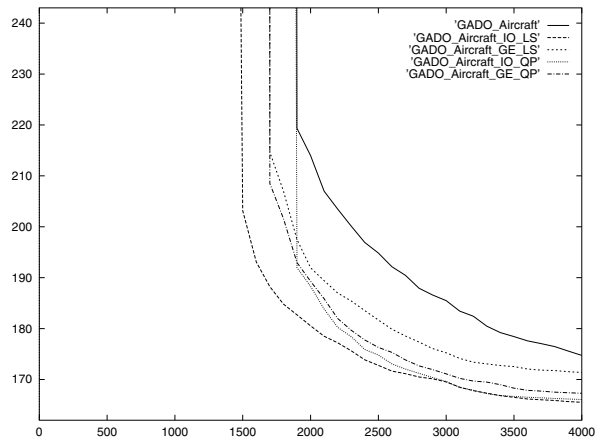


Figure 2: Comparison of average performance in application domain 1 (aircraft design)

measure of merit commonly used in the aircraft industry at the conceptual design stage. Takeoff mass is the sum of fuel mass, which provides a rough approximation of the operating cost of the aircraft, and "dry" mass, which provides a rough approximation of the cost of building the aircraft. In summary, the problem has 12 parameters and 37 inequality constraints. 0.6% of the search space is evaluable. No statistics exist regarding the fraction of the search space that is feasible because it is extremely small.

4.1.2 Experiments and results

Figure 2 shows the performance comparison in domain 1 (aircraft design). Each curve in the figure shows the average of 15 runs of GADO starting from random initial populations. The experiments were done once for each speedup method-approximation method combination (i.e. IO with LS,GE with LS, IO with QP and GE with QP) in addition to once without any speedup or approximation methods, with all other parameters kept the same. Figure 2 demonstrates the performance with each of the four combinations as well as the performance with no approximation or speedup at all (the solid line) in domain 1 . The figure plots the average (over the 15 runs) of the best measure of merit found so far in the optimization as a function of the number of iterations. (From now on we use the term "iteration" to denote an actual evaluation of the objective function, which is usually a call to a simulator or an analysis code. This is consistent with our goal of understanding how the speedup methods affect the number of calls to the objective function in problems where the informed operators or genetic engineering overhead is minuscule compared to the runtime of each objective function evaluation, as was the case here. This

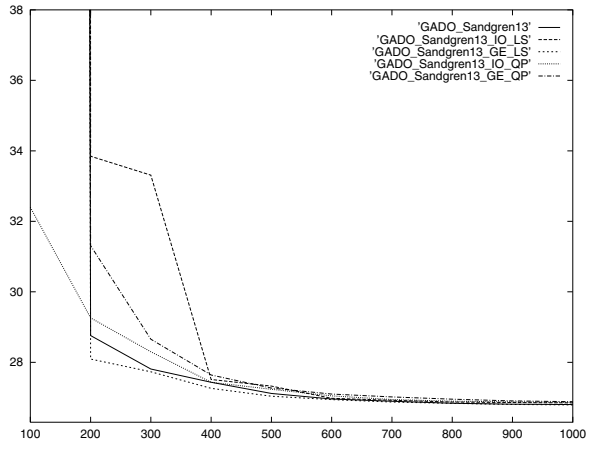


Figure 3: Comparison of average performance in benchmark domain 1

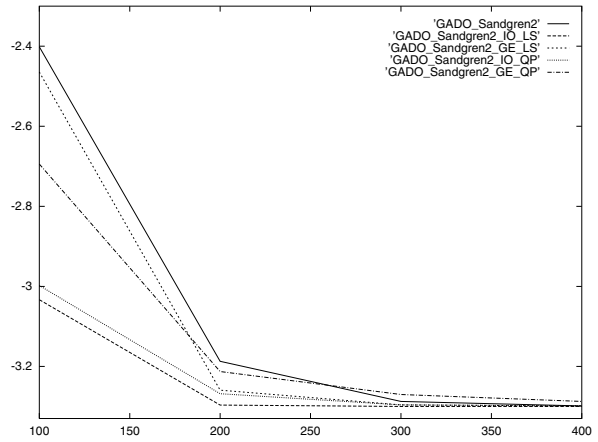


Figure 4: Comparison of average performance in benchmark domain 2

also helps us avoid the pitfalls of basing evaluations on run times, which can vary widely — for example across platforms and even across runs due to variations in memory available and hence caching effects.). The figure shows that the informed operators method improved the performance more than the genetic engineering method in most stages of the search regardless of which approximation method was used for forming the reduced models.

4.2 Benchmark engineering design domains

In order to further compare the two speedup methods, we compared their performance in several benchmark engineering design domains. These domains were introduced by Eric Sandgren in his Ph.D. thesis [1] in which he applied 35 nonlinear optimization algorithms to 30 engineering design optimization problems and compared their performance. Those problems have become used in engineering design optimization domains as benchmarks [17]. A detailed description of these domains is given in [1].

For each problem GADO was run 15 times using different random starting populations. As with the aircraft domain, the experiments were done once for each speedup method and approximation method combination, in addition to once without any speedup, with all other parameters kept the same. Figure 3 through Figure 9 show the performance with each of the four combinations as well as performance with no approximation or speedup at all (the solid lines) in the benchmark domains. Each curve in each figure shows the average of 15 runs of GADO with different random seeds. We found that in the first four benchmarks, which represent relatively easy optimization tasks, the

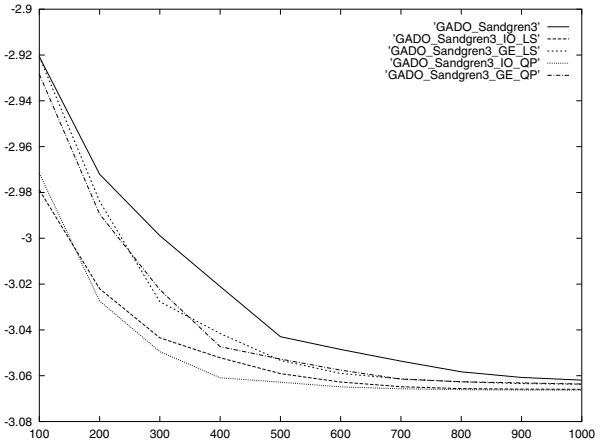


Figure 5: Comparison of average performance in benchmark domain 3

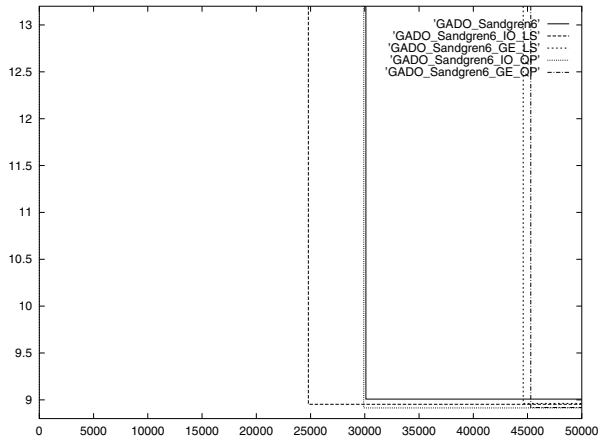


Figure 6: Comparison of average performance in benchmark domain 4

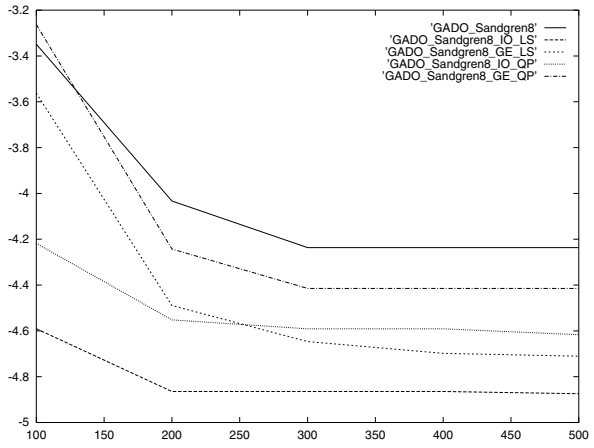


Figure 7: Comparison of average performance in benchmark domain 5

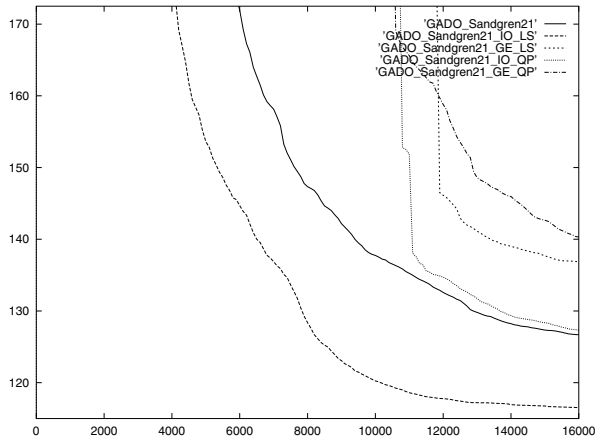


Figure 8: Comparison of average performance in benchmark domain 6

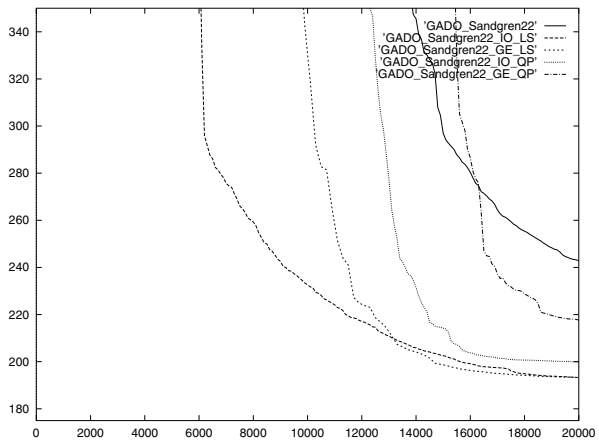


Figure 9: Comparison of average performance in benchmark domain 7

performance differences were small. The figures show that the IO based approach did better than the GE approach using the same approximation technique (LS or QP) in most stages of most domains. The figures also show that the IO method gave the best final performance in all domains. In fact, the results with the GE approach in benchmark 6 were worse than with no speedup at all. In benchmark 3 IO with QP was the winner while in all other benchmarks the IO with LS was the winner suggesting that LS was better than QP as an approximation method. We should also point out that in benchmark 7, in which GE appears to be doing better than IO for a segment of the optimization, we found that one of the runs did not find any feasible points but was slightly infeasible till the end. Thus, the GE performance in this domain is worse than the curve suggests.

5 Final Remarks

This paper has presented a comparison between two methods for using reduced models to speed up the search in GA-based engineering design optimization. Experiments were conducted in the domain of aircraft design optimization as well as several benchmark engineering design domains. The experiments show that the informed operators approach did consistently well and was better than the genetic engineering approach in all domains. Moreover, the genetic engineering approach called the approximate fitness function an order of magnitude more times than the informed operators approach. We believe that the reason for this result is that the reduced models used were not accurate enough for the genetic engineering approach to yield good results. The informed operators approach on the other hand makes a much weaker assumption about the accuracy of the reduced model (all it needs to speed up the optimization is that the reduced model be a better than random predictor of the actual model). The experiments also showed that using least squares approximations with any speedup approach usually yields better results than using the neural network approximations.

In the future, we intend to repeat the comparison of speedup approaches under different neural network models for approximation, such as radial-bases-function neural networks and multi-layer perceptrons. We also intend to explore ways of combining the informed-operator approach and the genetic engineering approach to achieve better performance than using any single approach. We also hope to be able to repeat the comparison in situations in which the reduced models are physical, pre-existent or somehow

more accurate but unfortunately we do not have access to such domains at this time. Finally we intend to explore other speedup approaches such as methods based on the formation and instantiation of statistical models.

Acknowledgments

This research was funded in part by a sub-contract from the Rutgers-based Self Adaptive Software project supported by the Advanced Research Projects Agency of the Department of Defense and by NASA under grant NAG2-1234.

References

[1] Eric Sandgren. The utility of nonlinear programming algorithms. Technical report, Purdue University, 1977. Ph.D. Thesis.

[2] Ian Patterson and Steve Readett. The quickprop algorithm project, 1998.

[3] Khaled Rasheed. GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, NJ, January 1998. Ph.D. Thesis, http://www.cs.rutgers.edu/~krasheed/thesis.ps.

[4] B. Dunham, D. Fridshal, R. Fridshal, and J. North. Design by natural selection. *Synthese*, 15:254–259, 1963.

[5] Scott E. Fahlmann. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.

[6] Khaled Rasheed. Guided crossover: A new operator for genetic algorithm based optimization. In *Proceedings of the Congress on Evolutionary Computation*, 1999.

[7] Khaled Rasheed and Haym Hirsh. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2000.

[8] Guido Cervone, Kenneth Kaufman, and Ryszard Michalski. Experimental validations of the learnable evolution model. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1064–1071, 6-9 July 2000.

[9] Vassili V. Toropov and Luis F. Alvarez. Application of genetic programming to the choice of a structure of global approximations. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 1998.

[10] D. Eby, R. Averill, W. Punch, and E. Goodman. Evaluation of injection island GA performance pn flywheel design optimization. In *Proceedings of the third Conference on adaptive computing in design and manufactoring*, 1998.

[11] Mohammed A. El-Beltagy, Prasanth B. Nair, and Andy J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 13-17 July 1999.

[12] Khaled Rasheed and Haym Hirsh. Learning to be selective in genetic-algorithm-based design optimization. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, 13:157–169, 1999.

[13] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : the Art of Scientific Computing*. Cambridge University Press, Cambridge [England] ; New York, 2nd edition, 1992.

[14] Khaled Rasheed. An incremental-approximate-clustering approach for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, 2000.

[15] Khaled Rasheed, Xiao Ni, and Swaroop Vattam. Comparison of methods for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, 2002.

[16] Andrew Gelsey, M. Schwabacher, and Don Smith. Using modeling knowledge to guide design space search. In *Fourth International Conference on Artificial Intelligence in Design '96*, 1996.

[17] D. Powell and M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431. Morgan Kaufmann, July 1993.

A Genetic Algorithm for Discovering Interesting Fuzzy Prediction Rules: applications to science and technology data

Wesley Romão
UEM, DIN-CTC
Av. Colombo, 5790
Maringá – PR.
87020-900 - Brazil
wesley@din.uem.br
http://www.din.uem.br/~wesley
(55) (44) 263-2479

Alex A. Freitas
PUC-PR, PPGIA-CCET
Rua Imaculada Conceição, 1155
Curitiba – PR.
80215-901 – Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex
(55) (41) 330-1347

Roberto C. S. Pacheco
UFSC, PPGEP-CTC
C. P. 476
Florianópolis, SC.
88040-900 – Brazil
pacheco@eps.ufsc.br
(55) (48) 331-7016

Abstract

Data mining consists of extracting knowledge from data. This paper addresses the discovery of knowledge in the form of prediction IF-THEN rules, which are a popular form of knowledge representation in data mining. In this context, we propose a new Genetic Algorithm (GA) designed specifically for discovering interesting fuzzy prediction rules. The GA searches for prediction rules that are interesting in the sense of being surprising for the user. More precisely, a prediction rule is considered interesting (or surprising) to the extent that it represents knowledge that not only was previously unknown by the user but also contradicts the original believes of the user. In addition, the use of fuzzy logic helps to improve the comprehensibility of the rules discovered by the GA, due to the use of linguistic terms that are natural for the user. The proposed GA is applied to a real-world science & technology data set, containing data about the scientific production of researchers. Experiments were performed to evaluate both the predictive accuracy and the degree of interestingness (or surprisingness) of the rules discovered by the GA, and the results were found to be satisfactory.

1 INTRODUCTION

The basic idea of data mining consists of extracting knowledge from data (Fayyad, 1996), (Han & Kamber, 2000). In this paper we address one general kind of data mining task, which we will refer to as the discovery of prediction rules. By prediction rule we mean an IF-THEN rule of the form:

IF <some_conditions_are_satisfied>
THEN <predict_the_value_of_some_goal_attribute> .

Hence, we aim at discovering rules whose consequent (THEN part) predict the value of some goal attribute for an example (a record of a data set) that satisfies all the conditions in the antecedent (IF part) of the rule. We assume there is a small set of goal attributes whose value is to be predicted. The goal attributes are chosen by the user, according to his/her interest and need.

It should be noted that this task can be regarded as a generalization of the well-known classification task of data mining. In classification there is a single goal attribute to be predicted, whereas we allow more than one goal attribute to be defined by the user.

Note that, although there are several goal attributes to be predicted, each rule predicts the value of a single goal attribute in its consequent. However, different rules can predict different values of different goal attributes.

In this paper we propose a new Genetic Algorithm (GA) designed specifically for discovering interesting fuzzy prediction rules. The main motivation for using a GA in prediction-rule discovery is that GAs, due to their ability to perform a global search, tend to cope better with attribute interaction than most greedy rule induction algorithms that are traditionally used in prediction-rule discovery (Dhar et al., 2000), (Freitas, 2001).

The justification for the "interesting" and "fuzzy" characteristics of the rules is as follows. In general, fuzzy logic is a flexible way of coping with uncertainties typically found in real-world applications. In particular, in the context of data mining, fuzzy logic seems a natural way of coping with continuous (real-valued) attributes. Using fuzzy linguistic terms, such as *low* or *high*, one can more naturally represent rule conditions involving continuous attributes, by comparison with crisp discretization of those attributes. For instance, the fuzzy condition "*Salary = low*" seems more natural for a user than the crisp condition "Salary < *$14,328.53*".

Although we do use fuzzy logic to improve the comprehensibility of the rules discovered by the GA, the focus of this paper is not on the use of fuzzy logic, but rather on the discovery of "interesting" rules. We emphasize that this is a difficult problem, relatively little explored in the literature. Most algorithms for discovering prediction rules focus on evaluating the predictive accuracy of the discovered rules (Hand, 1997), without trying to discover rules that are truly interesting for the user.

It should be noted that a rule can have a high predictive accuracy but be uninteresting for the user, because it represents some obvious or previously-known piece of knowledge. A classic example is the rule:

IF <patient is pregnant> THEN <patient is female>.

Hence, a major contribution of this paper is to propose a GA that searches for rules that not only have a high predictive accuracy but also are interesting, in the sense of being surprising (representing novel knowledge) for the user. As will be seen later, the core of the GA consists of an elaborate fitness function which takes both these aspects of rule quality into account.

Another contribution of this paper is that we apply the proposed GA to the mining of a real-world science & technology data set, containing data about the scientific production of researchers (cientometric data).

The remainder of this paper is organized as follows. Section 2 reviews relevant related work. Section 3 describes in detail the proposed GA for discovering interesting (surprising) fuzzy prediction rules. Section 4 reports the results of computational experiments. Finally, section 5 concludes the paper.

2 RELATED WORK

2.1 EAs FOR DISCOVERING FUZZY PREDICTION RULES

There has been very extensive research on evolutionary algorithms (EAs) for discovering fuzzy prediction rules. Roughly speaking, the algorithms can be divided into two broad groups:

(a) EAs evolving one or more aspects of membership functions, such as the number of membership functions (linguistic terms) for each attribute, the shape of the membership functions, etc. (Xiong & Litz, 1999), (Mota et al. 1999), (Mendes et al., 2001);

(b) EAs using user-defined membership functions, and evolving only the combinations of attribute values considered relevant for predicting a goal attribute (Ishibuchi & Nakashima, 1999), (Walter & Mohan, 2000).

We follow the later approach, due to two mains reasons. First, it allows us to incorporate the domain knowledge of the user into the specification of the membership functions, leading to membership functions which are more comprehensible for the user. This is important in our data mining application, where the discovered prediction rules are directly interpreted by a human decision maker. Second, it considerably reduces the search space, since the GA has to search only for combinations of attribute values to be included in the rules.

It should be noted the above-mentioned projects focus on the discovery of fuzzy rules with high predictive accuracy, without trying to discover surprising rules. Our work differs from these projects in that the proposed GA searches for fuzzy prediction rules that are not only accurate but also surprising for the user, representing knowledge that was previously unknown by the user, as will be seen later.

2.2 DISCOVERING INTERESTING PREDICTION RULES

There are two broad approaches for discovering interesting rules in data mining, namely the objective approach and the subjective approach. In general, the objective approach uses a rule-discovery method and a rule-quality measure that are independent of the user and the application domain (Major & Mangano, 1993), (Noda et. al, 1999).

By contrast, the subjective approach uses a rule-discovery method and/or a rule-quality measure that take into account the background knowledge of the user about the application domain (Silberchatz & Tuzhilin, 1996), (Liu & Hsu, 1996), (Liu et al., 1997).

Hence, in general the objective approach has more generality and autonomy than the subjective approach, whereas the subjective approach has the important advantage of using the user's background knowledge to guide the search for rules. Therefore, if the application domain is well-defined and a user who is an expert in the application domain is available, it makes sense to use the subjective approach. This is the case of the project reported in this paper. The proposed GA was developed with the primary goal of mining science & technology data, a well-defined application domain, and a user expert in this application domain was available. Therefore, in this paper we follow the subjective approach.

Out of the above-mentioned projects, there are two that are more related to our research. The first one is the work of (Liu & Hsu, 1996), (Liu et al., 1997). This work follows the subjective approach. It proposes the use of general impressions to guide the search for interesting rules. General impressions can be thought of as "rules" specified by the user, representing the background knowledge and believes of the user about the application domain. (General impressions will be discussed in more detail later.) Liu and his colleagues propose the use of general impressions as the basis for a post-processing method to select the most interesting rules, among all discovered rules. That is, first a data mining algorithm is run, discovering a potentially large number of rules. Then the discovered rules are matched against the user-specified general impressions, in order to select the most interesting rules.

Our work also uses the idea of user-specified general impressions to discover interesting rules. However, it differs from the above work in that we use general impressions directly in the search for rules, rather than as a post-processing method. In other words, instead of first generating a large number of rules and then selecting the most interesting ones, the set of general impressions is directly used by the data mining algorithm to generate only interesting rules, avoiding the unnecessary generation of many rules that will be later discarded due to their lack of interestingness for the user. In addition, we propose a GA for discovering interesting rules, whereas the work of Liu and his colleagues does not use any evolutionary algorithm.

The second work related to our research is the GA for discovering interesting rules proposed by (Noda et al., 1999). This GA also searches for rules that are both accurate and interesting, according to a certain rule-interestingness measure. However, our work differs from Noda et al.'s work in two major points. First, unlike their GA, our GA discovers fuzzy rules. Second, Noda et al. follow an objective approach for the discovery of interesting rules, whereas our GA follows a subjective

approach based on user-specified general impressions, as mentioned above.

3 A NEW GA FOR DISCOVERING INTERESTING (SURPRISING) FUZZY PREDICTION RULES

In this section we propose a new GA for discovering interesting (surprising) fuzzy rules. Hence, each individual represents a prediction rule. More precisely, each individual represents the antecedent (IF part) of a prediction rule. The consequent (THEN part) of the rule is not encoded in the genome. Rather, it is fixed for a given GA run, so that in each run all the individuals represent rules with the same consequent (value predicted for a goal attribute). Therefore, in order to discover rules predicting different goal attribute values, we need to run the GA several times, once for each value of each goal attribute.

Furthermore, the prediction rules represented by the individuals are fuzzy rules. We stress that only the rule antecedents are fuzzified. Rule consequents are always crisp. Concerning the rule antecedent, of course only conditions involving continuous (real-valued) attributes are fuzzified. Categorical (nominal) attributes are inherently crisp. For instance, there is no need to fuzzify a rule condition such as "*Sex = female*".

3.1 INDIVIDUAL REPRESENTATION

The genome of an individual represents a conjunction of conditions specifying a rule antecedent. Each condition is represented by a gene, and it consists of an attribute-value pair of the form $A_i = V_{ij}$, where A_i is the i-th attribute and V_{ij} is the j-th value belonging to the domain of A_i. In order to simplify the encoding of conditions in the genome, we use a positional encoding, where the i-th condition is encoded in the i-th gene. Therefore, we need to represent only the value V_{ij} of the i-th condition in the genome, since the attribute of the i-th condition is implicitly determined by the position of the gene. In addition, each gene also contains a boolean flag (B_i) that indicates whether or not the i-th condition is present in the rule antecedent. Hence, although all individuals have the same genome length, different individuals represent rules of different lengths (which is, of course, desirable in prediction rules, since one does not know a priori how many conditions will be necessary to create a good prediction rule). The structure of the genome of an individual is illustrated in Figure 1, where m is the number of attributes of the data being mined.

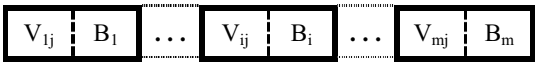


Figure 1: Genome of an individual representing a rule antecedent

We emphasize that the operator "=" is used for both fuzzy conditions and crisp conditions, as follows. As usual in the data mining and machine learning literature, our GA can cope with two kinds of attributes: continuous (real-valued) attributes and categorical (nominal) ones. Categorical attributes are inherently crisp, so that they are associated with crisp conditions such as "*Sex = female*". Continuous attributes are fuzzified, so that they are associated with fuzzy conditions such as "*Age = low*", where *low* is a fuzzy linguistic term.

3.2 FUZZIFYING CONTINUOUS ATTRIBUTES

Recall that, as discussed in section 2, in this work the GA uses user-defined membership functions. Hence, it evolves the combinations of attribute values considered relevant for predicting a goal attribute, but there is no need to evolve the membership functions.

In our GA the fuzzification of continuous attributes is performed as follows. Each continuous attribute is associated with either two or three linguistic terms (corresponding to the "values" of the fuzzified attribute), namely either {*low*, *high*} or {*low*, *medium*, *high*}. Each of these linguistic terms is defined by a user-specified membership function. These functions have a trapezoidal format, where there are three (or two) linguistic terms.

3.3 FITNESS FUNCTION

Recall that each individual is associated with a fuzzy prediction rule. In the vast majority of the literature, the main criterion used to evaluate the quality of a fuzzy prediction rule is predictive accuracy. This criterion is also important in our application, but it is not the only one. As discussed in the Introduction, a prediction rule can be accurate but not interesting for the user. This will be the case when the rule represents some relationship in the data that was already known by the user. To avoid this, our fitness function takes into account two criteria:

(a) The predictive accuracy of the rule;

(b) A measure of the degree of interestingness (or surprisingness) of the rule.

With respect to the latter criterion, our GA favors the discovery of rules that are explicitly surprising for the user, as will be seen later.

These two criteria are combined into a weighted formula as follows:

$$\text{Fitness}(i) = \text{Acc}(i) * \text{Surp}(i)$$

The measures of Acc(i) and Surp(i) are described in the next two subsections, respectively, since they are computed by separated elaborate procedures.

3.3.1 Measuring the Predictive Accuracy of a Fuzzy Rule

The first step to measure the predictive accuracy of a fuzzy rule is to compute the degree to which an example belongs to a rule antecedent. Recall that the rule antecedent consists of a conjunction of conditions. We use the standard fuzzy AND operator, where the degree of membership of an example to a rule antecedent is given by:

$$\min_{i=1}^{z}(\mu_i)$$

where μ_i denotes the degree to which the example belongs to the i-th condition of the rule antecedent, z is the number of conditions in the rule antecedent, and min is the minimum operator. The degree to which the example

belongs to the i-th condition is directly determined by the value of the corresponding membership function for the example's attribute value associated with that condition. Of course, crisp conditions can have only either 0 or 1 membership degrees.

For instance, consider a rule antecedent with the following two rule conditions: (*Age* = *low*) AND (*Sex* = *female*), where the first condition is fuzzy and the second one is crisp. Suppose that a given example has the values *23* and *female* for the attributes *Age* and *Sex*, respectively. Suppose also that the membership function for the *low* linguistic term of *Age* returns the value 0.8 for the value *23*. Then the degree to which this example belongs to this rule antecedent is min(0.8,1.0) = 0.8.

Let A be the antecedent of a given rule. Once the degree to which each example belongs to A has been computed, the predictive accuracy of the i-th individual (fuzzy rule), denoted Acc(i), is computed by the formula:

$$Acc(i) = (CorrPred - 1/2) / (TotPred)$$

where CorrPred (number of correct predictions) is the summation of the degrees of membership in A for all examples that have the value V_{ij} predicted by the rule and TotPred (total number of predictions) is the summation of the degrees of membership in A for all examples. This formula is essentially a fuzzy version of a crisp measure of predictive accuracy used by some data mining algorithms (Quinlan, 1987), (Noda et al., 1999). The rationale for subtracting 1/2 from CorrPred in the numerator is to penalize rules that are too specific, which are probably overfitted to the data. For instance, suppose CorrPred = 1 and TotPred = 1. Without subtracting 1/2 from CorrPred the modified formula would return a predictive accuracy of 100% for the rule, which intuitively is an over-optimistic estimate of predictive accuracy in this case. However, subtracting 1/2 from CorrPred the above formula returns 50%, which seems a more plausible estimate of predictive accuracy, given that the rule is too specific. Clearly, for large values of CorrPred and TotPred the subtraction of 1/2 will not have a significant influence in the value returned by the formula, so that this subtraction penalizes only rules which are very specific, covering just a few examples.

3.3.2 Measuring the Degree of Surprisingness of a Prediction Rule

We consider a prediction rule interesting to the extent that it is surprising for the user, in the sense of representing knowledge that not only was previously unknown but also contradicts the original believes of the user. Clearly, the problem of discovering surprising rules is a very difficult one, which has been relatively little investigated in the data mining literature. (As mentioned above, the vast majority of the literature focus on the discovery of rules with a high predictive accuracy, without trying to measure how novel or surprising the rule is for the user.)

In order to tackle this problem we follow a subjective approach for discovering surprising rules, based on the use of user-specified general impressions (Liu & Hsu, 1996), (Liu et al., 1997). In essence, a general impression specifies some relationship that the user believes to be true in the data being mined. General impressions, like prediction rules, are expressed in the form IF <conditions> THEN <predicted value>. The main difference is that general impressions are manually specified and represent believes of the user about relationships in the data, whereas prediction rules are automatically discovered and represent relationships that seem to hold in the data, according to the criteria used by the data mining algorithm. Therefore, the specification of general impressions assume that the user already has some previous knowledge or hypotheses about relationships that hold in the application domain - in our case, science and technology data.

Let $\{R_1,...,R_i,...R_{|R|}\}$ be the set of rules in the current population of the GA, where |R| denotes the number of rules (individuals); and let $\{GI_1,...,GI_j,...GI_{|GI|}\}$ be the set of general impressions representing the previous knowledge and believes of the user, where |GI| denotes the number of general impressions. Note that the set $\{GI_1,...,GI_j,...GI_{|GI|}\}$ is specified by the user before the GA starts to run, and it is kept fixed throughout the GA run. In order to compute the degrees of surprisingness of the rules in the current population, each rule is matched against every GI, as shown in Figure 2.

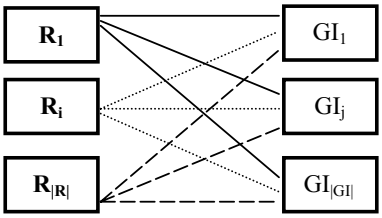


Figure 2: Matching between each rule and every general impression

A rule R_i is considered surprising, in the sense of contradicting a general impression GI_j of the user, to the extent that R_i and GI_j have similar antecedents and contradictory consequents. In other words, the larger the similarity of the antecedents of R_i and GI_j and the larger the degree of contradiction of the consequents of R_i and GI_j, the larger the degree of surprisingness of rule R_i with respect to general impression GI_j.

For each pair of rule R_i and GI_j - where i varies in the range 1,...,|R| and j varies in the range 1,...,|GI| - the GA computes the degree of surprisingness of R_i with respect to GI_j in three steps, as follows.

First step: finding the general impressions whose consequents are contradicted by the consequent of R_i. We say that the consequent of R_i contradicts the consequent of a general impression GI_j if and only if R_i and GI_j have the same goal attribute but a different goal attribute value in their consequent. For instance, this would be the case if R_i predicts "*production = low*" and GI_j predicts "*production = high*". Note that if R_i and GI_i predict different goal attributes, or if they predict the same value for the same goal attribute, there is no contradiction between them, and so the degree of surprisingness of R_i with respect to GI_i is considered zero, and in this case the second and third steps, described below, are ignored.

Second step: computing the similarity between the antecedents of R_i and GI_j. For each general impression GI_j

found in the previous step (i.e, each general impression GI_j contradicted by R_i), the system computes the similarity between the antecedents of R_i and GI_j. This similarity, denoted $AS_{(i,j)}$, is computed by the formula:

$$AS_{(i,j)} = |A_{(i,j)}| / \max(|R_i|,|GI_j|) ,$$

where $|R_i|$ is the number of conditions (attribute-value pairs) in rule R_i, $|GI_j|$ is the number of conditions in general impression GI_j, max is the maximum operator, and $|A_{(i,j)}|$ is the number of conditions that are exactly the same (i.e., have the same attribute and the same attribute value) in both R_i and GI_j. This formula is a somewhat simplified version of the formulas proposed by (Liu & Hsu, 1996) to measure the similarity between the antecedents of R_i and GI_j. Those authors proposed separate formulas to measure the similarity with respect to attributes and with respect to attribute values, whereas we have chosen to incorporate both aspects of antecedent similarity into a single formula, for the sake of simplicity.

Third step: computing the degree of surprisingness of R_i with respect to GI_j. Let Surp(i,j) denote the degree of surprisingness of R_i with respect to GI_j. Surp(i,j) depends on both $AS_{(i,j)}$, computed in the second step, and on the difference between the rule consequents of R_i and GI_j, computed in the first step, as follows. The goal attribute values in the consequents of R_i and GI_j can be either a value in the set {*low*, *high*} or a value {*low*, *medium*, *high*}, depending on the goal attribute. (The choice between these two attribute domains is made by the user for each goal attribute, as will be seen later.) If the difference between the consequents of R_i and GI_j is that one of them is *low* and the other one is *high*, characterizing the greatest possible difference between those consequents, then Surp(i,j) is assigned the value of $AS_{(i,j)}$, without any modification. If the difference between the consequents of R_i and GI_j is that one of them is *medium* and the other one is either *low* or *high*, characterizing a smaller difference between those consequents, then Surp(i,j) is assigned half the value of $AS_{(i,j)}$, i.e. Surp(i,j) = 0.5 x $AS_{(i,j)}$. In the latter case Surp(i,j) is assigned a smaller value than in the former case to reflect the fact that the degree of contradiction is correspondingly smaller.

Finally, once the above three steps have been completed for all general impressions, with respect to a given rule R_i, the system has computed all the degrees of surprisingness of R_i with respect to every general impression GI_j, i.e. all Surp(i,j), j=1,...,$|GI|$, where $|GI|$ is the number of general impressions. At this point the degree of surprisingness of rule R_i, denoted Surp(i), is simply computed by the formula:

$$Surp(i) = \max_{j=1}^{|GI|} \left[AS_{(i,j)} \right]$$

where max returns the maximum value among its arguments.

3.4 SELECTION AND GENETIC OPERATORS

The GA uses tournament selection (Blickle, 2000), which essentially works as follows. First, k individuals are randomly picked (k = 2), with replacement, from the population. Then the individual with the best fitness value, out of the k individuals, is selected as the winner of the tournament. This process is repeated P times, where P is the population size. Next the P selected individuals undergo genetic operators, as follows.

The GA uses relatively simple crossover and mutation operators. It uses uniform crossover (Goldberg, 1989). There is a probability for applying crossover to a pair of individuals and another probability for swapping each corresponding pair of gene (attribute)'s value in the genome of two individuals. The crossover probabilities used were 0.85 for the crossover operator and 0.5 for attribute value swapping. Our choice of uniform crossover was motivated by the fact that this operator has no positional bias, i.e., the probability of swapping each pair of attribute values is independent of the position of that attribute value in the genome. This is desirable in our data mining application, where the rule antecedent represented by the genome consists of an unordered set of conditions.

The mutation operator randomly transforms the value of an attribute into another (different) value belonging to the domain of that attribute. The mutation probability used was 0.02.

In addition to crossover and mutation operators, the GA also uses operators that insert/remove conditions to/from a rule. In essence, the condition-insertion operator switches on the flag of some condition in the genome, rendering it present in the decoded rule antecedent. Conversely, the condition-removal operator switches off the flag of some condition in the genome, which effectively removes that condition from the decoded rule antecedent. The condition-insertion and condition-removal operators perform specialization and generalization operations in the rule, respectively. Hence, they contribute for a broader exploration of the search space, facilitating the exploration of some regions of the search space that might not be so easily accessible to crossover and mutation operators.

4 COMPUTATIONAL RESULTS

We now report the results of computational experiments performed with the GA proposed in the previous section. In these experiments the set of general impressions was specified by the Head of Research of the State University of Maringá (Brazil). The same user also evaluated the interestingness of the rules discovered by the GA, as will be seen later. The data set used in our experiments is described in section 4.1.

The rules discovered by the GA were evaluated with respect to two criteria, namely:

(a) Predictive accuracy. As usual in the literature, predictive accuracy was measured in an objective way, by computing the prediction accuracy rate on a test set separate from the training set. The results with respect to predictive accuracy are reported in section 4.2.

(b) *Degree of interestingness (surprisingness)*. This is a measure of how surprising, novel the rule is for the user, as explained in the previous section. This was measured in a subjective way, by showing the discovered rules to the user and ask him to assess them according to how interesting they were. The results with respect to interestingness are reported in section 4.3.

4.1 THE DATA SET

The application domain addressed in this paper involves a science and technology database obtained from CNPq (the Brazilian government's National Council of Scientific and Technological Development). More precisely, we have mined a subset of the database containing data about the scientific production of researchers of the south region of Brazil. However, it should be noted that the design of the GA is generic enough to allow its use in virtually any other application domain, as long as proper general impressions and membership functions are specified by the user.

The experiments reported in this paper have been performed with 24 attributes. The selection and preparation of these attributes for data mining purposes was a time-consuming process, taking several months, since the original data set was not collected for data mining purposes.

The data set contained 5,690 records (examples), and each record had attributes describing a given researcher and his scientific production in the period from 1997 to 1999. Records that had any attribute with missing value were removed. Out of the 24 attributes, 6 were used as goal attributes to be predicted, and the other 18 attributes were used as predictor attributes. Out of the 18 predictor attributes, 8 were categorical (nationality, continent of origin, sex, state, city, skill in writing English, whether or not she/he was the head of a research group, main research area) and 10 were continuous (educational level, No. of years since last graduation, age, No. of completed technical projects, No. of delivered courses, No. of supervised Ph.D. thesis, No. of supervised M.Sc. dissertations, No. of supervised research essays (at the diploma level), No. of supervised final-year undergraduate projects, No. of supervised undergraduate students with a research scholarship). The 10 continuous attributes were fuzzified for rule-discovery purposes, as previously explained.

For prediction purposes, each goal attribute was discretized into either two values (referring to a low or high scientific production) or three values (referring to a low, medium or high scientific production), as determined by the user.

The 6 goal attributes, denoted $G_1,...,G_6$, have the following meaning and values to be predicted:

G_1 = No. of papers published in national journals - values: low, medium, high;
G_2 = No. of papers published in internat. journals - values: low, medium, high;
G_3 = No. of chapters published in national books - values: low, medium, high;
G_4 = No. of chapters published in international books - values: low, high;
G_5 = No. of national edited/published books - values: low, high;
G_6 = No. of internat. edited/published books - values: low, high.

Therefore, in total there are 15 goal attribute values to be predicted.

4.2 EVALUATING THE PREDICTIVE ACCURACY OF DISCOVERED RULES

In order to measure the predictive accuracy of discovered rules, we have performed a well-known 10-fold cross-validation procedure (Hand, 1997). In essence, this procedure works as follows. First, the data set is divided into 10 mutually exclusive and exhaustive partitions. Then the data mining algorithm is run 10 times. In the i-th run, i=1,...,10, the i-th partition is used as the test set, and the remaining 9 partitions are temporarily grouped and used as the training set. In each run the system computes the prediction accuracy rate on the test set, which is the ratio of the number of correct predictions over the total number of predictions. The reported result is the average prediction accuracy rate over the 10 runs.

We have compared the predictive accuracy of the rules discovered by our GA with the predictive accuracy of the rules discovered by J4.8 (Witten, 2000). The latter is a decision-tree-building algorithm which is included in a public-domain data mining tool available at: www.cs.waikato.ac.nz/ml/weka/index.html. J4.8 is a modified version of the very well-known decision-tree-building algorithm C4.5 (Quinlan, 1993).

Note that J4.8 (as well as C4.5) is an algorithm designed for the classification task of data mining, where there is a single goal attribute to be predicted. Similarly, each run of our GA discovers a rule predicting a different goal attribute value. Hence, both J4.8 and our GA have to be run several times in our application, since we are interested in discovering rules predicting several goal attributes. More precisely, J4.8 was "run" 6 times (each "run" actually consists of the 10 runs of a 10-fold cross-validation procedure), whereas our GA was "run" 15 times (again, each "run" was a 10-fold cross-validation procedure), corresponding to the 15 different goal attribute values for all the 6 goal attributes.

Note also that J4.8 and our GA were designed for discovering different kinds of prediction rules. The two main differences are as follows. First, J4.8 just tries to discover accurate rules. It does not try to discover interesting, surprising rules. By contrast, our GA tries to discover rules that are both accurate and surprising for the user. Second, J4.8 was designed for discovering classification rules covering all examples. That is, given any test example, J4.8 must have discovered a rule that can be used to predict its class. By contrast, our GA does not try to discover rules covering all examples. It tries to discover only a small set of interesting, surprising rules, the knowledge "nuggets". The discovered rules can collectively cover only a relatively small subset of examples, and yet be considered surprising, high-quality rules. These two differences make it difficult to compare the two algorithms in a fair way.

In order to make this comparison more fair, we have eliminated the above first difference. This was achieved by modifying the fitness function of the GA (only in the experiments reported in this section) so that the fitness of an individual (rule) is measured only by its predictive accuracy, ignoring its degree of surprisingness, i.e.:

$$\text{Fitness}(i) = \text{Acc}(i) = (\text{CorrPred} - 1/2) / (\text{TotPred})$$

Now both J4.8 and the GA search only for accurate rules.

The above second difference between the two algorithms is more difficult to eliminate, and it still remains a difference in our experiments. This problem will be the subject of future research.

The predictive accuracy obtained by J4.8 and our GA is reported in Table 1. The first column of this table identifies the goal attribute predicted by the rule (see the meaning of G_1...G_6 in the previous section), whereas the second column identifies the value predicted for that goal attribute. The third column identifies the relative frequency (in %) of the corresponding goal attribute value in the training set. The fourth and fifth columns report the prediction accuracy rate (in %) in the test set (10-fold cross-validation) of J4.8 and the GA, respectively. In each row, we show in bold the larger predictive accuracy rate, out of the rates obtained by the two algorithms.

Table 1: Prediction Accuracy Rate (%) of J4.8 and GA

Goal attrib.	Predicted value	Freq. (%)	J4.8	GA
G_1	low	46.9	**64.9**	58.8
	medium	50.6	**63.9**	60.4
	high	2.5	**9.1**	0.0
G_2	low	64.2	76.6	**90.7**
	medium	29.7	**45.3**	40.0
	high	6.1	**32.2**	25.0
G_3	low	76.9	82.2	**95.2**
	medium	21.2	45.3	**56.7**
	high	1.9	**27.4**	25.0
G_4	low	93.2	93.4	**98.4**
	high	6.8	**51.7**	14.3
G_5	low	83.5	86.0	**89.5**
	high	16.5	54.7	**56.9**
G_6	low	97.9	97.9	**98.9**
	high	2.1	0.0	0.0

As can be seen in the table, the prediction accuracy rate of the GA is larger than the one of J4.8 in seven rows (i.e., seven goal attribute values), whereas the converse is true in other seven rows. With the exception of the goal attribute G_1, in general the GA outperformed J4.8 in the prediction of goal attribute values with a larger frequency in the training set, whereas J4.8 outperformed the GA in values with a smaller frequency in the training set.

In any case, the focus of our experiments is the evaluation of the degree of interestingness of the rules discovered by the GA, reported in the next section.

4.3 EVALUATING THE INTERESTINGNESS OF THE RULES DISCOVERED BY THE GA

The rules discovered by the GA were also evaluated with respect to their degree of interestingness (surprisingness) for the user. In this experiment it was not possible to compare the GA with J4.8, since J4.8 was not designed to discover interesting rules. Actually, for the majority of the 6 goal attributes, J4.8 produced a very large decision tree, with literally hundreds of nodes. Therefore, it was not even feasible to show all rules discovered by J4.8 to the user, anyway.

By contrast, the GA was explicitly designed to discover a small set of interesting rules (one rule per goal attribute value to be predicted), so that it was very feasible to show all rules discovered by the GA to the user, for his subjective evaluation.

We emphasize that the user who evaluated the interestingness of the discovered rules was the same user who specified the general impressions, as mentioned above. Actually, when the user was shown a discovered rule, he was also shown his own general impression contradicted by that rule.

The user was asked to assign to each rule discovered by the GA one of the following three degrees of interestingness (surprisingness): low interestingness, medium interestingness or high interestingness. The results of the evaluation performed by the user is reported in Table 2. The rule consequent in the first column consists of an attribute-value pair "G_i= *val*" identifying the goal attribute value predicted by the rule, where G_i denotes the *i*-th attribute, *i*=1,...,6 (see section 4.1 for the meaning of these goal attributes) and *val* denotes the value predicted for the corresponding goal attribute. The second column of this table shows the degree of interestingness assigned to the rule by the user.

Table 2: Interestingness of rules discovered by the GA

Rule consequent	interestingness for the user
G_1 = low	high
G_1 = medium	medium
G_1 = high	medium
G_2 = low	high
G_2 = medium	medium
G_2 = high	low
G_3 = low	high
G_3 = medium	low
G_3 = high	low
G_4 = low	medium
G_4 = high	medium
G_5 = low	high
G_5 = high	low
G_6 = low	high
G_6 = high	low

The experiment reported in this section, involving 15 runs of the GA (one for each goal attribute value being predicted) took about 6 minutes. Each run of the GA had a population size of 100 individuals, which evolved during 60 generations.

The results reported in Table 2 were obtained by using the entire data set (i.e., all the 5,690 examples) as input data for the GA. This procedure is justified because when measuring the degree of interestingness of discovered rules there is no need for dividing the data into training and test sets, since there is no need for measuring predictive accuracy in the test set (which was already measured in the experiments reported in the previous section).

Out of the 15 rules discovered by the GA, 5 were assigned a high degree of interestingness by the user, 5 were assigned a medium degree of interestingness, and the remaining 5 were assigned a low degree of interestingness. Overall, this seems to be a relatively good result, considering how difficult it is to discover very interesting, surprising rules.

We have observed that there is a relationship between a rule's simplicity (in the sense of having a small number of conditions) and its degree of interestingness for the user. This relationship is due to an interaction between the measure of rule surprisingness used in this work and the kind of general impressions specified by the user, as follows. In our experiments, the user specified mainly short general impressions, having a small number of conditions. As a result, the measure of rule surprisingness favors the discovery of short rules too, since these rules can have a larger degree of similarity between the rule antecedent and the general impression antecedent.

5 CONCLUSIONS AND FUTURE WORK

We have proposed a GA for discovering interesting fuzzy prediction rules. The proposed GA was evaluated with respect to both the predictive accuracy and the interestingness of the discovered rules. With respect to the former criterion, the performance of the GA was compared with J4.8, a well-known decision-tree-building algorithm. Overall, the GA was found to be competitive with J4.8 with respect to this criterion.

In any case, the main focus of our experiments was on the discovery of rules that are interesting, in the sense of representing surprising, previously-unknown knowledge for the user. In our experiments the application domain was science & technology data, and the user was an expert in this domain. Overall, the GA was able to found several rules that were considered very interesting by the user. For instance, one of the general impressions specified by the user represented his previous knowledge (or belief) that biology researchers of a given region had a high number of international edited/published books. However, the GA was able to found an accurate rule contradicting this general impression. The rule had the same antecedent as the general impression but made the opposite prediction, i.e. it predicted that the researchers in question had a low number of international edited/published books. This rule was considered very interesting by the user.

The main direction for future research will be to compare the degree of interestingness of the rules discovered by our GA with the degree of interestingness of the rules discovered by another data mining algorithm that was specifically designed for the discovery of interesting rules.

References

T. Blickle (2000). Tournament selection. In: T. Back, D. B. Fogel and Z. Michalewicz (Eds.) *Evolutionary Computation 1: Basic Algorithms and Operators*. Chapter 24. Institute of Phisics Publishing.

V. Dhar, D. Chou and F. Provost (2000). Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid With Entropy Reduction. *Data Mining and Knowledge Discovery 4(4)*, 251-280.

U. M. Fayyad, G. Piatetsky-Shapiro and P. Smyth (1996). *Advances in knowledge discovery & data mining*. Chapter 1: From data mining to knowledge discovery: an overview. AAAI/MIT.

A. A. Freitas (2001). Understanding the Crucial Role of Attribute Interaction in Data Mining. *Artificial Intelligence Review 16(3)*, Nov. 2001, pp. 177-199.

D. E. Goldberg (1989). *Genetic algorithms in search, optimization, and machine learning***.** New York: Addison-Wesley Publishing Company, Inc.

J. Han and M. Kamber (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.

D. J. Hand (1997). *Construction and Assessment of Classification Rules*. John Wiley & Sons.

H. Ishibuchi and T. Nakashima (1999). Designing Compact Fuzzy Rule-Based Systems with Default Hierarchies for Linguistic Approximation. *CEC-99*, p. 2341-2348.

B. Liu and W. Hsu (1996). Post-analysis of learned rules. *AAAI-96*, p. 828-834.

B. Liu, W. Hsu and S. Chen (1997). Using general impressions to analyze discovered classification rules. *Third Int. Conf. on Knowledge Discovery and Data Mining, KDD-97*, p. 31-36.

J. A. Major and J. J. Mangano (1993). Selecting among rules induced from a hurricane database. *Knowledge Discovery in Databases Workshop at AAAI-93*, p. 28-44.

R. R. F. Mendes, F. B. Voznika, A. A. Freitas and J. C. Nievola. (2001) Discovering fuzzy classification rules with genetic programming and co-evolution. Principles of Data Mining and Knowledge Discovery (Proc. 5th European Conf., PKDD 2001) - Lecture Notes in Artificial Intelligence 2168, pp. 314-325. Springer-Verlag.

C. Mota, H. Ferreira and A. Rosa (1999). Independent and Simultaneous Evolution of Fuzzy Sleep Classifiers by Genetic Algorithms. *GECCO-99*, p. 1622-1629.

E. Noda, A. A. Freitas and H. S. Lopes (1999). Discovering interesting prediction rules with a genetic algorithm. *Proc. Congress on Evolutionary Computation (CEC-99),* 1322-1329. Washington D.C., USA.

J. R. Quinlan (1987). Generating production rules from decision trees. *Proc. IJCAI-87*, p. 304-307.

J. R. Quinlan (1993). *C4.5: Programs for Machine Learning***.** Morgan Kaufmann.

A. Silberschatz and A. Tuzhilin (1996). What Makes Patterns Interesting in Knowledge Discovery Systems. *IEEE Transactions on Knowledge and data engineering*, Vol. 8, No. 6, pp. 970-974.

D. Walter and C. K. Mohan. (2000) ClaDia: a fuzzy classifier system for disease diagnosis. *Proc. Congress on Evolutionary Computation (CEC-2000)*. La Jolla, CA, USA.

I. H. Witten and E. Frank (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations***.** Morgan Kaufmann Publishers.

N. Xiong and L. Litz (1999). Generating Linguistic Fuzzy Rules for Pattern Classification with Genetic Algorithms. *PKDD-99*, p. 574-579.

Hyperspectral Image Analysis Using Genetic Programming

Brian J. Ross and Anthony G. Gualtieri
Brock University
Dept. of Computer Science
St.Catharines, ON, Canada L2S 3A1
bross@cosc.brocku.ca
(905)688-5550 ext. 4284

Frank Fueten
Brock University
Dept. of Earth Sciences
St.Catharines, ON, Canada L2S 3A1
ffueten@craton.geol.brocku.ca
(905)688-5550 ext. 3856

Paul Budkewitsch
Canada Centre for Remote Sensing
588 Booth Street
Ottawa, ON, Canada K1A 0Y7
paul.budkewitsch@ccrs.nrcan.gc.ca
(613)947-1331

Abstract

Genetic programming is used to evolve mineral identification functions for hyperspectral images. The input image set comprises 168 images from different wavelengths ranging from 428 nm (visible blue) to 2507 nm (invisible shortwave in the infrared), taken over Cuprite, Nevada, with the AVIRIS hyperspectral sensor. A composite mineral image indicating the overall reflectance percentage of three minerals (alunite, kaolnite, buddingtonite) is used as a reference or "solution" image. The training set is manually selected from this composite image. The task of the GP system is to evolve mineral identifiers, where each identifier is trained to identify one of the three mineral specimens. A number of different GP experiments were undertaken, which parameterized features such as thresholded mineral reflectance intensity and target GP language. The results are promising, especially for minerals with higher reflectance thresholds (more intense concentrations).

1 INTRODUCTION

Remote sensing using aircraft and satellite photography is well-established technology. The use of hyperspectral imagery, however, is relatively new. Hyperspectral images are capable of precisely capturing narrow bands of spectra through a wide range of wavelengths. Since many organic and inorganic materials exhibit unique absorption and reflection characteristics at particular bandwidths, these spectra are useful for remotely identifying various materials and phenomena of interest. This is an important area of work, since hyperspectral data permits the discovery of valuable natural resources in areas largely inaccessible by foot. Literally any area of the Earth can be mapped by hyperspectral imagery, be it with aircraft or satellites.

One complication in using this technology is the time and expertise required to interpret the data. Hyperspectral imaging systems such as the NASA/JPL AVIRIS[1] sensor can capture over 200 bandwidths for a single geographic location (Green *et al.* 1998). This is denoted by a hyperspectral cube, which takes the form of many hundreds of mega-bytes of information. Interpreting this massive amount of data is difficult, especially considering that the spectra obtained represent mixed spectral signatures of a variety of materials. Moreover, noise and other unwanted effects must be considered. Deciphering this enormous volume of cryptic data is therefore next to impossible for humans to do manually.

This paper uses genetic programming (GP) to evolve mineral classifiers for use on hyperspectral images. Separate mineral classifiers are evolved for three specific minerals – buddingtonite, alunite, and kaolinite. The classifiers take the form of programs which, when given a vector data from a particular pixel location on a hyperspectral cube, determine whether the mineral of interest resides there or not. Evolution proceeds by evaluating the performance of classifiers on positive and negative training sets. In addition, given the effects of noise at low reflectance levels, separate threshold stages are examined. This is done in the hopes that more accurate classification arises at higher reflectance levels, where there are more intense mineral concentrations.

Section 2 reviews concepts in hyperspectral imaging. The experimental design is outlined in Section 3. Section 4 presents the results of the experiments. A discussion and comparison to related work concludes the paper in Section 5.

[1] Airborne Visible/Infrared Imaging Spectrometer.

2 BACKGROUND

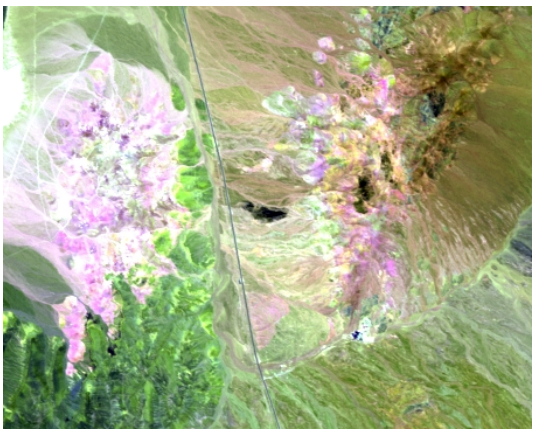


Figure 1: Cuprite, Nevada.

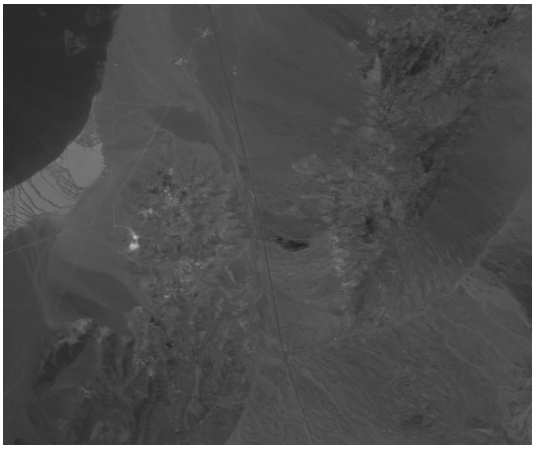


Figure 2: AVIRIS image, 2229nm.

The AVIRIS data used in this study was taken over Cuprite, Nevada on June 12, 1996 (19:31UT). The sensor acquires data in the wavelength region from 0.38 to 2.50 microns, with a ground sampling interval of 16.2m across track (horizontal) and 18.1m along track (vertical). At-sensor radiance data were converted to surface reflectance via an atmospheric correction using the MODTRAN3 radiative transfer (RT) code, as implemented in the imaging spectrometer data analysis system (ISDAS) (Staenz and Williams 1997). This removes spectral artifacts from solar flux and the earth's absorption bands (for example, water). This leaves surface reflectance, which is the data of interest, as it contains the spectral information pertinent to the identification and mapping of specific minerals and vegetation.

Figure 1 shows the Cuprite, Nevada, region studied

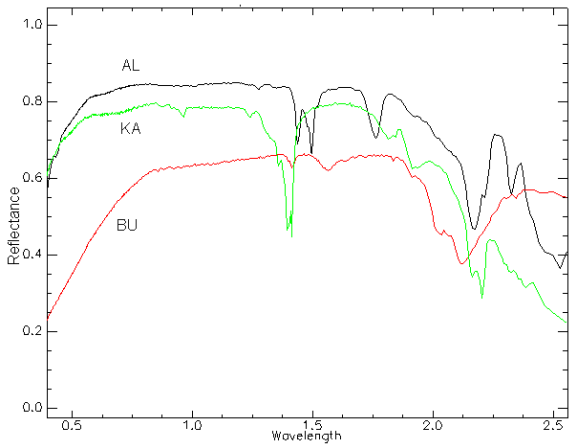


Figure 3: Spectra for alunite (AL), kaolinite (KA), and buddingtonite (BU)

in this paper[2]. Cuprite is a well-studied test area for remote sensing (Resmini *et al.* 1997). Figure 2 shows an AVIRIS hyperspectral reflectance image of the same area at the 2229nm bandwidth.

There is much ongoing research regarding the interpretation of hyperspectral reflectance imagery, and a survey is beyond the scope of this paper. A representative approach to the interpretation of reflectance data is the Tetracorder system(Clark and Swayze 1995). Identification is performed by the application of a least-squares fitting procedure to the total set of spectral data and reference spectra. Features in the absorption patterns of materials are enhanced during this fitting process, in order to promote effective identification. Multiple materials can be fitted simultaneously using this technique. Artificial neural networks are also commonly applied to the automatic analysis and classification of remote sensing data, including AVIRIS data (Ridd *et al.* 1992, Civco 1993, Dreyer 1993, Merenyi *et al.* 1993, Foody and Arora 1997, Yang *et al.* 1999, Aguilar *et al.* 2000).

Evolutionary computation has been applied to multispectral image analysis and remote sensing. (Larch 1994) uses genetic algorithms to evolve categorization production rules for Landsat images. (Daida *et al.* 1996) evolve genetic programs that identify ice-flow ridges from ERS SAR images. Images from aircraft are analyzed using GP in (Howard and Roberts 1999). (Rauss *et al.* 2000) evolve genetic programs for categorizing hyperspectral imagery. The GENIE system is used for hyperspectral image classification, which

[2]North is downwards in this and all the maps in this paper.

uses a hybrid combination of linear genetic programming with conventional classifier algorithms (Harvey *et al.* 2000, Perkins *et al.* 2000, Brumby *et al.* 2001). Among other things, GENIE has been used to classify features such as forest fires and golf courses.

Figure 3 shows the signature spectra for the minerals studied in this paper. Such spectra are measured in the laboratory, and represent reflectance signatures for various organic and inorganic materials. When minerals like these are resident in the environment, hyperspectral imaging will capture similar spectra. Tetracorder uses these lab spectra as guides for identifying minerals in hyperspectral data.

3 EXPERIMENT DESIGN

3.1 Hyperspectral data preparation

The reflectance data from Cuprite derived from the AVIRIS hyperspectral data set was analysed by (Neville *et al.* 1998). The mineral fraction maps which resulted from their work are used as the training solution for this study. From the full AVIRIS bandset available, we started with data at 0.428 microns and eliminated bands near 1.4 and 1.9, where strong absorption in the atmosphere occurs due to water vapour. This left 168 bands of data as input for our GP experiment.

3.2 Training set sampling

The general goal is to evolve a separate identifier for each of the three minerals being studied. The training scheme requires positive examples (pixels where the target mineral is resident) and negative examples (pixels where the mineral is absent). The solution data is given in a mineral distribution map. This is an RGB bitmap of the Cuprite area whose red, green, and blue channels denote the relative reflectance intensity for AL, KA, and BU respectively. Since these minerals are often mixed throughout the Cuprite site, the RGB channels represent mixed intensities of the minerals.

The majority of the Cuprite area is covered by weak mixtures of the minerals. For example, KA and BU exhibit weak distributions over most of the map. These weak areas will negatively influence evolved results, given both the low intensity of resident spectra and existence of spectra from other incident minerals not being studied. Our hypothesis is that better quality results will be obtained for areas with more intense reflectance values for the minerals of interest. Hence mineral identifiers will be evolved for different *thresholds* of reflectance intensity. Thresholds are used to determine the level of reflectance constituting a positive example. A threshold of 25% means that the reflectance value is considered positive if it has an intensity of at least 25% relative to the maximum reflectance observable (100%). Otherwise, it is treated as a negative instance.

Table 1: Training Set Sizes

	Threshold					
	0.0	0.05	0.15	0.25	0.35	0.50
AL pos:	40	40	31	26	20	11
neg:	80	80	89	94	100	109
KA pos:	75	72	57	46	38	23
neg:	80	83	98	109	117	132
BU pos:	77	73	30	19	12	10
neg:	90	94	137	148	155	157

The sizes of the training sets for different thresholds are given in Table 1. Initially, positive and negative example sets were obtained manually, by sampling a diverse selection of pixels throughout the entire map area. The sizes of these sets are listed in the 0% threshold column. The thresholded example sets are refined from these initial sets, by moving positive examples that do not meet the threshold requirements into the negative set. Hence, the positive set sizes decrease as the thresholds are raised.

3.3 GP experiment preparation

Table 2: GP Parameters

Parameter	Value
Population size	1000
Max. generations	100
Max. runs	10
Prob. crossover	0.90
Prob. mutation	0.10
Prob. leaf mutation	0.90
Max. initial depth	2 to 6
Max. depth	17
Tournament size, crossover	4
Tournament size, mutation	7

The GP system used is the typed lilGP 1.1 system (Zongker and Punch 1995). LilGP is a C-based system that implements basic tree-oriented GP. Typing is useful since both integer and floating point values are used in evolved programs (Montana 1995). Some

self-explanatory GP parameters are given in Table 2. Our use of GP is straight-forward: each program in the population is evaluated on the training examples, and its performance in correctly classifying the examples is measured.

Three target languages are used (Table 3). The languages test spectral properties at single pixel locations of the hyperspectral data. Spatial operators are not used. The spectral operators extract hyperspectral data at a pixel coordinate which is globally set for the current program execution. Language L_1 denotes a boolean decision tree, in which a *true* result means that the target mineral resides at the pixel in question. L_1's relational operators test floating point expressions on hyperspectral image parameters $p[I]$, where I is an index (modulo 168) to the hyperspectral image cube. The floating point function set F is self-explanatory. The *inc* operator increments its integer argument. Ephemeral numbers are randomly generated constants.

L_2 and L_3 are floating point languages, in which evaluated values greater than zero are interpreted as positive identification of the mineral. The L_2 language is the subset of L_1 without boolean expressions. The L_3 language is L_2 supplemented with floating point operators that compute over vectors (contiguous ranges of hyperspectral data). These 2-argument functions compute the minimum, maximum, average, and standard deviation over data vectors. The first argument denotes a starting level in the hyperspectral data. The second argument is evaluated modulo 3, and denotes the depth of the vector: 3, 7, or 11 levels. For example, *vavg(35, 2)* computes the average at the current pixel location for layers 35 through 45 inclusive.

The fitness value for a program is computed as:

$$Fitness = 1 - \left(\frac{ce}{te} * \frac{cn}{tn}\right)$$

where *ce* is the number of correctly identified positive examples, *te* is the total number of positive examples, *cn* is the number of correctly identified negative examples, and *tn* is the total number of negative examples. Since the negative training set dwarfs the positive set at higher thresholds, this formula balances positive and negative classification performance with respect to one another.

Table 3: Target Languages

Boolean language L_1:	
B ::=	(if F<F then B else B) | F<F | *true* | *false*
F ::=	p[I] | F+F | F-F | F*F | F/F | min(F,F) | max(F,F) | *ephem flt*
I ::=	inc(I) | *ephem int*
Float language L_2:	F, I from L_1
Float language L_3:	L_2∪ vmin(F,F) | vmax(F,F) | vavg(F,F) | vsdev(F,F)

4 RESULTS

Table 4: Testing and Training Results: % correctly classified pixels

	threshold				
AL	0.05	0.15	0.25	0.35	0.5
testing:					
avg overall	0.825	0.933	0.957	0.966	0.985
best soln	0.875	0.970	0.991	0.995	0.998
TP	0.420	0.587	0.593	0.722	0.644
TN	0.955	0.984	0.997	0.997	0.999
training:					
avg overall	0.87	0.946	0.953	0.961	0.973

	threshold				
KA	0.05	0.15	0.25	0.35	0.5
testing:					
avg overall	0.876	0.952	0.972	0.986	0.987
best soln	0.903	0.964	0.984	0.991	0.994
TP	0.731	0.838	0.869	0.906	0.830
TN	0.963	0.980	0.992	0.997	0.996
training:					
avg overall	0.908	0.963	0.986	0.990	0.966

	threshold				
BU	0.05	0.15	0.25	0.35	0.5
testing:					
avg overall	0.608	0.811	0.972	0.989	0.994
best soln	0.653	0.888	0.993	0.999	0.999
TP	0.797	0.314	0.366	0.592	0.768
TN	0.381	0.945	0.995	1.000	1.000
training:					
avg overall	0.834	0.919	0.986	0.990	0.990

Table 4 shows the training and testing performances for the GP runs. Every mineral and threshold experiment combines the results for 30 runs (3 target languages, 10 runs per language). The testing set is remainder of the input data excluding the training pixels. Testing "avg overall" denotes the percentage of correctly classified pixels averaged for all the solutions from the 30 runs. The performance of the single best solution obtained during the 30 runs is given in the "best soln", TP (true positive), and TN (true nega-

tive) entries. These values respectively report the percentages of correct pixel classifications for the entire testing image, the positive pixels, and negative pixels. The training "avg overall" is the percentage of positive and negative training examples correctly classified, averaged for all 30 solutions.

The training performance is fairly good amongst all the solutions in the runs, and it improves at higher thresholds. The solution ("best soln") programs obtained higher training performances than the averages reported in the table; in higher threshold cases, the best programs often had 100% training scores.

The testing performance of low-threshold results (especially at 0.05) is marginal. This is due to the noisy reflectance values at that low threshold. The overall testing performance improves at higher thresholds. For the best solutions, TN scores tend to be superior to TP scores, which boosts the overall classification score. The relative abundance of negative training examples compared to positive examples at higher thresholds may explain this.

With AL and KA, the best solutions' TP performance usually improves, while the best TN scores always improve. However, the best TP scores decrease when going to the 0.5 threshold with these minerals, and this was seen with other solutions obtained for these runs. Again, the low number of positive training instances of those minerals at this threshold may explain this (see Figure 1).

For BU, raising the threshold from 0.05 to 0.15, the TP fell from 79.7% to 31.4%. This was seen in most other BU runs as well. The distribution of positive examples of BU decreased dramatically from 73 to 30 examples when moving to the 0.15 threshold, and may not adequately characterize the mineral at this threshold.

Best solutions were distributed fairly evenly amongst the three target languages. L_3 solutions were generally the smallest in terms of tree size, followed by L_1 programs and L_2 programs. L_1 programs were the fastest in wall clock time. L_3 and L_2 solutions were respectively an average of 1.6 and 2.3 times slower than L_1 programs. Overall, runs took between 1 to 20 minutes to complete, with a typical run taking about 6 minutes.

Figure 4 shows classification plots for the best solutions listed in Table 4. In images (a) through (i), grey (TN) and white (TP) are correct classifications, while black denotes erroneous classifications. The classifiers clearly have the most difficulty with the lowest threshold value of 5%. For example, the BU example in (g) only classifies 65.3% of the image correctly. Low-threshold classifiers also varied widely in terms of output characteristics. The classifiers do better at the higher thresholds.

Image (j) deconstructs the classification errors in image (g), by rendering false positives with black, false negatives with white. and the remaining correctly classified pixels as grey. This particular classifier was eager to classify mineral instances, hence its relatively high TP score. Clearly, there is a distribution of BU at 5% and higher throughout a large portion of the map area.

The evolved solution program (in l-expression form) for images (g) and (j) is the following:

```
(- (p (inc 29199))
   (p (inc (inc (inc 23424)))))
```

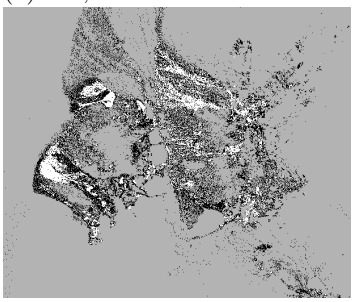
This simplifies to the expression "p[136] - p[75]". This is using the simple classification rule $p_{2129} > p_{1155}$, where p_{2129} is the pixel reflectance at the $2129nm$ bandwidth. Upon first inspection, this rule does not intuitively correspond to the BU spectra graph in Figure 3, where the BU reflectance at $2129nm$ is lower than at $1155nm$. However, the reflectance chart (k) in Figure 4 shows that this simple relation correctly characterizes BU at this low threshold. The chart was created by finding the average intensity of pixels over the range of spectra used in the testing set, for a constrained area that contained a high density of BU at the 5% threshold. From this graph, it is clear that the relation does in fact accurately classify weak densities of BU. It must be realized, however, that the hyperspectral data at low 5% thresholds are likely poor indicators for any of the minerals studied, given the noise resident at that threshold. In addition, the 1155nm position is near a water vapour absorption feature, and selection of this band by the GP solution may be an artifact of the atmospheric correction procedure. This will be investigated further.

Figure 5 shows the classification expression (simplified from the L_1 source program) for the the best solution for BU at the 50% threshold. The expression uses 12 different frequencies over the entire span of hyperspectral data used.

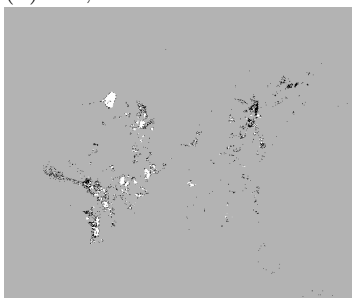
5 CONCLUSION

The hyperspectral mineral identifiers evolved by GP work quite differently from conventional approaches. With least-squares spectra fitting, signature spectra for materials of interest are fitted to the hyperspectral values at each pixel on the map. Identification entails exaggerating the signature differences between

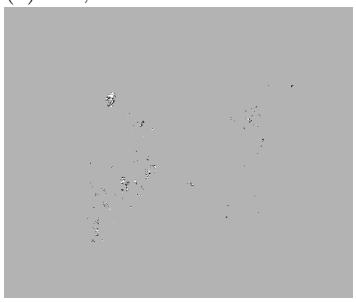
(a) AL, t = 0.05



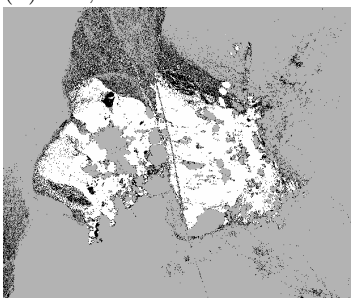
(b) AL, t = 0.25



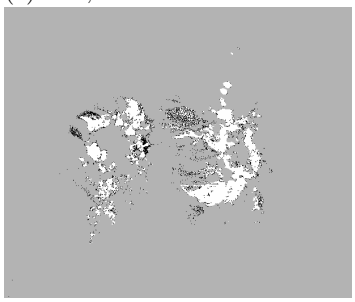
(c) AL, t = 0.50



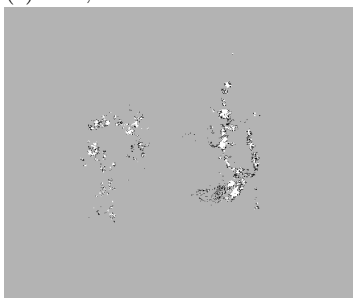
(d) KA, t = 0.05



(e) KA, t = 0.25

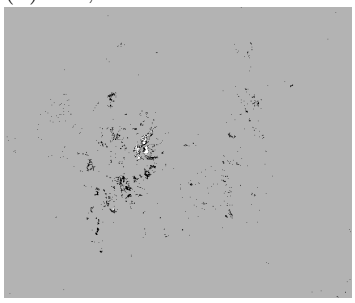


(f) KA, t = 0.50



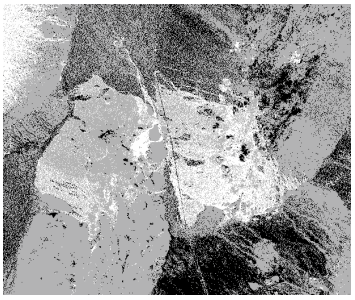
(g) BU, t = 0.05

(h) BU, t = 0.25



(i) BU, t = 0.50

(j) BU, t = 0.05, fp blk, fn wht



(k) BU area reflectance, t = 0.05

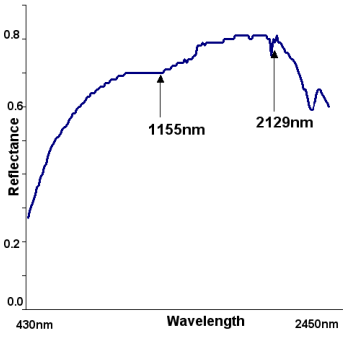


Figure 4: Classification results for map area: alunite (a-c), kaolinite (d-f), and buddingtonite (g-j). In images (a) through (i), grey is true negative, white is true positive, and black is false negative and false positive. In image (j), grey is both true positive and true negative, black is false positive, and white is false negative.

```
if (max(2.1725 + ((min((p[136] - 1.16407), 0.538215) * 0.19977)), p[88])
                / ((p[30] * 0.19731 / p[163]) + 0.88465))
    < (max(0.76650, p[14]) / min((0.18294 / p[11]), (0.00323 + max(-0.10691, p[74]))))
    then ((-1.1482 / min((p[2] - 0.48504), min(0.70156, p[71]))) < p[31])
    else (p[134] < p[151])
```

Figure 5: Evolved L_1 classifier for BU, 50% threshold

materials, and looking for such fluctuations in the hyperspectral data. GP evolves classifiers that find some spectral feature that correctly identifies the existence or absence of a particular mineral. These classifiers do not reference signature spectra, but rather, use the mixed reflectance values as resident in the data set. The success of the classifier depends upon its training performance in differentiating positive and negative examples for the mineral. As a result, material mixtures are automatically accounted for. For example, AL and KA are mixed in a large portion of the Cuprite data set, and the positive training sets for these minerals share many training points.

This implies that the effectiveness of the evolved classifiers implicitly depends upon the context of other materials resident in the geographic area analyzed. The classification logic evolved by GP is best characterized as a function which identifies a mineral in the context of the other minerals resident in the training set. We have not yet tested our mineral identifiers on hyperspectral images from other locations to see how robust these identifiers would be in the presence of materials unseen in the training set. Future work needs to explore the generality of evolved classifiers, in order to see whether a classifier is useful for other geographic locations.

Training set quality is important in our experiments. Our training sets were created by manually selecting positive and negative training points spanning the map area. Although manual sampling is fast and convenient, future work needs to address training sample quality more rigorously. A range of combinations of minerals at various thresholds should be sampled for the positive and negative training sets. This is probably best done via statistical sampling. Such training sets would better represent the varieties of combinations of mineral spectra resident in the images.

Although the fitness formula tries to balance the performance of positive and negative example scoring, many runs produce programs that tend towards being either liberal (eager to identify positive instances) or conservative (eager to report non-instances). Some solutions with very similar fitness scores often have dramatically different classification behaviours, usually falling somewhere on this liberal or conservative dichotomy. These results can mean that the training sets are too small, and evolution is converging prematurely to inadequate solutions.

This work is closest in spirit to that in (Rauss *et al.* 2000). Our L_2 language is similar to theirs, and we also use manually-selected training sets, albeit larger in size than theirs. Their work classified grass from non-grass in hyperspectral images, whereas we classify one of three minerals in each classifier. Our approach can also be compared to the GENIE system (Perkins *et al.* 2000). The GENIE system's application of GP is a bit unusual, as it uses 6 "scratch images", and a fixed-length linear program that may or may not reference these images. Hence the GENIE solution is not as robust a program as a general l-expression program. GENIE also uses conventional classifiers to help analyze and post-process the results from the evolved image analyzer. GENIE uses a large library of spectral and spatial primitive operators, where we use a fairly small set of exclusively spectral operators. When this technology has matured in the future, more careful comparisons between it and other paradigms needs to be undertaken.

Acknowledgement: Thanks to anonymous referees for their constructive comments. This research is supported through NSERC Operating Grants 138467 and 0046210, and an NSERC undergraduate research grant.

References

Aguilar, D.P.K., D.P.M. Cobo, D..R.P. Utrero and D.M.A.H. Nieves (2000). Abundance Extractions from AVIRIS Image Using a Self-Organizing Neural Network. In: *Proceedings of the Ninth Annual JPL Airborne Earth Science Workshop.*

Brumby, S.P., J. Theiler, S. Perkins, N.R. Harvey and J.J. Szymanski (2001). Genetic programming approach to extracting features from remotely sensed imagery. In: *Proceedings FUSION 2001.*

Civco, D.L. (1993). Artificial neural networks for land-cover classification and mapping. *Interna-*

tional Journal of Geographical Information Systems **7**(2), 173–186.

Clark, R.N. and G.A. Swayze (1995). Mapping Minerals, Amorphous Materials, Environmental Materials, Vegetation, Water, Ice and Snow, and Other Materials: The USGS Tricorder Algorithm. In: *Proceedings of the Fifth Annual JPL Airborne Earth Science Workshop* (R.O. Green, Ed.). pp. 39–40. JPL Publication 95-1.

Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross and J.F. Vesecky (1996). Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice. In: *Advances in Genetic Programming II* (P. Angeline and K.E. Kinnear, Eds.). pp. 417–442. MIT Press.

Dreyer, P. (1993). Classification of Land Cover Using Optimized Neural Nets on SPOT Data. *Photogrammetric Engineering and Remote Sensing* **59**(5), 617–621.

Foody, G.M. and M.K. Arora (1997). An evaluation of some factors affecting the accuracy of classification by an artificial neural network. *International Journal of Remote Sensing* **18**(4), 799–810.

Green, R.O., M.L. Eastwood, C.M. Sarture, T.G. Chrien, M. Aronsson, B.J. Chippendale, J.A. Faust, B.E. Pavri, C.J. Chovit, M. Solis, M.R. Olah and O. Williams (1998). Imaging Spectrometry and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). *Remote Sensing of Environment* **65**, 227–248.

Harvey, N.R., S.P. Brumby, S.J. Perkins, R.B. Porter, J. Theiler, A.C. Young, J.J. Szymanski and J.J. Bloch (2000). Parallel evolution of image processing tools for multispectral imagery. In: *Proceedings Imaging Spectrometry IV: SPIE-4132*. Intl. Soc. for Opt. Eng.. pp. 72–80.

Howard, D. and S.C. Roberts (1999). A Staged Genetic Programming Strategy for Image Analysis. In: *Proc. GECCO-99* (W. Banzhaf *et al*, Ed.). pp. 1047–1052.

Larch, D. (1994). Genetic Algorithms for Terrain Categorization of Landsat Images. In: *Proceedings SPIE-2231: Algorithms for Multispectral and Hyperspectral Imagery*. Intl. Soc. for Opt. Eng.. pp. 2–6.

Merenyi, E., R.B. Singer and W.H. Farrand (1993). Classification of the LCVF AVIRIS Test Site with a Kohonen Artificial Neural Network. In: *Proceedings of the Fourth Annual JPL Airborne Earth Science Workshop*. pp. 117–120.

Montana, D.J. (1995). Strongly Typed Genetic Programming. *Evolutionary Computation* **3**(2), 199–230.

Neville, R.A., C. Nadeau, J. Levesque, T. Szeredi, K. Staenz, P. Hauff and G.A. Borstad (1998). Hyperspectral Imagery for Mineral Exploration: Comparison of Data from Two Airborne Sensors. In: *Proceedings Imaging Spectrometry VI: SPIE-3438*. Intl. Soc. for Opt. Eng.. pp. 74–83.

Perkins, S.J., J. Theiler, S.P. Brumby, N.R. Harvey R.B. Porter, J.J. Szymanski and J.J. Bloch (2000). GENIE: A Hybrid Genetic Algorithm for Feature Classification in Multi-Spectral Images. In: *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation III* (B. Bosacchi, D.B. Fogel and J.C. Bezdel, Eds.). pp. 52–62.

Rauss, P.J., J.M. Daida and S. Chaudhary (2000). Classification of Spectral Imagery Using Genetic Programming. In: *GECCO-2000* (D. Whitley *et al.*, Ed.). Morgan Kaufmann. pp. 726–733.

Resmini, R.G., M.E. Jappus, W.S Aldrich, J.C. Harsanyi and M. Anderson (1997). Mineral mapping with Hyperspectral Digital Imagery Collection Experiment (HYDICE) sensor data at Cuprite, Nevada, U.S.A.. *International Journal of Remote Sensing* **18**(7), 1553–1570.

Ridd, M.K., N.D. Ritter, N.A. Bryant and R.O Green (1992). AVIRIS Data and Neural Networks Applied to an Urban Ecosystem. In: *Proceedings of the Second Annual JPL Airborne Earth Science Workshop*. pp. 129–131.

Staenz, K. and D.J. Williams (1997). Retrieval of surface reflectance from hyperspectral data using a look-up table approach. *Canadian Journal Remote Sensing* **23**, 345–368.

Yang, H., F. van der Meer, W. Bakker and Z.J. Tan (1999). A back-propagation neural network for mineralogical mapping from AVIRIS data. *International Journal of Remote Sensing* **20**(1), 97–110.

Zongker, D. and B. Punch (1995). *lil-gp 1.0 User's Manual*. Dept. of Computer Science, Michigan State University.

Voice Conversion Using Interactive Evolution of Prosodic Control

Yuji Sato

Faculty of Computer and Information Sciences
Hosei University
3-7-2, Kajino-cho, Koganei-shi, Tokyo 184-8584, Japan
E-mail: yuji@k.hosei.ac.jp

Abstract

This paper proposes the application of evolutionary computation, a stochastic search technique that parallels the evolution of living organisms, to parameter adjustment for voice conversion, and reports on several experimental results applicable to the fitting of prosodic coefficients. Here, because of the difficulty involved in providing a clear fitness function for evaluating evolutionary computation, we adopt a system of interactive evolution in which genetic manipulation is repeated while evaluation is performed subjectively based on human feelings. It was found that the use of evolutionary computation achieves voice conversion closer to the target in question than parameter adjustment based on designer experience or trial and error, and that degradation in sound quality is relatively small giving no impression of a processed voice.

1 INTRODUCTION

With the coming of the multimedia era, the market for multimedia information devices centered about personal computers is experiencing rapid growth. Likewise, the market for multimedia application software is taking off giving rise to an environment in which users can manipulate images and sound with ease. In particular, speech synthesis technology is expected to generate a large market for a wide rage of applications from the reading of E-mail and text data on the World Wide Web to the speaking of road traffic reports provided by navigation devices. Nevertheless, mechanically synthesized speech by a rule-based speech synthesis system or similar suffers from a variety of problems. These include an impression of discontinuity between phoneme fragments, degraded sound quality due to repeated signal processing, and limitations in sound-source/articulation segregation models. In other words, the synthesis of natural speech is extremely difficult. Current technology tends to produce mechanical or unintelligible speech, and problems such as these are simply delaying the spread of speech synthesis products.

Research has also begun on the application of voice processing to narration when editing multimedia content as in a spoken presentation. The need for voice conversion (processing) arises from the fact that most people have difficulty speaking with an expressive and clear voice. However, only qualitative know-how has so far been obtained in the development of voice-processing technology for converting original speech to clear narration. Parameter setting is currently performed on a trial and error basis making adjustments difficult.

Against the above background, this research aims to establish technology for converting original human speech or speech mechanically synthesized from text to clear speech rich in prosodic stress. As the first step to this end, we have proposed the application of evolutionary computation to parameter adjustment for the sake of voice conversion using original speech recorded by a microphone as input data, and have reported on several experimental results applicable to the fitting of prosodic coefficients [Sato 1997]. In this paper, we show that parameter adjustment using evolutionary computation can be effective not only for voice conversion using original speech as input but also for improving the clarity of speech mechanically synthesized from text. We also investigate why parameter adjustment using evolutionary computation is more effective than that based on trial and error by an experienced designer.

2 VOICE ELEMENTS AND VOICE CONVERSION

This section summarizes the feature quantities needed for voice conversion and describes voice conversion by prosodic control.

2.1 VOICE ELEMENTS

In human speech production, the vocal cords serve as the sound generator. The vocal cords, which are a highly

flexible type of muscle located deep in the throat, are made to vibrate by breath expelled from the lungs, thereby causing acoustic vibrations in the air (sound waves). The waveform of this acoustic signal is approximately triangular or saw-tooth in form and consists of harmonic components that are integer multiples of the fundamental frequency of the sound wave. This acoustic signal that has a broad range of harmonic components of a constant interval propagates through the vocal tract from the vocal cords to the lips and acquires resonances that depend on the shape of the vocal tract. This transformations results in the production of phonemes such as /a/ or /i/, which are finally emitted from the lips as speech. That is to say, the human voice characteristics are determined by three factors: sound generation, propagation in the vocal tract, and emission. The vocal cords control the pitch of the voice and the shape of the vocal tract controls prosody. If we define voice quality in terms of properties such as timbre, we can consider voice quality to be determined by both the state of the vocal cords and the state of the vocal tract [Klatt 1990]. That is to say, we can consider pitch structure, amplitude structure, temporal structure and spectral structure as the feature quantities for the control of voice quality.

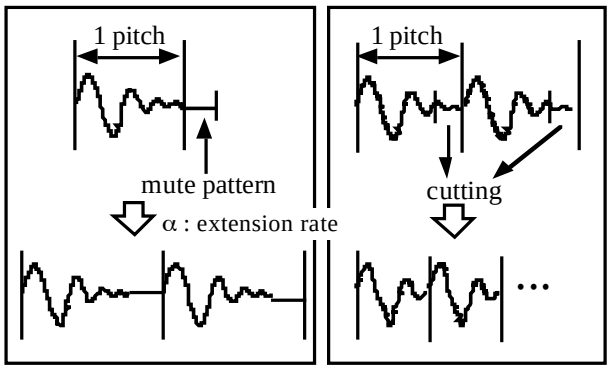
2.2 MODIFICATION OF VOICE QUALITY THROUGH PROSODIC ADJUSTMENT

Research on the features of the voices of professional announcers has clarified to some extent the qualitative tendencies that are related to highly-intelligible speech. It is known, for example, that raising the overall pitch slightly and increasing the acoustic power of consonants slightly increases intelligibility [Kitahara 1992]. It remains unclear, however, to what specific values those parameters should be set. Moreover, it is generally difficult to control dynamic spectral characteristics in real time. In other words, it is difficult to even consider adjusting all of the control parameters to begin with. Therefore, sought to achieve voice conversion by limiting the data to be controlled to pitch data, amplitude data, and temporal structure prosodic data**.**

The pitch conversion method is shown in Fig. 1. Pitch is raised by cutting out a part of the waveform within one pitch unit. Pitch is lowered by inserting silence into a pitch unit. Modification of the temporal structure is accomplished as illustrated in Fig. 2. The continuation length is accomplished by using the TDHS [Malah 1979] enhancement method to extend or contract the sound length without changing the pitch. Amplitude is modified on a logarithmic power scale according to the formula

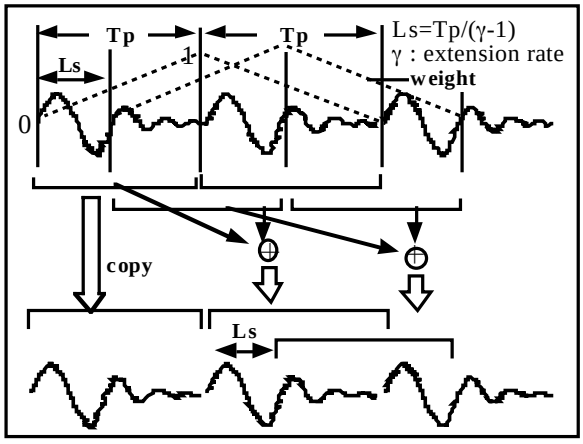
$$\log_{10} W_{i+1}{}^2 = \log_{10} W_i^2 + \beta \qquad (1)$$

Where W_i is the current value and β is the modification coefficient.

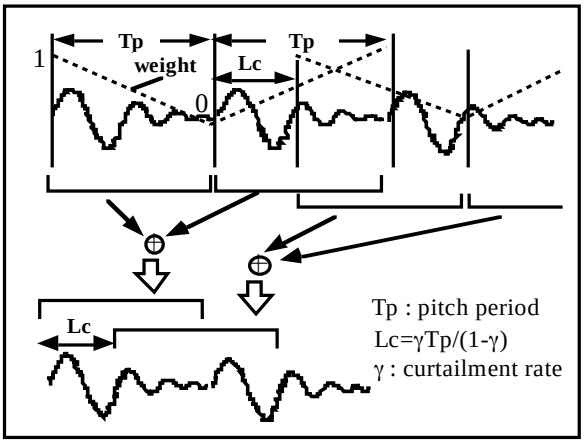


(a) Extension (b) Curtailment

Figure 1: Extension and curtailment of pitch period. Pitch is raised by cutting out a part of the waveform within one pitch unit. Pitch is lowered by inserting silence into a pitch unit..



(a) Extension of temporal structure



(b) Curtailment of temporal structure

Figure 2: Extension and curtailment of temporal structure. The continuation length is accomplished by using the TDHS enhancement method to extend or contract the sound length without changing the pitch.

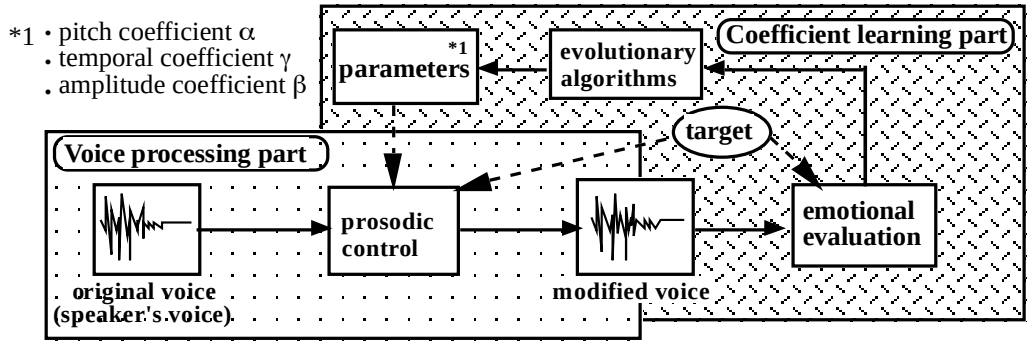


Figure 3: Block diagram of proposed voice quality conversion system. The system comprises a voice processing part and prosody control coefficient learning part.

3 PROSODIC COEFFICIENT FITTING BY EVOLUTIONARY COMPUTATION

3.1 CONFIGURATION OF THE VOICE MODIFICATION SYSTEM

The configuration of the voice modification system is illustrated in Fig. 3. The system comprises a voice processing part and prosody control coefficient learning part. The voice modification unit changes voice quality, targeting terms that express emotional feelings, such as "clear," and "cute." The modification of prosodic information is done by the prosodic control unit. To prevent degradation of voice quality, the processing is done at the waveform level as described above rather than at the parameter level, as is done in the usual analysis-synthesis systems. The modification coefficient learning unit is provided with qualitative objectives, such as terms of emotion, and the modification coefficients used for prosodic modification targeting those objectives are acquired automatically by learning. As the learning algorithm, this unit employs evolutionary computation, which is generally known as an effective method for solving problems that involve optimization of a large number of combinations.

3.2 OVERVIEW OF INTERACTIVE EVOLUTION OF PROSODIC CONTROL

The first step in this procedure is to define chromosomes, i.e., to substitute the search problem for one of determining an optimum chromosome. As shown in Fig. 4, we define a chromosome as a one-dimensional real-number array corresponding to a voice-conversion target (an emotive term) and consisting of three prosody modification coefficients. Specifically, denoting the pitch modification factor as α, the amplitude modification factor as β, and the continuation time factor as γ, we define a chromosome as the array [α, β, γ]. The next step is to generate individuals.

target	prosodic components transform parameters		
	pitch structure	amplitude structure	temporal structure
intelligible	**1.172**	**1.365**	**0.918**
childlike	**1.383**	**-1.366**	**0.907**
calm	**0.992**	**1.074**	**1.015**

chromosomes

Figure 4: Example of the chromosomes. It is defined by an array, [pitch modification factor α, amplitude modification factor β, continuation time factor γ].

Here, we generate 20, and for half of these, that is, 10 individuals, chromosomes are defined so that their prosody modification coefficients change randomly for each voice-conversion target. For the remaining 10, chromosomes are defined so that their coefficients change randomly only within the vicinity of prosody-modification-coefficient values determined from experience on a trial and error basis. In the following step, evaluation, selection, and genetic manipulation are repeated until satisfactory voice quality for conversion is attained. Several methods of evaluation can be considered here, such as granting points based on human subjectivity or preparing a target speech waveform beforehand and evaluating the mean square difference between this target waveform and the output speech waveform from voice-conversion equipment. In the case of evolutionary computation, a designer will generally define a clear evaluation function beforehand for use in automatic recursion of change from one generation to another. It is difficult to imagine, however, a working format in which an end user himself sets up a clear evaluation function, and in recognition of this difficulty, we adopt a system of interactive evolution [Sims 1991, Takagi 2001] in which people evaluate results subjectively (based on feelings) for each generation.

3.3 GENETIC MANIPULATION

3.3.1 Selection Rule

Culling and selection are based on a fitness value, as shown in Fig. 5. First, the individuals are sorted by their fitness values. In the example shown in Fig. 5, 20 individuals are sorted in order of high fitness value with respect to the objective of high intelligibility. The population is then culled. Here, The half of the individuals with the lowest fitness values is culled. The proportion of the population culled does not have to be 50%; another approach is to cull all individuals whose fitness values are below a certain standard value. Next, the population is replenished by replacing the culled individuals with a new generation of individuals picked by roulette selection [Goldberg 1989] in this example. To produce the new generation, first two chromosomes are selected as the parents. Offspring are generated from the parents by the crossover and mutation process described below. Here, the probability of selecting the two parent chromosomes is proportional to the fitness values. Furthermore, duplication in the selection is permitted. All individuals are parent candidates, including the culled individuals. In other words, taking M as the number of individuals to be culled, we randomly select only M pairs of individuals from the current generation of N individuals (I_1 to I_N), permitting duplication in the selection. The crossover and mutation genetic manipulation operations are performed on those pairs to provide M pairs of individuals for replenishing the population. Here, the probability $P(I_i)$ of an individual I_i being selected as a parent for creating the next generation of individuals is determined by the following equation. The term $f(I_i)$ in this equation expresses the degree of adaptability of I_i.

$$P(I_i) = f(I_i) / \left\{ \sum_{j=1}^{N} f(I_j) / N \right\} \qquad (2)$$

Although the method used here is to assign a fitness value to each individual and cull the individuals that have low values, it is also possible to select the individuals to be culled by a tournament system. In that case, we do not have access to the fitness values, so we considered random selection of the parent individuals.

3.3.2 Crossover and Mutation

Figure 6 presents an example of crossover. In the crossover operation, any one column is chosen and the values in that column are swapped in the two parent individuals. In Fig. 6, the modification coefficients for continuation length are exchanged between the two parents. The crossover genetic manipulation has the effect of propagating bit strings (chromosome structural components) that are linked to high fitness values to another individual. If these structural components, which are referred to as building blocks [Goldberg 1989], are successfully assembled in an accurate manner, then an effective search is accomplished.

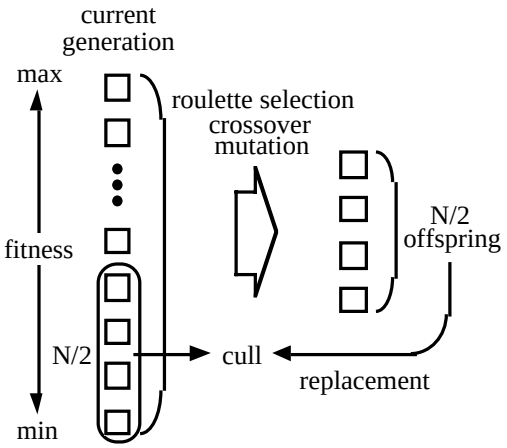


Figure 5: Selection rule.

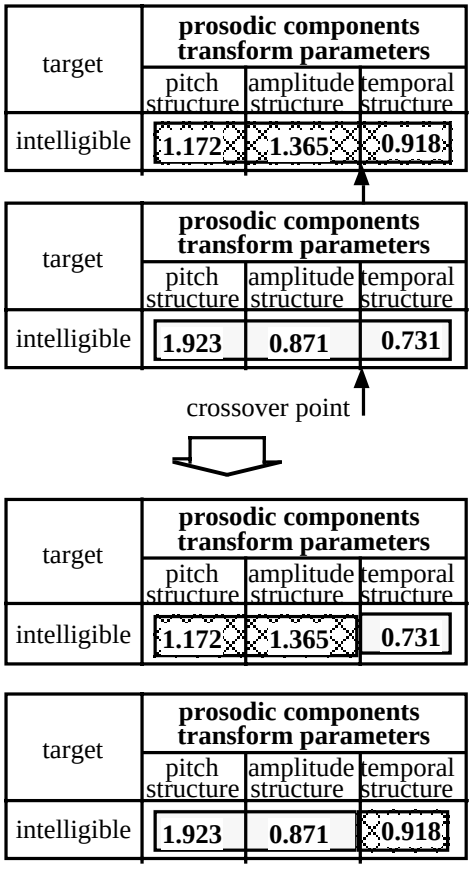


Figure 6: Example of crossover. In the crossover operation, any one column is chosen and the values in that column are swapped in the two parent individuals.

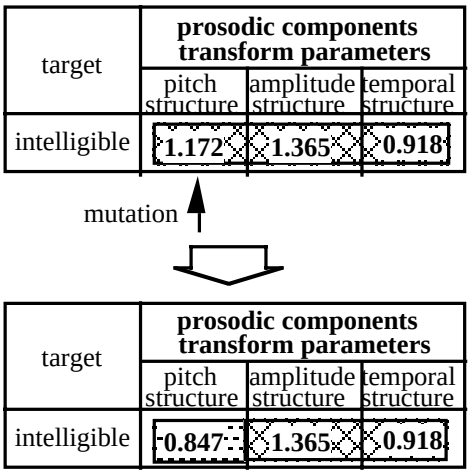


Figure 7: Example of mutation. In this example, the modification parameter for pitch is chosen and the value is varied in the range from 1.172 to 0.847.

Figure 7 shows an example of mutation whereby a prosody modification coefficient is arbitrarily selected and randomly changed. In this example, the operation selects the modification coefficient related to pitch and its value mutates from 1.172 to 0.847. Here, we use mutation as represented by Eq. (3) to raise the probability that target mutants are in the vicinity of parents and to improve local searching. In the equation, C_i represents a modification coefficient for generation i, I is a unit matrix, k is a constant, and N is a normal distribution function with a mean vector of 0 and a covariance of kI and is common to all elements.

$$C_{i+1} = C_i + \delta_i \quad (i = 1 \textit{ to } 20) \tag{3}$$

$$\delta_i = N(0, kI) \tag{4}$$

This mutation operation has the effects of escaping from local solutions and creating diversity. In addition, crossover and mutation combined raise the fitness value, that is, the vicinity of the modification coefficient can be efficiently searched near the voice-conversion target. Moreover, as multiple individuals are performing parallel searches from different initial values, initial-value dependency is low and positive effects from parallel processing can be expected.

In the experiments described below, we used a crossover rate of 0.5 and a mutation rate of 0.3.

4 EVALUATION EXPERIMENTS

4.1 EXPERIMENT WITH ORIGINAL SPEECH AS INPUT DATA

4.1.1 Voice Stimuli

The original voice sample, *S*0, was the sentence, “Let me tell you about this company.” spoken by a female in Japanese. Five modified samples, *SA*1 through *SA*5, that correspond to the five emotive terms, “intelligible,” “childish,” “joyful,” “calm,” and “angry,” were produced by applying prosody modification coefficients obtained by the evolutionary computation learning scheme described above. In addition, five modified samples, *SB*1 through *SB*5, that correspond to the same five emotive terms, “intelligible,” “childish,” “joyful,” “calm,” and “angry,” were produced by applying prosody modification coefficients obtained by trial and error based on the experience of a designer.

4.1.2 Experimental Method

The subjects of the experiments were 10 randomly selected males and females between the ages of 20 and 30 who were unaware of the purpose of the experiment. Voice sample pairs *S*0 together with *SAi* (*i* = 1 to 5) and *S*0 together with *SBi* (*i* = 1 to 5) were presented to the test subjects through speakers. The subjects were instructed to judge for each sample pair whether voice modification corresponding to the five emotive terms specified above had been done by selecting one of three responses: “Close to the target expressed by the emotive term,” “Can't say,” and “Very unlike the target.” To allow quantitative comparison, we evaluated the degree of attainment (how close the modification came to the target) and the degree of good or bad impression of the sample pairs on a nine-point scale for the childish emotive classification. Subjects were allowed to hear each sample pair multiple times.

4.1.3 Experimental Results

The results of the judgments of all subjects for voice sample pairs *S*0 - *SAi* (*i* = 1 to 5) and *S*0 - *SBi* (*i* = 1 to 5) are presented in Fig. 8 as a histogram for the responses “Close to the target” and “Very unlike the target”. From those results, we can see that although the trial and error approach to obtaining the modification coefficients was successful for the “childish”, “intelligible”, and “joyful” classifications, the modification results were judged to be rather unlike the target for the “calm” and “angry” classifications. In contrast to those results, the samples produced using the modification coefficients obtained by the evolutionary computation approach were all judged to be close to the target on the average.

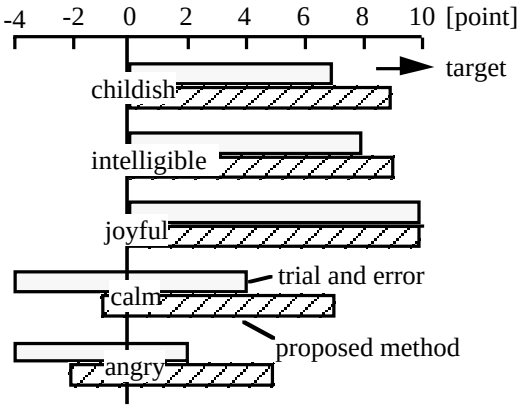


Figure 8: The results of the judgments of all subjects for voice sample pairs. The results are presented as a histogram for the responses "Close to the target" and "Very unlike the target".

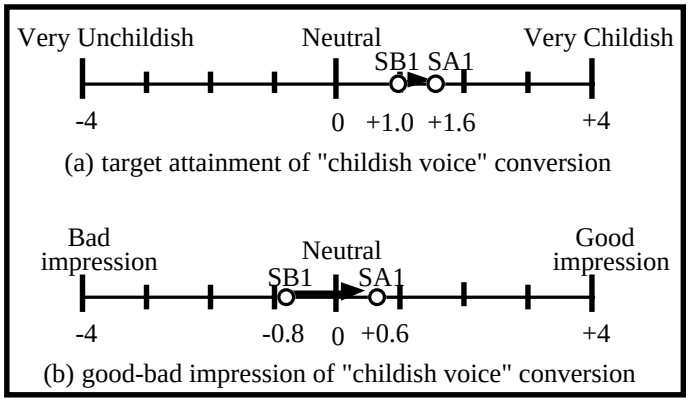


Figure 9: The results of the evaluation of target attainment and good-bad impression. The values averaged for all subjects are presented.

Next, we consider the results of the evaluation of target attainment and good/bad impression. The values averaged for all subjects are presented in Fig. 9. Relative to an attainment rate of +1.0 for the prosody modification coefficient combination obtained by a designer according to experience, the attainment rate for the evolutionary approach was 1.6, or an improvement of 0.6. For the impression evaluation, the scores were -0.8 for the human design approach and +0.6 for the evolutionary computation approach, or an improvement of 1.6. We believe that the reason for these results is that there was a strong tendency to raise the pitch in the adjustment by the designer to achieve the "childish voice" modification, resulting in a mechanical quality that produced an unnatural impression. The evolutionary computation approach, on the other hand, resulted in a modification that matched the objective without noticeable degradation in sound quality, and thus did not give the impression of processed voice.

4.2 EXPERIMENT WITH SYNTHESIZED SPEECH AS INPUT DATA

4.2.1 Voice Stimuli

The voice stimuli used in this experiment were as follows. Voice sample *S*1 consisted of the words "voice conversion using evolutionary computation of prosodic control" mechanically synthesized from text using Macintosh provided software (Macin Talk3). Voice samples *SC*1 to *SC*3 were obtained by performing voice conversion on the above sample for the three emotive terms of "childish," "intelligible," and "masculine" applying prosody modification coefficients obtained by the learning system using evolutionary computation as described above.

4.2.2 Experimental Method

As in the experiment using original speech, the subjects were 10 randomly selected males and females between the ages of 20 and 30 knowing nothing about the purpose of the experiment. Voice sample pairs *S*1 and *SCi* (*I*= 1-3) were presented through a speaker to these 10 subjects who were asked to judge whether voice conversion had succeeded in representing the above three emotive terms. This judgement was made in a three-level manner by selecting one of the following three responses: "close to the target expressed by the emotive term," "can't say," and "very unlike the target." Furthermore, for the sake of obtaining a quantitative comparison with respect to the emotive term "intelligible," we also had the subjects perform a nine-level evaluation for both degree of attainment in voice conversion and good/bad impression for this voice sample pair. Subjects were allowed to hear each sample pair several times.

4.2.3 Experimental Results

The judgments of all subjects for voice sample pairs S1 and *SCi* (*i* = 1-3) are summarized in Fig. 10 in the form of a histogram for the responses "close to the target" and "very unlike the target." These results demonstrate that voice conversion is effective for all emotive terms on average.

Figure 11 shows the results of judging degree of attainment and reporting good/bad impression averaged for all subjects. We see that degree of attainment improved by +1.2 from a value of +0.0 before conversion by determining an optimum combination of prosody modification coefficients using evolutionary computation. We also see that good/bad impression improved by +0.8 changing from +0.6 to +1.4.

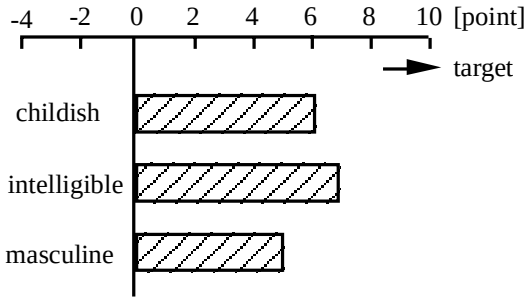


Figure 10: The results of the judgments of all subjects for voice sample pairs. The results are presented as a histogram for the responses "Close to the target" and "Very unlike the target".

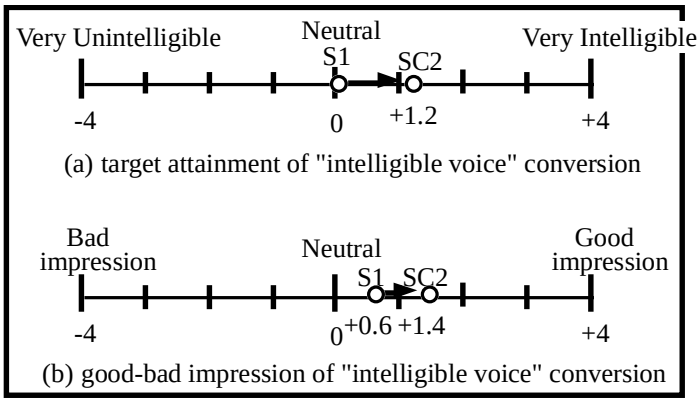


Figure 11: The results of the evaluation of target attainment and good-bad impression. The values averaged for all subjects are presented.

5 DISCUSSION

The above experiments have shown that voice conversion using evolutionary computation can get closer to a target than parameter adjustment based on a designer's experience or trail and error. They have also shown that degradation in sound quality is relatively small and that listeners are not given a strong impression of a processed voice in the case of evolutionary computation. We here examine the question as to why evolutionary computation is superior. First, we consider the problem of accuracy in prosody modification coefficients. In the past, coefficients have been adjusted manually using real numbers of two or three significant digits such as 1.5 and 2.14. Such manual adjustment, however, becomes difficult if the search space becomes exceedingly large. On the other hand, it has been observed that a slight modification to a prosody modification coefficient can have a significant effect on voice conversion. For example, while raising pitch is an effective way of making a voice "childish," increasing the pitch modification factor gradually while keeping the amplitude modification factor and continuation time factor constant can suddenly produce an unnatural voice like that of a "spaceman." This can occur even by making a slight modification to the fourth or fifth decimal place. In other words, there are times when the accuracy demanded of prosody modification coefficients will exceed the range of manual adjustment.

Second, we consider the fact that each type of prosody information, that is, pitch, amplitude, and time continuation, is not independent but related to the other types. When manually adjusting coefficients, it is common to determine optimum coefficients one at a time, such as by first adjusting the pitch modification factor while keeping the amplitude modification factor and continuation time factor constant, and then adjusting the amplitude modification factor.

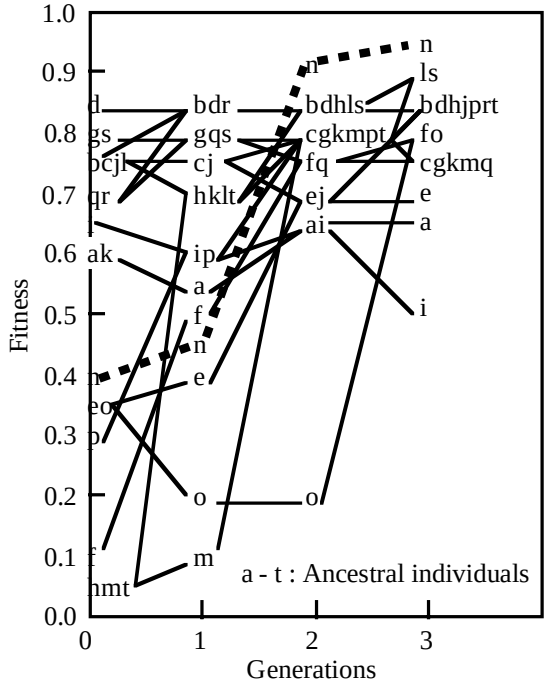


Figure 12: Fitness landscapes for "childish voice" conversion.

However, as pitch, amplitude, and time continuation are not independent of each other but exhibit correlation, it has been observed that changing the amplitude modification factor after setting an optimum value for the pitch modification factor will consequently change the optimum solution for pitch. This suggests that the modification coefficients for pitch, amplitude, and continuation time must be searched for in parallel.

Third, we consider the problem of multimodality accompanied by time fluctuation. For example, it often happens that a subject may not necessarily find an optimum solution from a voice that has already been subjected to several types of conversion. It has also been observed that optimum solutions may vary slightly according to the time that experiments are held and the physical condition of subjects at that time. In other words, we can view the problem as being one of determining a practical semi-optimum solution in as short a time as possible from a search space having multimodality and temporal fluctuation in the difficulty of prediction.

On the basis of the above discussion, we can see that the problems of voice conversion are indeed complex. For one, a practical semi-optimum solution must be determined in as short a time as possible from a search space having multimodality and temporal fluctuation in the difficulty of prediction. For another, high accuracy is demanded of modification coefficients and several types of modification coefficients must be searched for in parallel. In these experiments, we have shown that evolutionary computation is promising as an effective means of voice conversion compared to the complex real-world problems associated with finding an explicit algorithm and a solution based on trail and error by a designer. As a specific example, Fig. 12 shows the relationship between number of generations and fitness with respect to a "childish voice." Ancestral individual information is shown from "a" to "t". Here, individuals having prosody modification coefficients determined by experience are placed in the vicinity of a local optimum solution, and it takes only three generations to converge to a practical solution by performing genetic manipulation between these individuals and other individuals whose prosody modification coefficients are randomly set. Please see the example of voice conversion provided at http://webclub.kcom.ne.jp/ma/y-sato/demo/demo1.html for reference.

In future work, we will attempt to improve the accuracy of voice conversion by modifying spectral data as well, and must examine the application of evolutionary computation to parameter adjustment with the aim of synthesizing truly natural voices from arbitrary text. In this experiment, people evaluate results subjectively (based on feelings) and assign a fitness value to each individuals, it is also possible to select the individuals to be culled by a tournament system. It is also important to compare with other Evolutionary Computation method [Bäck 1997].

6 CONCLUSIONS

We have proposed the application of evolutionary computation to the adjustment of prosody modification coefficients for voice conversion, and have conducted voice-conversion experiments on both original speech recorded by a microphone and speech mechanically synthesized from text to evaluate the effectiveness of the proposed method. The results of these experiments revealed that adjustment of prosody modification coefficients by evolutionary computation performs voice conversion more efficiently than manual adjustment, and that degradation in sound quality is relatively small with no impression of a processed voice in the case of evolutionary computation. Future research must work on improving the accuracy of voice conversion by modifying spectral data as well, and must examine the application of evolutionary computation to parameter adjustment with the aim of synthesizing truly natural voices from arbitrary text.

Acknowledgments

The author would like to express their gratitude to Dr. Y. Kitahara, Mrs. H. Ando, and Mrs. M. Sato of Hitachi, Ltd., for their valuable discussions on experimental results.

References

T. Bäck, U. Hammel and H.-P. Schwefel (1997). Evolutionary Computation: Comments on the History and Current State, *IEEE Trans. on Evolutionary Computation*, Vol.1, No.1, pp.3-17.

D.E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Y. Kitahara and Y. Tohkura (1992). Prosodic Control to Express Emotions for Man-Machine Speech Interaction", *IEICE Trans. Fundamentals.*, Vol. E75, No. 2, pp. 155-163.

D.H. Klatt and L.C. Klatt (1990). Analysis, Synthesis, and Perception of Voice Quality Variations among Female and Male Talkers, *Jounal of Acoustic Society America*, 8

J D. Malah (1979). Time-domain algorithms for harmonic bandwidth reduction and time scaling of speech signals", *IEEE Trans. Acoust. Speech, Signal Processing*, Vol. ASSP-27, pp. 121-133.

Y. Sato (1997). Voice Conversion Using Evolutionary Computation of Prosodic Control, in *Proc. of the Australasia-Pacific Forum on Intelligent Processing and Manufacturing of Materials*, pp. 342-348.

K. Sims (1991). Interactive Evolution of Dynamical Systems, in F.J. Varela and P. Bourgine (eds.), Toward a Practice of Autonomous Systems, in *Proc. of the First European Conf. on Artificial Life*, MIT Press, pp.171-178.

H. Takagi (2001): Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation, *Tutorial Book of the 2001 Congress on Evolutionary Computation.*

Improving Digital Video Commercial Detectors with Genetic Algorithms

J. David Schaffer **Lalitha Agnihotri** **Nevanka Dimitrova** **Thomas McGee** **Sylvie Jeannin**

Philips Research
345 Scarborough Road
Briarcliff Manor, NY 10510, USA
{dave.schaffer, lalitha.agnihotri, nevenka.dimitrova, thomas.mcgee}@philips.com

Abstract

The advent of digital video offers many opportunities to add features that enhance the viewing experience. One much-discussed feature is the possibility that commercials might be automatically detected in the video stream. We report on initial experiments with a class of commercial detection algorithms and show how their performance can be enhanced by applying genetic search to the optimization of some of their internal parameters. We show how a scalar genetic algorithm can locate sets of parameters in a multi-objective space (precision and recall) that outperform the values selected by an expert engineer. While a useful observation in itself, we also argue that this approach may be a necessity as the features that distinguish commercials from other video content will certainly vary with video format, the country of broadcast and possibly over time. We present the results of optimizing a commercial detection algorithm for different data sets and parameter sets. We are convinced that GAs drastically improved our approach and enabled fast prototyping and performance tuning of commercial detection algorithms.

1 INTRODUCTION

Digital consumer storage functionality will appear on many consumer devices such as video recorders advanced set-top boxes and personal mobile storage servers. Content-based video analysis can be applied to introduce more advanced retrieval, scanning and playback features. One of the most important features consumers want is commercial detection, indication, and skipping. In addition, commercial detection plays a major role in automatic analysis and structure detection from multimedia signals for generating summaries and table of contents for a program. A major challenge in bringing these features into consumer devices is to overcome the low processing power inherent in these low-cost consumer devices. It is therefore, important to take full advantage of the MPEG[1] hardware compression and perform the analysis on the features already available during the encoding of the input video stream thereby saving precious computational cycles.

The implementation of a commercial detection algorithm using features derived from MPEG parameters has been described by Dimitrova et al. [3]. This implementation assumes that the target platform includes an encoder and the features extracted from the encoder are processed on a low-end host processor. Consequently, the chosen algorithms are based on simple voting and thresholding techniques. An important challenge is to provide high accuracy commercial detectors for given test material. The process of benchmarking and fine tuning is tedious and requires many experiments with various thresholds. It normally takes weeks to fine tune an algorithm. Also, experiments are needed to see the impact of threshold ranges on the algorithms for different types of TV programs. It is extremely important to provide methodology for fast tuning of the algorithm parameters and providing tools for analysis of the algorithms for given test genres. Here, we report on experiments that used a genetic algorithm (GA) to locate improved sets of thresholds on a chosen commercial detection algorithm [3]. In addition, GAs provide a framework for fast tuning and analysis of parameters for commercial algorithms.

2 THE CHALLENGE OF COMMERCIAL DETECTION

What we encounter is a fairly traditional pattern discrimination problem, yet there are reasons to believe that the achievement of successful performance will require the use of powerful optimization methods, and not just once. The patterns of features and their combinations

[1] MPEG stands for Moving Picture Expert Group. This body establishes standards that are used in the compression, transmission, and decompression of digital video.

that distinguish commercials from other video content are known to change with video format (e.g. NTSC vs PAL[2]), and with culture (cinematic styles differ from culture to culture). Furthermore, these patterns are not well understood, but our initial investigations suggest that they are highly non-linear. In addition, they can be expected to change over time as styles of programming as well as styles of advertising change. The broadcaster and/or advertiser may also change the advertisement characteristics to avoid detection. Therefore, we envision a need to more or less continuously resample video content and re-adjust the detector's parameters. This calls for a robust automatic optimization method such as a GA.

3 RELATED WORK

In the literature there are many methods that have been proposed for detecting commercials extending back more than 20 years [1, 2, 5, 6, 7, 8, 9, 10, 11]. One common method is detection of high activity rate and black frame detection coupled with silence detection before a commercial break. These methods show partially promising results [8]. The use of monochrome images, scene breaks, and action (the number of edge pixels changing between consecutive frames and motion vector length) as indicative features have also been reported [8]. Blum et al. used black frame and "activity" detectors [1]. Activity is the rate of change in luminance level between two different sets of frames. Commercials are generally rich in activity. When a low activity is detected, the commercial is deemed to have ended. Unfortunately, it is difficult to determine what is "activity' and what is the duration of the activity. In addition, black frames are also found in dissolves. Any sequence of black frames followed by a high action sequence can be misjudged and skipped as a commercial. Another technique by Iggulden is using the distance between black frame sequences to determine the presence of a commercial [6]. Lewine et al. determined commercials based on matching images. Similarly, Forbes et al. use a video signal identifier to memorize repetitive television signals in order to automatically control recording of TV programs [5]. However, the commercial has to be identified to the system before it can recognize it. Nafeh proposed a method for classifying patterns of television programs and commercials based on learning and discerning of broadcast audio and video signals using a neural network [10]. However, none of the reported methods used an automatic optimization method for tuning of algorithm parameters and analysis of the parameters behavior. In this sense, we are proposing a fast method for algorithm benchmarking and fine tuning using GAs.

[2] NTSC is the National Television Standards Committee and PAL is Phase Alternating Line. NTSC designates the video standard used in North America and some other countries and PAL is standard for most of Europe.

4 MPEG-RELATED FEATURES

There are different sets of features that are extractable during the MPEG-encoding process. The encoder internal parameters, called low-level features, extracted during the encoding process are:

- frame type indicator, which discriminates between intra-coded (I), predicted (P) and bi-directional (B) frames;
- luminance DC value at macroblock level on I-frames only; A macroblock is a coding layer used in MPEG.
- VTS (video time stamp) of the observed frame, which assures correct synchronization of extracted video features and matching video frames;
- *macroblock correlation factor*, which represents the correlation between the current macroblock and reference macroblock in the reference frame.

A schema of a representative MPEG-2 encoder is shown in Figure 1. The luminance DC values are available in the information chain at point *a* after the *Discrete Cosine Transformation* and the *Quantizer* before the *Variable Length Coding* . These DC values can be used for content analysis algorithms. The *Motion Estimation* encoding block at point *b* generates a spatio-temporal representation of the macroblock motion for an efficient encoding process. This macroblock motion recovery process uses the correlation of an actual macroblock and possible reference macroblocks in the reference frame. The correlation factor of the best match of actual macroblock and reference counter macroblock is tapped for further content analysis.

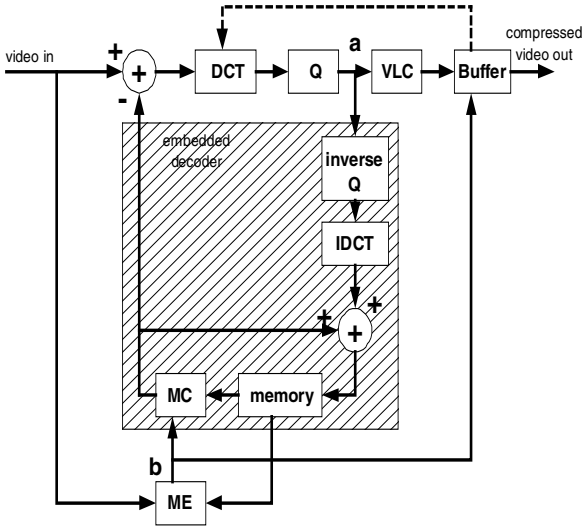


Figure 1: MPEG-2 video encoder.
DCT = Discrete Cosine Transform,
Q = Quantization, VLC = Variable Length Coding,
MC = Motion Compensation, ME = Motion Estimation,
IDCT = Inverse DCT.

The low-level parameters are used to derive more meaningful features, called mid-level features, such as black/unicolor frame, scene change, and letterbox. A

simple threshold technique is sufficient to implement a reliable black frame detection algorithm by using the sum of the luminance DC values of the entire frame (*Luminance DC Summation*) as input value to the algorithm. Luminance DC values can also be used to discriminate between a 4:3 and 16:9 aspect ratio video frame (Letterbox) by similarly evaluating the appropriate the luminance values of upper and lower macroblock slices of the video frame. Unicolor frames are detected by applying a threshold on the average of the absolute differences of luminance DC values between adjacent blocks, on the whole frame. The absolute differences of luminance DC values between adjacent blocks can also be used for black frame and letterbox detection. The use of the variation of the *Luminance DC Summation* over consecutive frames and the *macroblock correlation value* facilitate the implementation of a scene change detection algorithm. The mid-level features such as black frame, letterbox, and scene changes are used for commercial detection.

5 THE COMMERCIAL DETECTOR ALGORITHM

There are different families of algorithms that can be used for commercial detection based on low and mid-level features. In this implementation we have experimented with the available mid-level features:

1. black frame
2. unicolor frame
3. keyframe distance (i.e. consecutive scene changes distance, also denoted as KF distance in the following)
4. letterbox (i.e. 4:3 versus 16:9 aspect ratio discrimination)

All the above values are computed for each I frame. In the first step, the algorithm checks for "triggers" that could flag the possible start of a commercial break. The algorithm, then verifies if the detected segment is a commercial break.

5.1 TRIGGERS

In the current experiments we have used the time interval between detected black or uni-color frames as triggers. Normally, black frames (or unicolor frames) are used by the content creators to delineate commercials within a commercial break, as well as at the beginning and ending of a whole commercial break. We assume that a commercial break starts with a series of black (unicolor) frames and that during the commercial break we will encounter black (unicolor) frames within a predetermined threshold (e.g. 50 seconds). Also, we have placed constraints on the duration of the commercials. We have determined by looking at a number of commercials that commercial breaks can not be shorter than one minute and can not be longer than six minutes. An additional constraint that is derived from the material we have seen is that commercial breaks have to be at least one and a half minute apart. This last constraint is important for the linking of the segments that potentially represent commercials. If the linking is allowed for a long period of time, we might end up with very long commercial breaks, which in fact might contain a commercial break and an action scene from a movie. After some number of black sequences the probability of commercial being present increases and potential commercial end is searched for.

5.2 VERIFIERS

Once a potential commercial is detected, other features are tested to increase or decrease the probability of a commercial break. Presence of a letterbox change or high cut rate expressed in terms of low keyframe distance can be used as verifiers. In the case of letterbox change, the probability that the given area is a commercial break is increased. In the case of low keyframe distance (or high cut rate), the probability of a commercial being present is increased. If the cut rate is below a certain threshold then the probability is decreased. Average keyframe distance is defined as the average shots duration between the last n scene cuts. The threshold used for the keyframe distance can be varied from 6 to 10 for good results. Again, segments which are close by can be linked to infer the whole commercial break. There are commercials such as Calvin Klein which are very slow and this can increase the average cut distance temporarily. We allow for the keyframe distance to be high for 30 seconds before decreasing the probability of being in a commercial break. As with the black frame indicators, we have placed constraints on the duration of the commercials. Other mid-level features can also be extracted from MPEG-2 encoding parameters, and be used as verifiers. Progressive versus interlaced video material changes or coding cost are some examples. As can be seen, there are a number of thresholds that need to be experimented with for algorithm fine tuning. In the next section we explain the experiments that we carried out in order to determine the optimal thresholds for eleven different parameters for best accuracy.

6 THE EXPERIMENTS

We obtained two samples of broadcast video. One contains about 8 hours of Dutch television and comprised 13 different TV programs of various genres including movies, news, sports programs, talk shows, and sitcoms. It contains 28 different commercial breaks, for a total duration of about 1.5 hours of commercials. This video was in PAL format with a GOP[3] of six. We refer to it as the EMPIRE set after the name of the MPEG video encoder chip that extracted its low-level features. The other set contained about 5 hours of US content from 11 different programs including sports, movies, games, talk shows, MTV music videos, and news. It contained 35 different commercial breaks, for a total duration of more

[3] GOP is Group of Pictures and six means that every sixth frame was an I frame.

than 1.5 hours of commercials. This video was in NTSC format with a GOP of 12 and was called the EMPRESS set, again after the encoder chip.

These video sequences were scanned by a human and the beginning and end of each commercial break was marked. Thus we obtained a ground truth value (yes/no) for every video frame. The performance of any instantiated detector algorithm was assessed by counting the number of true positives (TP, commercial frames labeled as such), false positives (FP) and false negatives (FN). From these we computed the usual measures of recall and precision used in pattern recognition:

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

Since we ran experiments with a scalar GA, we experimented with a number of different combinations of these data as fitness metrics. It should be noted that because the commercial detector algorithm has built-in contiguity constraints, isolated error frames do not occur – to be labeled as a commercial, a block of frames must fall between certain minimum and maximum limits.

To guard against any optimization procedure's tendency to overfit the training data, we split the data set into training and test sets. For the EMPIRE data set, we split each TV show roughly in half, approximately 50% of the shows had the data taken from first-half of the show while for the other 50% had the data from the second half used in the training set. A slightly different procedure was followed for the EMPRESS data set: within each genre, shows were paired for similarity (human judgement) and one whole show of each pair was used for training. The test set was always the complement of the training set. When we sought to validate any given detector, we ran the algorithm on the combined test and training sets. This we call the validation set – which was the set used by the engineer when manually tuning the algorithm.

The GA used was Eshelman's CHC [4]. CHC is a generational style GA with three distinguishing features. First, selection in CHC is monotonic: only the best *M* individuals, where *M* is the population size, survive from the pool of both the offspring and parents. Second, CHC prevents parents from mating if their genetic material is too similar (i.e., incest prevention). Controlling the production of offspring in this way maintains genetic diversity and slows population convergence. Finally, CHC uses a soft-restart mechanism. When convergence has been detected, or the search stops making progress, the best individual found so far in the search is preserved. The rest of the population is reinitialized, using the best string as a template and flipping some percentage (i.e., the divergence rate) of the template's bits. This is known as a soft-restart and introduces new diversity into the population to continue search. CHC does not use mutation between restarts. We used the recommended parameter settings (e.g popsize 50, normal triggers for the dropping of the incest threshold and the initialization of soft-restarts [4]) with one exception: the divergence rate was 0.5. This means that a soft restart created a population with one copy of the best-so-far individual and the rest of the population was completely re-randomized. Initial experiments suggested that a lower divergence rate leads to the population quickly stagnating with inferior results.

6.1 EMPIRE EXPERIMENTS

The EMPIRE data were in hand first and had served as the data upon which the commercial detection algorithm was originally developed [3]. Hence, for this data set we had the algorithm developed and tuned by the engineer as a benchmark. As the first experiments performed, we wanted to get an idea what fitness measure would be best to use. The set of algorithm parameters (the genes in the chromosome) for these experiments are listed in Table 1. Note that the last three parameters were used only in experiment 5. Parameter 1= SeparationThreshold, 2= DistForSuccThreshold1, 3= DistForSuccThreshold2, 4= DistForSuccThreshold3, 5= UnicolorInSuccThreshold, 6= MinCommThreshold, 7= MaxCommThreshold, 8= RestartThreshold, 9= BlackIFrameThreshold, 10= UnicolorIFrameThreshold, 11=LowInfoIFrameThreshold.

Table 1: Parameters Used in EMPIRE Experiments

PARAMETER	BITS	MIN-MAX	STEPS	EXPERIMENTS
1	6	100 - 3250	50	1-5
2	3	50 - 225	25	1-5
3	3	50 - 225	25	1-5
4	3	50 - 225	25	1-5
5	4	1 - 16	1	1-5
6	3	500 - 5750	750	1-5
7	2	7000 - 10000	1000	1-5
8	3	250 - 775	75	1-5
9	4	1 - 16	1	5
10	3	100 - 450	50	5
11	4	30 - 105	5	5

The experiments 1-5 are briefly summarized in Table 2 where R stands for recall and P for precision, FP and FN are false positives and false negatives respectively. The only difference among experiments 1-4 was the fitness metric used by the GA for selection: R+P, R*P, FP+FN, 4*FP+FN. There was intuitive reasoning for having multiple fitness metrics backed up by experiments. R*P should be sensitive to either measure being small (both recall and precision are numbers between zero and one). FP + FN seemed a reasonable metric to minimize, but

perhaps in this domain, false positives should be weighted more heavily than false negatives (better to let a commercial through than risk cutting out some TV program content). Consequently, we tried 4*FP + FN. Then the simple R+P was included for completeness. Experiment 5 used the R+P fitness metric for reasons given below and added three additional parameters in an attempt to improve the commercial detector. We should also note that the only reason experiments 2 and 3 did not achieve 30 replications was that power failures caused the computers to crash and we observed that all experiments located chromosomes with the same best performance – 12 replications were obviously enough.

Although all experiments discovered the same best performance level, there was still diversity in the final populations indicating that performance was less sensitive to some parameters than others.

Table 2. Summary of Experiments with the EMPIRE Data Set. R-Recall, P=Precision, FP= Fales Positives, and FN = False Negatives

EXP. NUM	# OF PARAMS	# OF BITS	REPLIC ATIONS	FITNESS METRIC
1	8	27	30	R*P
2	8	27	12	4FP+FN
3	8	27	12	FP+FN
4	8	27	30	R+P
5	11	38	30	R+P

One way to examine the outcomes of these experiments is to look at the non-dominated individuals encountered at any time in each experiment. Figure 2 shows these data in the Precision/Recall space. Since the experimental data were all generated on the training data set, some of the best performers were selected (by hand) and run on the complete set of validation data. These points are shown as + signs in the figure and each is connected by a dashed line to its corresponding training set performance point. While not statistically rigorous, these observations do suggest the region in performance space where those commercial detectors are likely to perform. The performance of the engineer's best manually tuned algorithm is also shown, this time as a * symbol. (located at recall 0.83 and precision 0.95.)

It is interesting to observe that the non-dominated set was identical for experiments 1, 2, and 3 even though they used different fitness metrics. We speculate that this is because the parameter sets that yield these R and P values are readily produced even though the selection pressures on the populations are slightly different. It may also indicate that, although the fitness metrics appear to be different, the ranking of the individuals involved is not different; the discretization of the search space may

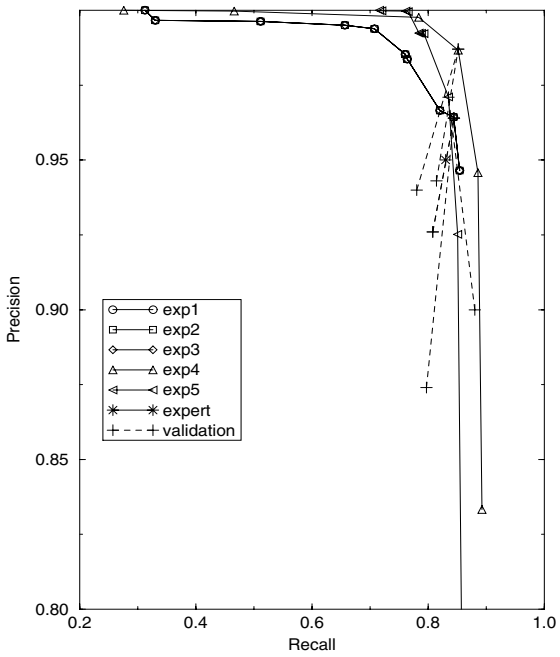


Figure 2. Results from EMPIRE experimental set

simply not permit that much variety. It also appears that the results from experiments 1, 2, and 3 were inferior to those from experiment 4. The comparison of experiment 4 with 1, 2, and 3 caused us to select the R+P metric for all subsequent experiments. The addition of the extra genes in experiment 5 seems not to have provided significant improvement.

The dashed lines between the test and validation performances are designed to suggest the expected intervals for the respective commercial detectors. We do see that the evolved detectors are performing in same region with the best performance achieved by an expert engineer after many months of tinkering.

6.2 EMPRESS EXPERIMENTS

Experiments 6 and 7 were performed with the EMPRESS data set. This data set was acquired after the above work had already been done and may serve as a test of our claim that a robust GA is a valuable tool for adjusting a commercial detector to new data. The parameters used in these experiments are summarized in Table 3. The last column of Table 3 summarizes the location of the best performers in parameter space. We can see that some parameters are very sensitive (all bests have the same allele value) and some are completely insensitive (all permitted values are present among the bests). Parameter 1= SeparationThreshold, 2= DistForSuccThreshold1, 3=

UnicolorInSuccThreshold, 4= MinCommThreshold, 5= MaxCommThreshold, 6= AdjSceneCutsThreshold, 7= AverageCutDistThreshold, 8= CutNumberInAverage, 9= BlackIFrameThreshold, 10= LetterboxLengthThreshold, 11= LetterboxThreshold.

Table 3. Parameters Used in EMPRESS Experiments

PARAMETER	BITS	MIN-MAX	STEP SIZE	EXP NUM	BEST SOLUTION
1	5	100 - 7850	250	6-7	100-7850
2	6	100 - 6400	100	6-7	4500-4700
3	3	0 - 7	1	6-7	0
4	5	100 - 7850	250	6-7	100-850
5	3	7000 - 14000	1000	6-7	13000-14000
6	5	100 - 1650	50	6-7	3
7	4	100 - 850	50	6-7	100-850
8	4	2 - 17	1	6-7	2-17
9	3	80000 - 115000	5000	6-7	95000
10	5	10 - 320	10	7	150-160
11	5	7500 - 32000	500	7	20500-21500

Experiments 6 and 7 conducted for the EMPRESS dataset are briefly summarized in Table 4 and the comparable results are shown in Figure 3.

Table 4. Summary of Experiments with the EMPRESS Data Set

EXP. NUM	# OF PARAMS	# OF BITS	REPLICATIONS	FITNESS METRIC
6	9	38	30	R+P
7	11	48	30	R+P

Figure 3 tells two intersecting tales. Experiment 6 was the first run on the EMPRESS data set. Independently, the expert searched for (using the entire validation set) and reported a detector whose performance on the validation data is shown as the leftmost asterisk in the figure. We see that the GA located points that look superior to the expert's, but these points may represent overfittting to the training data. When we run two (manually-selected) of the GA's detectors on the validation set, we see that performance deviates and no longer dominates the expert's point. At this point, we experimented with additional features that assess the likelihood that the video frame is in letterbox format. Believing this may be valuable additional information for commercial detection, the detector algorithm was augmented with logic that considered this new feature. The expert, being in possession of the experiment 6 results, used them as the starting point for an effort to discover good parameter settings for the new letterbox thresholds. The result of this effort (again using the entire validation set) is the righmost asterisk in Figure 3. In experiment 7 the GA searched the augmented threshold set. Two validation tests from this run are also shown. We see that the GA located the best performance seen to date. We believe this shows the potential of the GAs for this emerging application.

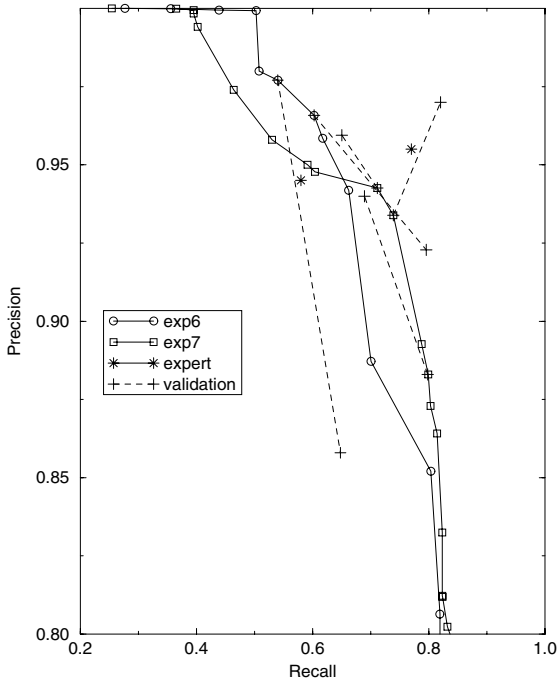


Figure 3. Results of the EMPRESS experimental set

7 DISCUSSION

We have presented results from first experiments using a GA to fine tune the parameters for a commercial detection algorithm for digital video.

We propose that this approach is particularly well suited to this domain because 1) the complexity of the mapping from algorithm parameters to accuracy is unknown, but unlikely to be very simple, 2) it is unlikely that the best parameter sets to use will be the same from culture to culture or for different video standards, 3) the low-level

features available from different encoder chips is likely to change and 4) the mapping is unlikely to remain unchanged over time (perhaps because of deliberate attempts to avoid automatic detection). These properties all suggest that an automated method to readjust the detector algorithm will be needed in the industry. Dynamically downloading new parameters to products in the field is already being practiced.

From our experiments we learned that we can benchmark and fine tune the performance of the commercial detection algorithm rapidly for a new set of data. This means that we used the GAs as a tool for mapping from algorithm parameters to accuracy. In addition, this allowed us to experiment with new parameters (e.g. letterbox in experiment 7) and obtain an optimized version of the algorithm without spending additional weeks of effort.

The class of algorithms explored was limited, constrained by a desire to use algorithms that could be driven by low-level features immediately available from MPEG encoder chips and that could run on the modest computing resources available in today's consumer electronic products. In addition, only a limited amount of video material was available for these initial experiments.

There are several directions in which to continue this work. Clearly more validation work is needed before the utility of the detectors can be assessed against levels needed for consumer acceptance. In addition, a broader class of detector algorithms could be explored, including allowing the GA to explore this dimension in addition to simply tuning parameters. A truly multi-objective GA also should be tried.

References

1. D. W. Blum, "Method and Apparatus for Identifying and Eliminating Specific Material from Video Signals, " US patent 5,151,788, September 1992.
2. Edgar L. Bonner and Nelson A. Faerber, "Editing system for video apparatus," US patent 4,314,285, February 1982.
3. N. Dimitrova, S. Jeannin, J. Nesvadba, T. McGee, L. Agnihotri, G. Mekenkamp, "Real time commercial detection using MPEG features," Information Processing and Management Uncertainty in Knowledge-based System, IPMU 2002, Annecy, France, July 1-5, 2002.
4. L.J. Eshelman, "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," Foundations of Genetic Algorithms, Gregory Rawlins (ed.), Morgan Kaufmann, San Mateo, 1991.
5. S. J. Forbes, F. F. Forbes, "Video Signal Identifier for Controlling a VCR and Television based on the Occurrence of Commercials," US patent 5,708,477, Jan. 13, 1998.
6. J. Iggulden, K. Fields, A. McFarland, J. Wu, "Method and Apparatus for Eliminating Television Commercial Messages, " US Patent 5,696,866, Dec. 7, 1997.
7. Y. Li and C.C.J. Kuo, "Detecting commercial breaks in real TV programs based on audiovisual information," Proc. Of SPIE The International Society for Optical Engineering (USA), vol.4210, p.225-236, 2000.
8. R. Lienhart, C. Kuhmunch and W. Effelsberg, "On the Detection and Recognition of Television Commercials," in Proc. Of IEEE International Conference on Multimedia Computing and Systems, pp. 509-516, 1997.
9. T. McGee, N. Dimitrova, "Parsing TV programs for identification and removal of non-story segments," SPIE Conference on Storage and Retrieval in Image and Video Databases VII, vol. 3656, pp. 243-251, San Jose, 1999.
10. J. Nafeh, "Method and Apparatus for Classifying patterns of Television Programs and Commercials Based on Discerning of Broadcast Audio and Video Signals," US patent 5,343,251, Aug. 30, 1994.
11. A. P. Novak, "Method and System for Editing Unwanted Program Material from Broadcast Signals," US patent 4,750,213, Jun. 7,1988.

An Application Service Provider Approach for Hybrid Evolutionary Algorithm-based Real-world Flexible Job Shop Scheduling Problem

Ivan T. Tanev
Synthetic Planning Industry Co.Ltd.
HK Building 2F, 2-21-10 Nishiogikubo,
Suginami, Tokyo 167-0053,
Japan
i.tanev@computer.org

Takashi Uozumi
Dept. of Computer Science and
System Engineering,
Muroran Institute of Technology,
Mizumoto 27-1, Muroran 050-8585,
Japan
uozumi@csse.muroran-it.ac.jp

Yoshiharu Morotome
Synthetic Planning Industry Co.Ltd.
HK Building 2F, 2-21-10 Nishiogikubo,
Suginami, Tokyo 167-0053
Japan
moro@spi-sys.co.jp

Abstract

This paper presents an approach for scheduling of customers' orders in factories of plastic injection machines (FPIM) as a case of real-world flexible job shop scheduling problem (FJSS). The objective of discussed work is to provide FPIM with high business speed which implies (a) providing a customers with convenient way for remote online access to the factory's database and (b) developing an efficient scheduling routine for planning the assignment of the submitted customers' orders to FPIM machines. Remote online access to FPIM database, approached via delivering the software as a Web-service in accordance with the application service provider (ASP) paradigm is proposed. As an approach addressing the issue of efficient scheduling routine a hybrid evolutionary algorithm (HEA) combining priority-dispatching rules (PDRs) with GA, is developed. An implementation of HEA as a database stored procedure is discussed. Performance evaluation results are presented. The results obtained for evolving a schedule of 400 customers' orders on experimental model of FPIM indicate that the business delays in order of half an hour can be achieved.

1 INTRODUCTION

Until recently the role of the production factories had been associated with the manufacturing of a high volume of low-cost and high-quality goods. However, an evolution of these features is lately observed as a result of the recently emerged trend in the major world's economies of decreasing the rate of economic growth. Still maintaining the importance of producing low-cost and high quality goods, the relevance of the high manufactured volume is going to be gradually replaced by the role of the high business speed – the ability to react quickly in submitting and modifying the customers' orders. The high business speed implies that factories should provide the customers with services such as remote submission of orders in operative mode; prompt feedback to allow for customers' awareness about the anticipated due dates of their orders as well as about the expected ratio of tardy orders and their respective delays; and tracking the state of the submitted orders.

Within this context, the objective of our research is to investigate the feasibility of developing a scheduling system for FPIM, emphasizing on providing the mentioned above customers services needed for achieving factory's high business speed. Fulfilling the objective implies addressing of the following two main tasks. First, allowing for submission of orders and tracking their statuses requires providing a convenient way for remote online access to the factory's database. And second, allowing for prompt customers' awareness about the anticipated due dates of their orders assumes developing of efficient (both in terms of runtime and quality of solution) scheduling routine for planning the assignment of the submitted customers' orders to the factory's machines. Our work is intended to address these main tasks, and its contents could be viewed from three different aspects, representing the following three layers of abstraction of the proposed scheduling system:

- Problem aspect – the task from the specific problem domain intended to be solved,
- Aspect of algorithmic paradigm – the algorithmic paradigm employed to solve the problem,
- Implementation aspect – the system architecture used to solve the problem exploiting the adopted algorithmic paradigm.

The discussion, presented in this document, is attempting to highlight these aspects of our work, and the remaining of the paper is structured as follows. Section 2 briefly explains the problem aspect – a real-world problem of scheduling of FPIM as an instance of the class of FJSS. Section 3 discusses the aspect of algorithmic paradigm – the main attributes of the hybrid evolutionary algorithm we developed to solve the targeted FPIM FJSS. Section 4 considers the implementation aspect – the ASP

approach, focusing on developing of three-tiered Web-based system architecture. Performance evaluation results are given in Section 5. Finally, Section 6 draws a conclusion and discusses some directions for future work.

2 REAL-WORLD CASE OF INJECTION MACHINES SCHEDULING

The FPIM FJSS problem consists of a finite set of orders to be processed on a finite set of machines. Each order specifies the amount of just one good from the finite set of goods, produced by the factory. Each good can be produced using any of currently available molds from the finite set of mold instances of at least one of the finite set of the available mold types. Each mold type can be attached to at least one machine from the available finite set of machines. The one-to-many relationship between the goods and molds, and between the molds and the machines implies that any order can be processed in at least one machine. In general, processing the order on specified machine is preceded by the set-up phase, needed to attach the required mold (if mold of the current order differs from the previous one) and to change the resin (if needed). Analogically, the processing of the order might be followed by completion phase, required to remove the mold in case that the next scheduled order requires an attachment of different mold type.

The capacity *constraints* specify that each mold can be attached to just one machine at a time and each machine can attach just one mold. Consequently a machine can process only one order and each order can be processed by only one machine at a time. An additional constraint stipulates that the amount of the molds of specified type is limited; therefore an order can be processed only if the required mold is currently available. Also, the machines can be suspended for scheduled maintenance and for daily operation breaks. The order cannot be preempted by another order, however, depending on the specified machine operation mode, the orders, started before the suspension time should be interrupted upon the commencing the maintenance interval or might be allowed to complete within the maintenance interval. In the former case, the processing of the interrupted orders resumes upon resuming the operations of the corresponding machine.

The *objective* of the scheduler is to determine the processing starting time, the processing machine, the mold and the mold type for each order, obeying the imposed constraints and minimizing the ratio of tardy jobs, the variance of the flow time, the amount of mold changes, and maximizing the efficiency of the machines. The schedule is viewed as a table of rows each including creation date/time, customer's name, customer's order, processing machine, mold, mold type, starting and finishing times for setup, processing, and completion phases respectively.

In our approach, the FPIM data about orders, manufactured goods, available resins, mold types, molds, machines and operation patterns constraints are organized as an entities (tables) in relational database. The entity-relationship diagram for FPIM database is shown in Figure 1. The mechanisms used to access the data in FPIM database is elaborated later in Section 4.

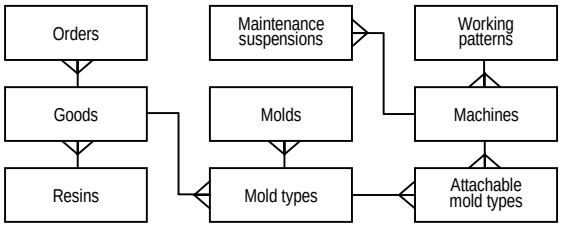


Figure 1: Entity-relationship Diagram for FPIM Database

The main differences between the theoretical models and the real-world case of FPIM FJSS are the availability of the setup and completion times, which depend on previous and next order respectively; the availability of maintenance time and operational break time; order interruption and restart; and limited and dynamically changeable amount of available machines and molds. These differences additionally complicate the concrete instance of the FJSS. The latter, being NP-hard is well known to be a notoriously difficult to solve. Such an additional complication affected our choice of algorithmic paradigm intended to solve the FPIM FJSS, as elaborated in the following Section 3.

3 HYBRID EVOLUTIONARY ALGORITHM FOR FPIM FJSS

In our approach we propose a hybrid evolutionary algorithm, which combines the approaches of using PDRs with GA (Holland, 1975; Goldberg, 1989; M.Varquez and L.D.Whitley, 2000). A PDR is a rule that is used to determine which order is to be executed next, from the list of unscheduled orders. Compared with other approximation approaches, PDR-based approaches offer the advantage of simplicity, featuring low computational cost and can therefore be applied to complex real-world problems such as FPIM FJSS. They are usually temporally local without trying to predict the future. Instead, making decisions based on the present; they are very useful in factories such FPIM, where the future availability of the resources (machines, molds, etc.) is very unpredictable. The main disadvantage of PDR is their myopic nature: often the quality of the overall schedule, build using locally applied PDR is far from optimal. In addition, no single PDR can be success-fully applied for the whole range of possible cases of FJSS (Pierreval and N. Mebarki, 1997). This might require to empirically evolve the PDRs and their combination, which are most suitable for the concrete instance of FJSS. In order to address the disadvantages of PDRs we propose a hybrid evolutionary algorithm (HEA) as a combination of PDRs with GA. GA is used as a way to empirically evolve the most suitable combinations (strings) of PDRs for the considered FPIM

FJSS. Also, GA is intended to address the myopic nature of PDRs given that GA is based on the survival of the overall fittest individuals (i.e. schedules), rather than these with better local features. The only concern of combining GA with PDR-based sys-tem into HEA for solving real-world problem is whether GA would invalidate the advantage of PDR of being less time consuming. However, despite the longer computational times of GA, we believe that the GA might be successfully applied in FPIM FJSS since it can be considered as a form of anytime algorithm (Ciesielski and Scerri, 1998), and as a result, a feasible tradeoff between the runtime and the desirable quality of schedule could be easily maintained.

It is generally accepted that the GA feature five basic attributes: genetic representation of solutions to the problem; way to create an initial population; 'genetic' operators that alter the genetic population; evaluation function and values for the parameters. The remaining of the current Section is intended to elaborate on these basic attributes of GA.

3.1 THE GENETIC REPRESENTATION OF SCHEDULES. THE INITIAL POPULATION

There are two basic approaches for genetic representation, which can be applied for FPIM FJSS: direct and indirect. Direct representation encodes the schedules as chromosomes, and genetic operations are used to evolve the population of such chromosomes into better schedules. In the indirect encoding schemes a sequence of schedule-building instructions is encoded ("generative encoding") in the chromosome (Fang *et al*, 1994; O'Neill and Ryan, 2000). The genetic operations are used to evolve the population of such sequences of schedule-building instructions into ones that generate better schedules. Considering the complexity of various constraints imposed on FPIM FJSS, it is highly likely that direct representation of chromosome would yield unfeasible schedules, i.e. schedules that violate some of the constraints. The repairing, needed in such cases might be inefficient in both that it requires additional runtime and tends to break the developed building blocks of the solutions. In addition, dynamic nature of some of the constraints (as, for example, the limited amount of molds) assumes that obeying them (i.e. using mold that currently is not being used by other orders) requires corresponding runtime verification on the build-so-far schedule. These concerns indicate that the eventual direct encoding is impractical for the concrete case of FPIM FJSS. In the proposed approach a PDR-based indirect representation of the schedule is used, where the allele in chromosome represent the PDR used for assigning the order to the specified machine. Each chromosome (the genotype) is represented as a string $'g_0, g_1, g_2, \ldots'$ which is mapped into the corresponding schedule (the phenotype) by schedule builder during the chromosome evaluation phase of HEA. Each of the genes g_i of the chromosome $'g_0, g_1, g_2, \ldots'$ is interpreted by schedule builder as follows: "for the currently becoming free machine m_k, select all the unscheduled orders that can be currently processed on m_k and range them in accordance with the g_i–th PDR; then select the first order o_j from the arranged list of unscheduled orders and assign o_j to m_k". The following nine PDRs have been used: FIFO (also known as AT- arrival time, and TIS- time in the sys-tem); FIFO SM – the same as FIFO but trying the same mold (SM); FIFO SMR – the same as FIFO but trying the same mold and same resin; SPT – shortest processing time; LPT – longest processing time; DT – order due time; DT SM – the same as DT but trying the same mold; DT SMR – the same DT but trying the same mold and same resin; and ST – order start time.

The preliminary comparative results of convergence of the fitness of best individuals for typical runs of HEA and GA without PDRs confirm the advantages of incorporating PDRs into GA. The results indicate that in contrast to the GA without PDRs, HEA features much faster fitness convergence with better values of absolute fitness. The desirable schedules (schedules with no tardy jobs) are evolved relatively quickly by HEA within several generations.

Initial population is created by generating a $(N_{PS}-2)$ chromosomes where N_{PS} is the population size. The genes of each of these chromosomes are set to a random numbers within the range $(0, N_{PDR}-1)$ where N_{PDR} is the total amount of used PDRs. Two additional chromosomes are created, alleles of which contain a single PDR only – FIFO and DT respectively, in order to allow for the HEA to quickly find the solution in some trivial scheduling cases. Note that due to the adopted indirect genetic representation the process of creating initial population always generates feasible schedules only, where no constraints are violated.

3.2 GENETIC OPERATORS

The main genetic operators are selection, crossover, and mutation. In our work we used binary tournament selection – a robust, commonly used selection mechanism, which has proved to be efficient and simple to code. In addition, it results a selection pressure that provides a good convergence rates yet avoiding premature convergence to a sub-optimal solutions. The canonical two-point crossover operation is employed. The mutation operation changes the genes from each chromosome with specified probability to the value within the range $(0, N_{PDR}-1)$, where N_{PDR} is the total amount of used PDRs.

3.3 EVALUATION FUNCTION

The evaluation function estimates the fitness of the chromosomes (respectively, the schedules they generate) by measuring the severity of constraints violation and the extent of approaching the scheduling objectives. In our approach all the imposed constraints are considered as hard in that on neither stage of HEA they are violated. Regarding the objectives, applying the heuristics rule that from the customer viewpoint any schedule containing tardy orders, is worse (feasible, but undesirable schedule)

that schedule that do not have ones (desirable schedule), we consider an evaluation function that allows HEA to clearly distinguish the desirable schedules from undesirable ones. In our approach the evaluation function maps the fitness of all the desirable schedules within the range of $(0,Q)$ while maintaining the fitness of undesirable schedules within the $(Q,+\infty)$. For both the cases the lower values of the fitness correspond to the better schedules. The evaluation function for desirable schedules $eval_D(x)$ can be expressed as follows:

$$eval_D(x)=eval(x)$$

where $eval(x)$ is a sum, normalized to 100, of the ratio of tardy jobs, the variance of the flow time, the relative amount of mold changes, and complement to one of the efficiency of the machines usage (as a ratio of the sum of setup and completion time to the order processing time). Respectively, the evaluation function for undesirable schedules $eval_U(x)$ is defined as

$$eval_U(x)=eval(x)+Q$$

where Q is the penalty for schedule having at least one tardy order. The penalty value should fulfill the condition $Q> max(eval_D(x))$, and in our approach $Q=101$.

3.4 VALUES OF PARAMETERS

The values of parameters are as follows. Population size is 20 individuals (chromosomes), selection method is binary tournament with elitism where selection and elitism ratio are 0.2 and 0.1 respectively, and mutation rate is 0.01. The termination criteria are runtime, fitness of the best of individuals, or number of generations. Notice the relatively small population size. The results of parameters tuning experiments indicate that varying the population size yields negligible small variance in computational effort of developed HEA. Smaller population sizes reduce the runtime for evolving a single generation, and consequently, to allows for the authorized user to quickly intervene in the evolution process if needed.

4 IMPLEMENTATION

As we mentioned before, achieving our objective of developing FPIM FJSS that features high business speed implies the addressing of the task of providing the customers with convenient way for remote access to the FPIM data. Considering the Web as most favorable deployment platform due to its ubiquitous nature, this task could be decomposed into the following two problems: how to implement the HEA on the Web, and how to make the FPIM database (including the schedules, build as result of HEA functionality) available on the Web. Regarding the implementations of HEA on the Web, the developed-so-far approaches of using Internet as a deployment environment for EA are exclusively focused on the issue of parallel, distributed implementation of EA (Chong, 1999; Tanev *et al*, 2001), improving the computational speed of the latter. As a result the issue of incorporating the adequate user interface providing remote access to the real-world problem-related databases is not considered as relevant in these approaches. In addition, taking into consideration the distributed nature of the Internet-based implementations of EA in these methods, their eventual straightforward use for the considered case of FPIM FJSS would feature considerable performance degradation of the HEA due to heavy data traffic due to the need for the distributed entities of these architectures to intensively access the centralized FPIM business data (shown in Figure 1) during scheduling. The volume of such data for the moderately scaled FPIM might be in order of few hundreds of Megabytes, which also proves the unfeasibility of the idea of downloading such data (caching) for intended future local use by HEA.

To address the first of the mentioned problems – providing Web-access to the FPIM FJSS we used the ASP-based approach. And for the problem of efficient implementation the HEA on the Web we employed a method of implementing HEA as a database stored procedure (SP). An additional motivation for considering SP as a way to implement HEA is that to our best knowledge, we are not aware about any work regarding implementing EA as a SP, and we were interested about the feasibility for applying such an approach for the considered case of real-world FPIM FJSS problem. The remainder of the Section elaborates the approaches we propose to address these two problems.

4.1 THE APPROACH OF ASP

ASPs are a recently emerged way to sell and distribute software and software services via Internet. In most cases the ASPs can be viewed as companies that supply software applications and/or software-related services over the Internet. The significant advantages offered both to the factories and to their customers by providing the web-access to business solutions instead of using the traditional model to physically deliver the required specialized applications are the low cost of entry, considerably less expensive pay-as-you-go model, and shifting the Internet bandwidth to the ASP, who can often provide it at lower cost. Implementing FJSS as ASP allows the FPIM to focus on its core competencies instead of managing the complexities of today's IT infrastructure. In addition, ASP significantly alleviates the problem related to the maintenance of complex software system and the need for software upgrades. The eventual distribution of corresponding “fat”-clients to the hundreds of customers for the real-world instance of FPIM FJSS would become extremely expensive both from FPIM and customers standpoints; and the need for future upgrades deteriorates the problem even more.

The three-tiered system structure incorporating Web-browser (as thin client), Web-server, and database server is widely adopted as a de facto standard for building applications using ASP paradigm. Following the common trend, we adopted the three-tiered architecture (Figure 2) with the following functionality of the main entities.

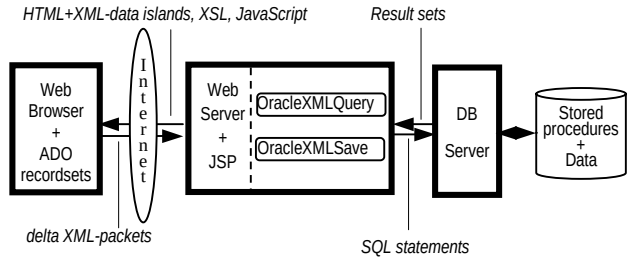


Figure 2: Three Tiered Architecture For ASP-Based Implementation Of FPIM FJSS

4.1.1 Web-browser

Web-browser represents the client-side functionality of developed ASP-based approach for FPIM FJSS. Depending on the access rights, two separate user roles for clients are defined: customers, and factory users. Customers are allowed to submit the orders in operative mode, accessing three main FPIM data entities: their own orders (for updating), the manufactured goods (for reading only), and the schedule, generated for their own orders. The factory users are granted with full access allowing reading and updating of all the available data in FPIM database. In addition, factory users are allowed to initiate the HEA for creating schedule of all orders, including recently submitted and still unscheduled ones. For both types of user roles maintaining adequate user interface was considered as a crucial issue in developing the Web-client side of FPIM FJSS. We use an ActiveX Data Objects (ADO) recordsets incorporated into Web-browser for maintaining the data obtained from FPIM database. ADO-recordsets offer a way to adequately handle the database tables by the browser. The FPIM database data are received by Web-browser as XML-data islands within HTML-pages. In order to minimize the network traffic, and consequently, to provide better scalability characteristics of the system, the Web-browser updates the data in offline mode in that all the changes of ADO-recordsets are buffered on client side in a form of delta XML-packet. Using a single HTML-form submission, the delta XML-packet is forwarded to the Web-server, which performs all the accumulated updates in batch mode. The functionality of Web-browser, including browsing and updating the ADO-recordsets, maintaining a XML-delta packet, managing the master-detail and lookup relationships in FPIM database is accomplished by specially developed API written in JavaScript. Figure 3 depicts the snapshot of the client screen for browsing/updating the FPIM database table containing mold types and the corresponding detail table of machines these mold types could be attached to. Figure 4 shows the screen snapshot of viewing the schedule of all orders. In both cases the screen snapshots correspond to user role of factory user.

4.1.2 Web-server

The Apache Web server, used in proposed implementation of FPIM FJSS employs the Java Server Pages (JSP) technology to provide the browsers with the content, dynamically generated in result of FPIM database access. JSP incorporates both formatting, static HTML-tags, which are directly passed back to the response page, and Java scriplets that are dynamically executed by Web-server and the result of their execution is incorporated into the response page. The scriplets included in JSP call the application logic components for database access. In our approach we use `OracleXMLQuery` and `OracleXMLSave` Java classes for accessing the FPIM database. The former is used by JSP for serving the request from Web-clients for displaying the contents of corresponding tables in FPIM database. Upon activation by JSP, it submits a corresponding SELECT SQL-statement against FPIM database and returns the XML-encoded result set. The latter is then incorporated into the response page and forwarded to Web-browser as an XML-data island. The `OracleXMLSave` class is used by JSP for updating the FPIM database with the changes made by Web-clients. `OracleXMLSave` accepts the delta XML-packet, parses it, generates the corresponding set of INSERT, UPDATE and/or DELETE SQL-statements, and finally submits these statements to the FPIM database for their execution.

4.1.3 Database Server

As we stated before, the FPIM-data containing the created schedule, submitted orders, available machines, molds, mold types, resins, manufactured goods etc. are organized as an entities of a relational database system. In our implementation we use Oracle 8.1.7 database server as a platform, well known with its performance, scalability, reliability, providing adequate data security and integrity. It offers seamless integration with the adopted JSP-technology providing a sufficient set of Web-server side deployed application logic components (such as `OracleXMLQuery` and `OracleXMLSave`).

4.2 IMPLEMENTING HEA AS DATABASE STORED PROCEDURE

Few ways to implement and deploy the HEA on the Web exists depending on which entity of system structure (Web-client, Web-server, or database server) runs the HEA code. In our approach HEA is developed using Oracle PL/SQL programming language and stored on database server as a stored procedure (SP). Database server also handles the execution of SP. The benefits of implementing HEA as SP are improved performance – database server compiles SP once and then reruns the compiled execution plan; minimized interconnection network overhead – SP reduces the eventual long sequences of SQL statements into a single line, and enhanced security - Web-clients are granted with permission to execute a HEA SP independently of underlying table permissions. The functionality of HEA SP includes code, organized in two routines:

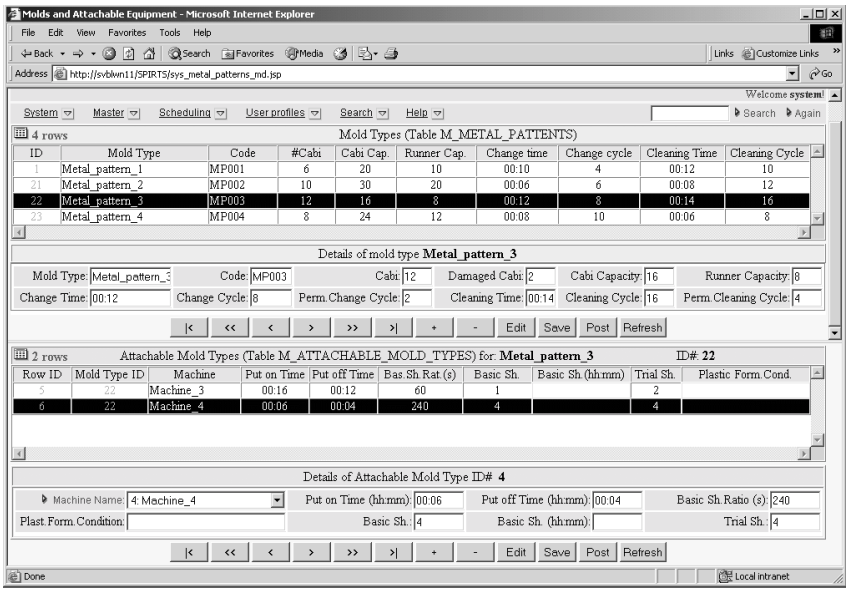


Figure 3: A Snapshot Of The Client Screen For Browsing/updating The FPIM Database

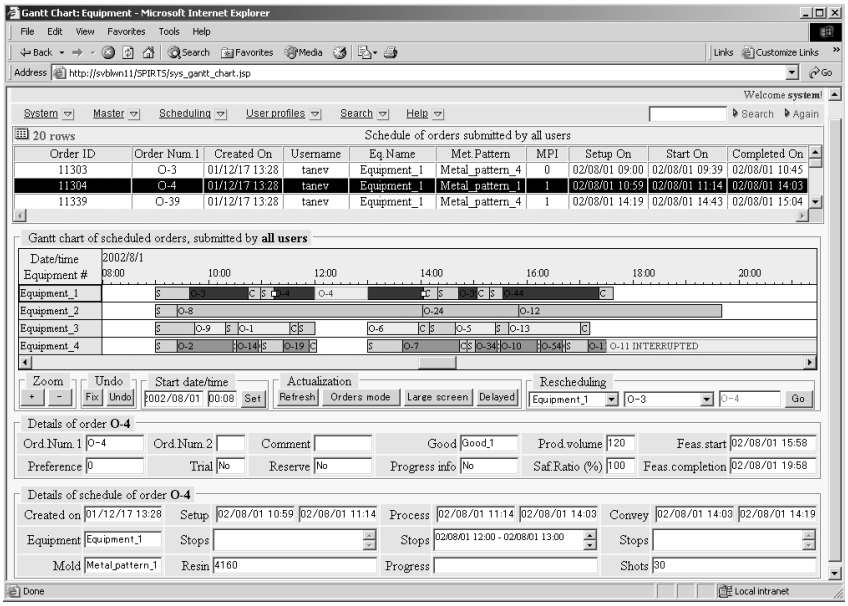


Figure 4: A Snapshot Of The Client Screen During Viewing The Schedule

- Routine which performing the main genetic operations evolves the population of chromosomes, and
- Evaluation of the fitness of the chromosomes.

The first of the routines implements the canonical GA. The routine of fitness evaluation incorporates the schedule builder that maps each of the chromosomes into corresponding schedule and evaluates it using the evaluation function as described earlier in Section 3.3. The mapping itself includes the scanning of all the genes in chromosome $'g_0, g_1, g_2, \ldots'$ and applying the mapping rule (as elaborated earlier in Section 3.1) for each of the genes g_i in accordance with the following steps:

- Step 1: Defining the currently becoming free machine m_i, and the instance t_k when it will be available,
- Step 2: Selecting all the unscheduled orders that can be currently processed on m_i at t_k and range them in accordance with the g_i–th PDR,
- Step 3: Acquiring the first order o_j from the given list of unscheduled orders and assign o_j to m_i".

While the first step requires only an access to the FPIM database table of so-far-generated schedule, the following two steps require intensive database access (shown in Table 1 and Table 2 respectively). As a result, the fitness evaluation routine consumes more than 95% of HEA runtime. The performance evaluation results are discussed in the following Section 5.

Table 1: Defining The Set Of Orders That Can Be Scheduled On Machine m_i At Instance t_k

STEP	DB TABLE	INFORMATION ACQUIRED
2a	Machines	Machine m_i which becomes free and should be scheduled at instant t_k
2b	Working patterns	Acquiring whether t_k is within the defined working pattern for considered day
2c	Attachable mold types	Set of mold types $\{MT\}$ that can be attached to m_i
2d	Mold types, Molds	Set of molds instances $\{MI\}$ of types $\{MT\}$ that can be attached to m_i and are not being used by other machines at the same instant t_k
2e	Goods	The set of goods $\{G\}$ that can be produced using $\{MI\}$ of types $\{MT\}$
2f	Orders	The set of orders $\{O\}$, which request the production of $\{G\}$

Table 2: Assigning Order o_j At Instance t_k On Machine m_i Using Mold Type MT_m

STEP	DB TABLE	INFORMATION ACQUIRED
3a	Orders	Manufactured volume, manufactured good, trial shots (Yes/No) for order o_i
3b	Mold types	Change time, change cycle (in shots), cleaning time, cleaning cycle for mold type MT_m
3c	Attachable mold types	Put-on time, put-off time, shot time interval, #trial shots for MT_m when attached to m_i
3d	Resins	Change time for the resin R_p, used for production of good G_q, requested by order o_i
3e	Machines	Working pattern consideration mode for machine m_i
3f	Working patterns	Current working pattern for machine m_i which wraps t_i

5 PERFORMANCE EVALUATION

The performance evaluation results have been experimentally obtained for the developed prototype of FPIM FJSS deployed on system with the following configuration. Apache Web-server and Oracle 8.7.1 database server are running on the same Hitachi Flora 370 featuring 450 MHz Pentium II CPU with 128 Mbytes of main memory running W2000 Professional Edition. The Web-client is Microsoft Internet Explorer Version 6.0 running on same type of computer as servers. Client and servers are connected in 100 Base-TX LAN via Hitachi Summit-48 hub.

The task of scheduling feature 400 orders to be scheduled in the experimental model of FPIM that produces 4 different types of goods using 4 machines. Each of the good can be produced with 2 of the totally 4 available mold types, and each of the mold types can be attached to 2 of the totally 4 available machines. There are 2 molds available for each molds type. The working patterns for each of the machines are defined as 9:00 - 12:00 and 13:00 - 17:30, with day-offs on Saturdays and Sundays. In addition, the working patterns are considered as “hard” for two of the machines implying that the being processed orders which are unable to complete beyond the scope of off time should be interrupted and resumed later when the corresponding machine resumes operation. Figure 4, presented earlier depicts a possible solution to the considered case of FPIM FJSS.

The estimated computational performance of HEA is about one individual (mapped into schedule with 400 orders) per 13 seconds, or 32 order trials per second. The overall performance of HEA depends also on computational effort needed to solve the FPIM FJSS. We adhered to the approach suggested by (Koza, 1992) which defines the notion of computational effort as an amount of individuals to be processed in order to solve the problem with specified probability (e.g. 90%). The diagram of the probability of success R_S for FPIM FJSS, build from the data of 50 independent runs is shown in Figure 5. The values of HEA parameters are as stated in 3.4. The termination criterion is fitness of the best individual is less or equal to 100 (desirable schedule).

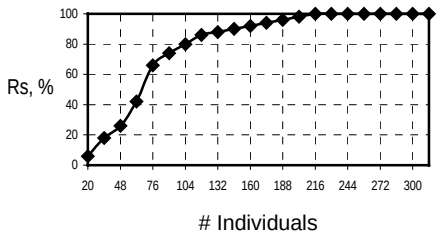


Figure 5: Computational Effort Of HEA

As Figure 5 illustrates, the 90% probability of success in developing a desirable schedule is achieved when processing 146 individuals, which, considering the computational performance of HEA would require about

30 minutes of runtime. This runtime can be viewed as a business delays for the task of evolving a desirable (without tardy orders) schedule of 400 customers' orders in the considered experimental model of FPIM.

6 CONCLUSION AND DIRECTIONS FOR FUTURE WORK

We proposed an approach for solving the problem of scheduling the customers' orders in FPIM as a case of real-world JSSP. The objective of our work is to provide the FPIM with high business speed implying addressing of the following two main issues: (a) providing a convenient way for remote online access to the factory's database and (b) developing an efficient (both in terms of runtime and quality of solution) scheduling routine for planning the assignment of the submitted customers' orders to the FPIM machines. The first issue is addressed by the proposed approach of delivering the software as a service in accordance with the ASP paradigm, which offers the benefits of easy software maintenance and future upgrade, low cost of entry into the business (especially for small and medium scaled FPIM), and considerably less expensive pay-as-you-go model. The issue of efficient scheduling routine is addressed by developed HEA which combines the approaches of using PDR with GA. PDR-based approaches offer the advantage of simplicity, featuring low computational cost and can therefore be applied to complex real-world problems such as FPIM FJSS. GA, incorporated into proposed HEA addresses the issues of the myopic nature of PDR and the necessity to empirically evolve the most suitable PDRs and their combination. Implementing HEA as a database SP offers the benefits of reduced communication network overhead and improved performance characteristics Performance evaluation results obtained for evolving a desirable (without tardy orders) schedule of 400 customer's orders on experimental model of FPIM indicate that the business delays are in order of half an hour.

We are intending to explore the following two approaches to future reduce the business delays. The first approach is aimed at reducing the computational effort of HEA and it would exploit the continuous nature of the scheduling process. Taking into consideration the empirical observation that newly submitted orders are unlikely to be scheduled in a way that requires significant modifications to the orders, scheduled earlier, we are interested in the feasibility to incorporate few of the best schedules from previous run into the initial population of the current run. The second approach is intended to improve the overall performance of HEA by inducing a noise (Miller and Goldberg, 1995) in fitness evaluation - instead of creating and evaluating the whole schedule, it is much faster to create and evaluate only the initial part of it and to make a judgment about the fitness of the whole schedule. The preliminary obtained results are encouraging in that varying the amount of the induced noise a tradeoff between the improved computational performance and the deteriorated computational effort can be achieved, leading to the better overall performance of HEA.

Acknowledgements

We would like to thank the staff of Sumitomo Heavy Industries Ltd. and NEC involved in this project for providing the data and for disclosing the details about the considered case of real-world FPIM scheduling.

References

F. S. Chong (1999), Java based distributed Genetic Programming on the Internet, *Technical Report CSRP-99-7*, The University of Birmingham, Birmingham, UK, "ftp://ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/papers /p.chong/p.chong.msc.25-sep-98.ps.gz

V. Ciesielski and P. Scerri (1998). Real Time Genetic Scheduling of Aircraft Landing Times, *The IEEE International Conference on Evolutionary Computation* (ICEC98), David Fodel, ed, Anchorage, Alaska May 4-9

H.-L.Fang, P.Ross, and D.Corne (1994). A Promising Hybrid GA/heuristic approach for open shop scheduling problems. In A. G. Cohn, editor, *Proceedings of ECAI-94*: 11th European Conference on Artificial Intelligence, pages 590--594. John Wiley and Sons Ltd.

D. E. Goldberg (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, MA: Addison-Wesley, Reading.

J. H. Holland (1975). *Adaptation in Natural and Artificial System*s, The University of Michigan Press, Ann Arbor.

J. R. Koza (1992*). Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press.

B.L. Miller and D. E. Goldberg (1995). Genetic algorithms, tournament selection, and the effects of noise, *Illigal Report No. 95006*, University of Illinois.

M. O'Neill and C. Ryan (2000). Incorporating gene expression models into evolutionary algorithm 2000, *In Proceedings of the workshops of Genetic and Evolutionary Computing Conference 2000 (GECCO 2000)*, 167-172, Las Vegas, Nevada, USA, July 10-12.

H. Pierreval and N. Mebarki (1997). Dynamic Selection of Dispatching Rules for Manufacturing System Scheduling, *International Journal of production Research*, 35 (6): 1575-1591.

I.Tanev, T.Uozumi, and K.Ono (2001), Scalable Architecture for Parallel Distributed Implementation of Genetic Programming on Network of Workstations, *Journal of System Architecture*, Special Issue on Evolutionary Computing, Elsevier Science, 47: 557-572.

M.Varquez and L.D.Whitley (2000). A Comparison of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem, *In Proceedings of the Genetic and Evolutionary Computing Conference 2000 (GECCO 2000)*, 1011-1018, Las Vegas, Nevada, USA, July 10-12.

A NEW METHODOLOGY FOR EMERGENT SYSTEM IDENTIFICATION USING PARTICLE SWARM OPTIMIZATION (PSO) AND THE GROUP METHOD OF DATA HANDLING (GMDH)

Mark S. Voss
Dept. of Civil and Environmental Engineering
Marquette University
Milwaukee, WI
Voss@Rocketmail.com

Xin Feng
Dept. of Electrical and Computer Engineering
Marquette University
Milwaukee, WI
Xin.Feng@Marquette.edu

Abstract

A new methodology for Emergent System Identification is proposed in this paper. The new method applies the self-organizing Group Method of Data Handling (GMDH) functional networks, Particle Swarm Optimization (PSO), and Genetic Programming (GP) that is effective in identifying complex dynamic systems. The focus of the paper will be on how Particle Swarm Optimization (PSO) is applied within Group Method of Data Handling (GMDH) which is used as the modeling framework.

1 INTRODUCTION

The methodology of System Identification was developed for the extraction of mathematical models from system data. Evolutionary System Identification has been used to designate the use of evolutionary computation for the determination of the approximate mathematical model from experimental data. Evolutionary systems rely on a notion of competition among a population of individuals that compete to reproduce to form future generations. Emergence is used to describe the self-organization (order for free) exhibited by Complex Dynamic Systems. Emergent systems rely on the self-organization properties of the underlying system (information) (Holland, 1998). These emergent systems use iterative stochastic methodologies to discover the underlying connections implied by the system data. From this perspective, evolutionary methodologies are also emergent, but the opposite is not always true. The methodology of Emergent System Identification that is proposed here is concerned with extending the concept of Evolutionary System Identification by combining the methodologies of System Identification (Pandit, 1984) self-organizing functional networks (GMDH) (Ivakhnenko, 1968a, 1971b; Madala and Ivakhnenko, 1994), Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 2001a) and Genetic Programming (GP) (Iba and Kurita, 1994).

The rest of this paper is organized as follows. Section 2 describes traditional System Identification and introduces the use of Particle Swarm Optimization (PSO) for determining the coefficients of a simple autoregressive moving average model (SwARMA). Section 3 explains Particle Swarm Optimization. Section 4 describes the results of using PSO for determining the ARMA model parameter (SwARMA) for an example problem. Section 5 introduces and explains the Group Method of Data Handling (GMDH) and the extension of the GMDH algorithms using PSO. Section 6 describes the results of using the GMDH combined with PSO for two example problems and an additional example problem that illustrates nodal selection criterion. The paper ends with the primary conclusions we draw from the results.

2 SYSTEM IDENTIFICATION

Generally speaking, the discipline of system Identification is concerned with the derivation of mathematical models from experimental data. When given a data set one typically applies a set of candidate models and chooses one of the models based on a set of rules by which the models can be assessed. One of the simplest System Identification models is the Autoregressive Moving Average (ARMA) model as shown in equation 1.

$$\begin{aligned} & y_t + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_n y_{t-n} \\ & = \theta_0 + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_m a_{t-m} \end{aligned} \tag{1}$$

where ϕ_i, $i = 1,n$ and θ_j, $j = 0,m$ are the parameters for an ARMA(n,m) model. For a given ARMA(n,m) model the model parameters ϕ_i and θ_j are selected such that equation (2) is minimized,

$$\sum_{t=1}^{N} \left(y_{t\_data} - y_{t\_ARMA} \right)^2 \tag{2}$$

where,

$$N = \text{ number of data points.} \tag{3}$$

ARMA models are numerically efficient due to their ability to utilize traditional parameter estimation methods and typically employ non-linear least squares for the determination of their parameters. In this paper it is shown that Particle Swarm Optimization (PSO) can be used to determine the parameters for ARMA models. The use of PSO to determine the constants of the ARMA model is denoted by what the authors are calling SwARMA. It will be shown that SwARMA was able to determine a better parameterization for the ARMA model than the IMSL (International Mathematical and Statistical Libraries) routines.

3 PARTICLE SWARM OPTIMIZATION

The Particle Swarm Algorithm is an adaptive algorithm based on a social-psychological metaphor (Kennedy and Eberhart, 2001a). A population of individuals adapt by returning stochastically towards previously successful regions in the search space, and are influenced by the successes of their topological neighbors. Most particle swarms are based on two sociometric principles. Particles fly through the solution space and are influenced by both the best particle in the particle population and the best solution that a current particle has discovered so far. The best particle in the population is typically denoted by (global best), while the best position that has been visited by the current particle is denoted by (local best). The (global best) individual conceptually connects all members of the population to one another. That is, each particle is influenced by the very best performance of any member in the entire population. The (local best) individual is conceptually seen as the ability for particles to remember past personal successes.

Particle Swarm Optimization is a relatively new addition to the evolutionary computation methodology, but the performance of PSO has been shown to be competitive with more mature methodologies (Eberhart and Shi, 1998a; Kennedy and Spears, 1998). Since it is relatively straightforward to extend PSO by attaching mechanisms employed by other evolutionary computation methods that increase their performance; PSO has the potential to become an excellent framework for building custom high-performance stochastic optimizers (Løvbjerg, *et al*., 2001). It is interesting to note that PSO can be considered as a form of continuous valued Cellular Automata. This allows its hybridizations to extend into areas other than computational intelligence (Kennedy and Eberhart, 2001).

3.1 PSO EQUATIONS

The ith particle is represented as,

$$X_I = (x_{i1}, x_{i2}, ..., x_{iD}) \tag{4}$$

where D is the dimensionality of the problem. The rate of the position change (velocity) of the i^{th} particle is represented by,

$$V_I = (v_{i1}, v_{i2}, ..., v_{iD}) \tag{5}$$

where v_{ik} is the velocity for dimension "k" for particle "i". The best previous position (the position giving the best fitness value) of the i^{th} particle is represented as,

$$P_I = (p_{i1}, p_{i2}, ..., p_{iD}) \tag{6}$$

The best previous position so far achieved by any of the particles (the position giving the best fitness value) of the i^{th} particle is recorded and represented as,

$$P_G = (p_{g1}, p_{g2}, ..., p_{gD}) \tag{7}$$

On each iteration the velocity for each dimension of each particle is updated by,

$$v_{ik} = w_k v_{ik} + c_1 \varphi_1 p_{ik} + c_2 \varphi_2 p_{gk} \quad \{k, g\} \in \{1, 2, ..., D\} \tag{8}$$

where w_k is the inertia weight that typically ranges from 0.9 to 1.2. c_1 and c_2 are constant values typically in the range of 2 to 4. These constants are multiplied by φ (a uniform random number between 0 and 1) and a measure of how far the particle is from its personal best and the best particle so far. From a social point of view, the particle moves based on its current direction (w_k), its memory of where it found its personal best (p_{ik}), and a desire to be like the best particle in the population (p_{gk}).

3.2 PSO – POSITION UPDATE RULE

After a new velocity for each particle is calculated, each particle's position is updated according to:

$$x_{ik} = x_{ik} + v_{ik} \tag{9}$$

It typically takes a particle swarm a few hundred to a few thousand updates for convergence depending on the parameter selections within the PSO algorithm (Eberhart and Shi, 1998b).

3.3 RESULTS: PSO+ARMA

Particle Swarm Optimization was used to determine the ARMA parameters for the Wolfer Sunspot Data (1770-1869). The results from the study are shown in Fig. 1. This model was chosen to demonstrate the use of PSO on a well understood System Identification problem. The authors were somewhat surprised at the results. The particle swarm converged after a few thousand iterations in less that a minute on a 200 Mhz Pentium PC. In all cases the solution found was substantially better than those found using the IMSL Libraries.

For both the ARMA(2,1) and ARMA(4,2) models the PSO solution was better than the IMSL solution. In light of these results the authors chose to call the combination of PSO with ARMA: SwARMA (Voss and Feng, 2001). These results suggest some interesting future research with regards to traditional System Identification.

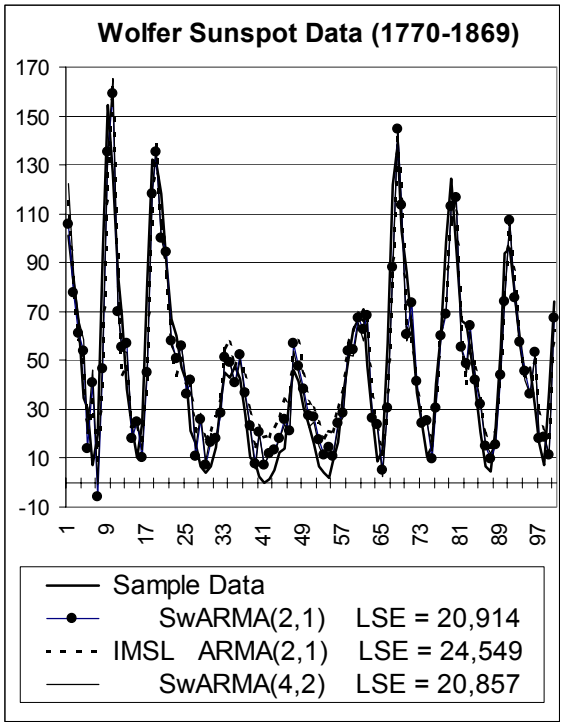


Fig. 1. Wolfer Sunspot Data.
Static SwARMA models. Weight = 0.7.

4 THE GROUP METHOD OF DATA HANDELING

The group method of data handling (GMDH) was first proposed by Alexy G. Ivakhnenko (1968). The traditional GMDH method is based on an underlying assumption that the data can be modeled by using an approximation of the Voltera Series or Kolmorgorov-Gabor polynomial as shown in the following equation,

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_i x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_i x_i x_j x_k \cdots \qquad (10)$$

Ivakhnenko accomplished this by using a feed-forward self-organizing polynomial functional network shown in Fig 2.

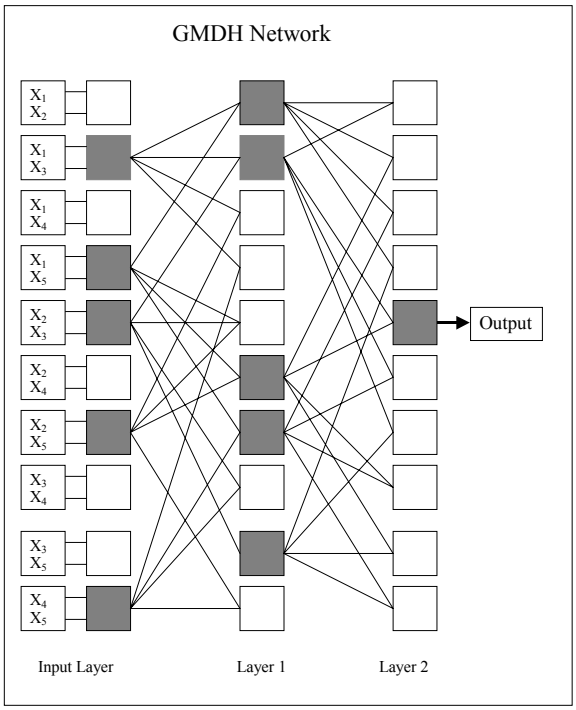


Fig. 2. GHDH forward feed functional network

The inputs to the input layer are determined by taking all combinations (taken two at a time) of the input vector "x_i". Each combination of inputs forms an input node that tries to model the corresponding system output using a second order polynomial surface specified by the polynomial in equation 11.

$$y = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_1 x_2 + c_4 x_1^2 + c_5 x_2^2 \qquad (11)$$

The nodes in the input layer that do the "best job" (shaded nodes) at modeling the system output are retained and form the input to the next layer. The inputs for layer 1 are formed by taking all combinations of the surviving output approximations from the input layer nodes. It is seen that at each layer the order of the approximation is increased by two. The layer 2 nodes that do the "best job" at approximating the system output are retained and form the layer 3 inputs. This process is repeated until the current layer's best approximation is inferior to the previous layer's best approximation. The previous layer's best approximation is then used as the final solution. Determining the method for ranking the nodes at a given layer is somewhat problem dependant. Typically the data is spit into two groups. One data group is used to train the network and the other data group is used to rank the nodes to determine which nodes survive to form the input to the next layer. The GMDH can thus be seen as a methodology for distributed self-organizing computation.

4.1 EXTENDING THE GMDH

The GMDH can be used as an embryo for more complex methodologies for distributed self-organizing computation (Nikolaev,N. and Iba,H., 2001). The methodology is modified here by substituting equation 12 in place of the traditional six term linear polynomial approximation,

$$y = c_0 + c_1 x_1 + c_2 x_2 + sign(x_1)\,|\,x_1\,|^{c_3} + sign(x_2)\,|\,x_2\,|^{c_4} \quad (12)$$

This non-linear equation was designed to test the application of Particle Swarm Optimization for nodal optimization within a GMDH network. The non-linear equation has one less parameter than the traditional polynomial approximation and does not admit the application of simple gradient based optimization methods due to the incorporation of the absolute value function. The GMDH methodology has also been used as a starting point for many new approaches to the System Identification problem (Iba and Kurita, 1994). Here it is demonstrated that it is practical to allow for low-level non-linear emergent nodal representations embedded in a higher-level self-organizing network. This is the type hierarchical model that is necessary for the methodology of Emergent System Identification.

5 RESULTS: PSO + GMDH

Three example problems were considered. The first problem was a test of the applicability of substituting equation 12 in place of equation 11 in a GMDH node. The second problem was a comparison of using equations 11 and 12 for the System Identification of a simple string vibrating in a non-linear fluid, where the damping force was set proportional to the square of the velocity of the string movement. The third problem was the prediction of natural gas flow for a location in the Midwest United States.

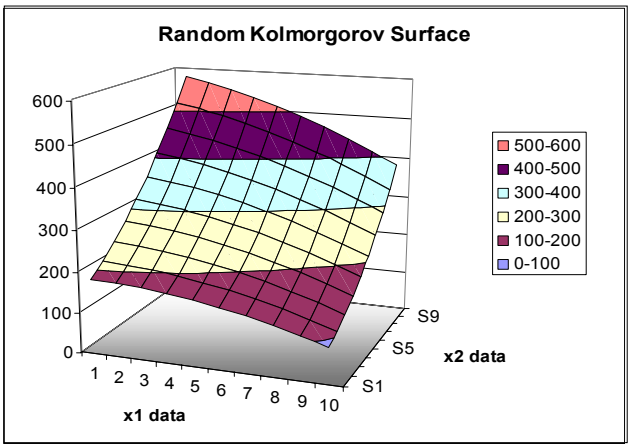


Fig. 3. Random Kolmorgorov Surface

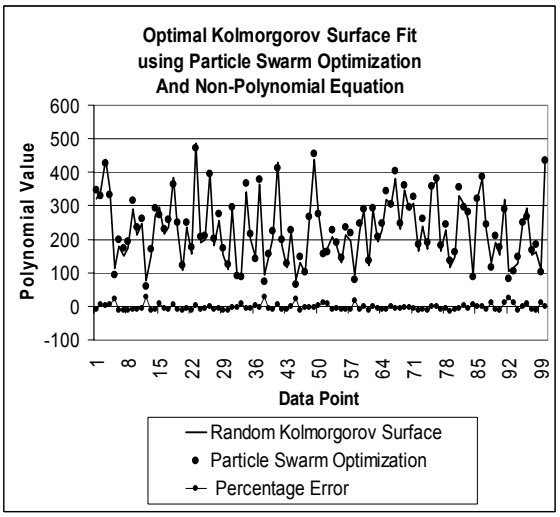


Fig. 4. PSO - non-linear surface fit

5.1 GMDH AND NON-LINEAR NODAL FUNCTIONAL REPRESENTATION

For the purposes of determining the applicability of equation 6 a surface as shown in Fig. 3 was generated using 100 random values for x_1 and x_2 between 0 and 1 using the following form of equation 11.

$$y = 150 - 175x_1 + 500x_2 - 200x_1x_2 + 100{x_1}^2 - 175{x_2}^2 \quad (13)$$

The results for using Particle Swarm Optimization are shown in Fig. 4.

The results were very good considering that the Particle Swarm Algorithm parameters where not optimized for solving this problem. These results lend support for the use of Particle Swarm Optimization in combination with non-linear formulations for the GMDH nodes (such as equation 12) within the GMDH methodology.

5.2 VIBRATING STRING – GMDH AND NON-LINEAR NODAL FUNCTIONAL REPRESENTATION

Since we are investigating the application of Particle Swarm Optimization for its utility in optimizing non-linear models within the GMDH nodes, the specifics of the discrete string model are not given here. The inputs to the GMDH were the previous four amplitudes calculated at the center of a string vibrating in a non-linear fluid. The string was given an initial displacement of 1.0 and then released. The previous four amplitudes were then used to predict the future position of the string at its central location. The System Identification results for equation 11 and 12 are shown in Fig. 5. The particle swarm quickly converged for the two models with the results for the two equations almost equal.

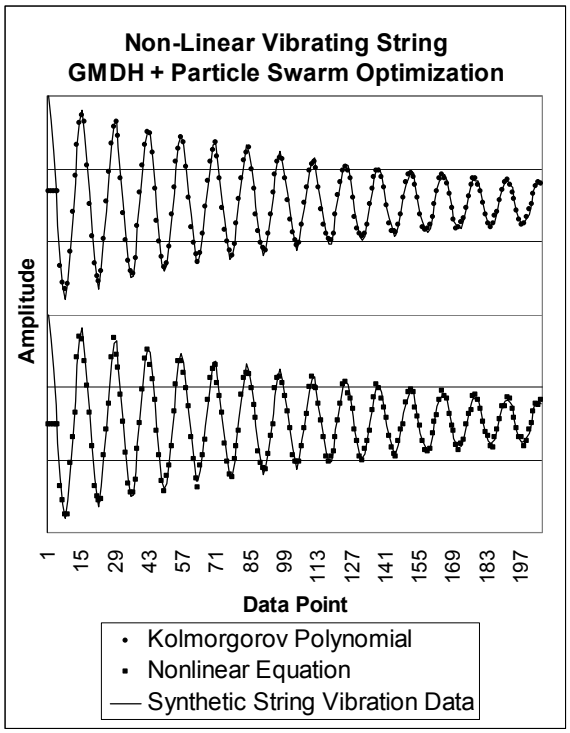


Fig. 5. GMDH - Non-linear vibrating string.

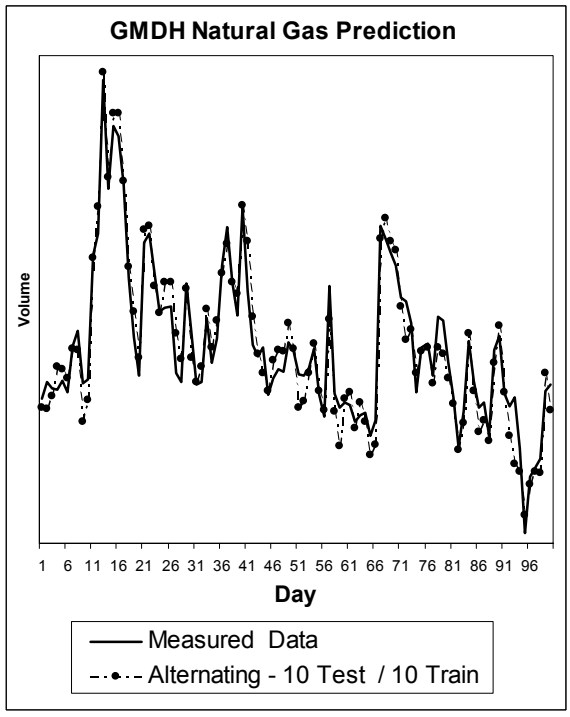


Fig. 6. GMDH - Natural Gas: Alt. Test/Train.

This lends support for GMDH nodal equations of the form given in equation 12 since it requires one less degree of freedom than equation 11. For models where training time is not critical these results support the use of PSO and non-linear functional representations within the GMDH nodes.

5.3 NATURAL GAS PREDICTION - GMDH

In the last example we investigated the applicability of a traditional GMDH for predicting natural gas consumption. In Fig. 6 the GMDH network was trained on all 100 days in the data set. Fig. 7 was trained and tested on alternating 10 day periods.

For both gas consumption studies the temperature, wind and volume for the previous two days were used to predict today's required volume. No solid conclusions can be drawn from this study, but it does illustrate the trade-off that is made with respect to choosing a criterion for selecting the surviving nodes at a given layer in a traditional GMDH network. The GMDH network, shown in Fig. 6., that was trained on 10 days with the next 10 days used for selecting the surviving nodes within a layer does not do as good a job on average but never over or under predicts as much as the GMDH network trained on all the data.

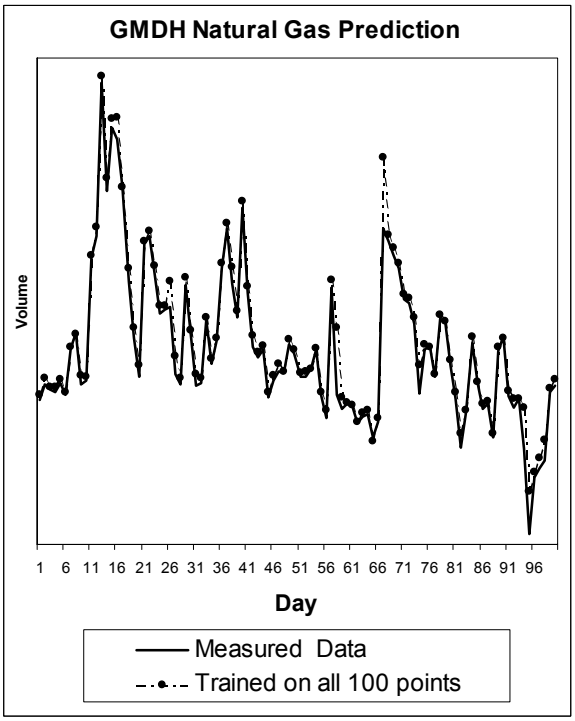


Fig. 7. GMDH - Natural Gas: Trained on all data.

The GMDH network trained on all the data shown in Fig. 7 does a good job on almost all of the days except for day 70 where the prediction is noticeably high.

A more in depth study would have to be undertaken to determine the quality of these results as compared to traditional neural networks, but the results are promising

when one takes into account that these networks were trained in a few seconds.

For both gas consumption studies the temperature, wind and volume for the previous two days were used to predict today's required volume. No solid conclusions can be drawn from this study, but it does illustrate the trade-off that is made with respect to choosing a criterion for selecting the surviving nodes at a given layer in a traditional GMDH network. The GMDH network, shown in Fig. 6., that was trained on 10 days with the next 10 days used for selecting the surviving nodes within a layer does not do as good a job on average but never over or under predicts as much as the GMDH network trained on all the data. The GMDH network trained on all the data shown in Fig. 7 does a good job on almost all of the days except for day 70 where the prediction is noticeably high. A more in depth study would have to undertaken to determine the quality of these results as compared to traditional neural networks, but the results are promising when one takes into account that these networks were trained in a few seconds.

6 CONCLUSION

Preliminary studies indicate that Particle Swarm Optimization can be used to develop superior estimates for the ARMA model parameters for noisy (real world) data. Since the run time for these studies was only a few minutes (at most) it can be inferred that Particle Swarm Optimization is competitive with traditional non-linear least squares algorithms for determining the parameters for many traditional System Identification tasks. Additionally, Particle Swarm Optimization does not need to exploit any mathematical properties that are specific to a particular system model.

The practical use of Particle Swarm Optimization for training non-linear nodes within a GMDH network was demonstrated. This was illustrated using a non-linear equation that has one less parameter than the traditional polynomial approximation while producing competitive training results. Since the non-linear nodal equation that was demonstrated is only one of many that can be used, this implies that families of non-linear functions could be trained for each node. This would allow for GMDH networks that are self-organizing at multiple levels. The examples studied provide experimental support for the practical use of low-level non-linear emergent nodal representations embedded in a higher-level self-organizing network. This hierarchical network (self-organizing at many levels) forms the basis for the methodology that we are calling Emergent System Identification.

References

Iba, H., Kurita, T., (1993), System Identification using Structured Genetic Algorithms, *Proceedings Fifth International Conference on Genetic Algorithms*, 279-286, Morgan Kaufmann.

Eberhart, R.C., Shi, Y., (1998a). Comparison between Genetic Algorithms and Particle Swarm Optimization, *Evolutionary Programming VII*, Porto, V.W., Saravanan, N. Waagen, D., Eiben, A.E., 1447, 611-618, Springer-Verlag.

Eberhart, R.C., Shi, Y., (1998b). Parameter Selection in Particle Swarm Optimization, *Evolutionary Programming VII*, Porto, V.W., Saravanan, N. Waagen, D., Eiben, A.E., 1447, 591-600, Springer-Verlag.

Holland, J. H., (1998). *Emergence, From Chaos to Order*, Adison-Wesley.

Ivakhnenko A.G., (1968a). The Group Method of Data Handling – A rival of the Method of Stochastic Approximation, *Soviet Automatic Control*, Vol. 13, No. 3.

Ivakhnenko A.G., (1971b). Polynomial theory of complex systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 364--378.

Kennedy, J., Eberhart, R.C., (Contributor), Shi, Y., (2001a). *Swarm Intelligence*, Morgan Kaufmann, San Francisco, CA.

Kennedy, J., and Spears, W.M., (1998b). Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator, *Proceedings of the 1998 International Conference on Evolutionary Computation*.

Løvbjerg, M., Rasmussen, T.K., Krink, T., (1998). Hybrid Particle Swarm Optimizer with Breeding and Subpopulations, *Proceedings Genetic and Evolutionary Computation Conference*, GECCO-2001, Morgan Kaufmann, San Francisco, CA.

Madala, H.R., Ivakhnenko, A.G., (1994). *Inductive Learning Algorithm for Complex Systems Modeling*, CRC Press.

Nikolaev,N. and Iba,H., (2001) Accelerated Genetic Programming of Polynomials, *Genetic Programmimg and Evolvable Machines*, vol.2, N: 3.

Pandit, S.M., Wu, S., (1983) *Time Series and System Analysis with Applications*, John Wiley & Sons.

Voss, M.S., Feng, X., (2001). Emergent System Identification using Particle Swarm Optimization, *Proceedings Complex Adaptive Structures Conference*, SPIE, Bellingham, WA.

Voss, M.S., Feng, X., (2002). ARMA Model Selection using Particle Swarm Optimization and AIC Criteria, *Accepted for Procedings 15th International Federation of Automatic Control World Congress*, IFAC, Barcelona, Spain.

Automatic Test Data Generation for Structural Testing of Embedded Software Systems by Evolutionary Testing

Joachim Wegener, Kerstin Buhr, Hartmut Pohlheim
DaimlerChrysler AG, Research and Technology
Alt-Moabit 96a, D-10559 Berlin, Germany, +49 30 39982 232
{Joachim.Wegener, Kerstin.Buhr, Hartmut.Pohlheim}@DaimlerChrysler.com

Abstract

Testing is the most important analytic quality assurance measure for software. The systematic design of test cases is crucial for test quality. Structure-oriented test methods, which define test cases on the basis of the internal program structures, are widely used.
Evolutionary testing is a promising approach for the automation of structural test case design which searches test data that fulfil given structural test criteria by means of evolutionary computation.
In this paper we present our evolutionary test environment, which performs fully automatic test data generation for most structural test methods. We shall report on the results gained from the testing of real-world software modules. For most modules we reached full coverage for the structural test criteria.

1 INTRODUCTION

A great number of today's products is based on the deployment of embedded systems. In industrial applications embedded systems are predominantly used for controlling and monitoring technical processes. There are examples in nearly all industrial areas, for example in aerospace technology, railway and motor vehicle technology, process and automation technology, communication technology, process and power engineering, as well as in defense electronics. Nearly 90% of all electronic components produced today are used in embedded systems.

In order to achieve high quality in the development of embedded systems, central importance is attributed to analytical quality assurance. In practice, the most important analytical quality assurance measure is dynamic testing. Thorough testing of the systems developed is essential for product quality. The aim of the test is to detect errors in the system under test, and, if no errors are found during comprehensive testing, to convey confidence in the correct functioning of the system. This is the only procedure which allows the testing of dynamical system behavior in a real application environment.

The most significant weakness of the test is that the postulated functioning of the tested system can, in principle, only be verified for those input situations selected as test data. Testing can only show the existence and not the non-existence of errors. Therefore, the correctness proof can only be produced by a complete test. In practice, a complete test, with the exception of a few trivial cases, is not executable because of the enormous amount of possible input situations. Thus, the test is a sampling procedure. Accordingly, a task which is essential to testing is the selection of an appropriate sample containing the most error-sensitive test data.

Among the different test activities (test case design, test execution, monitoring, test evaluation, test planning, test organization, and test documentation – see Fig. 1) test case design is of essential importance.

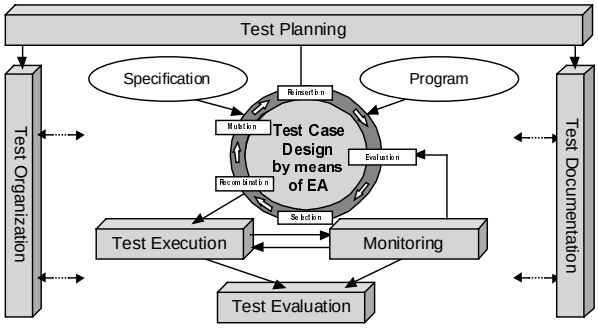


Fig. 1 Structure and interaction of test activities including test case design by means of evolutionary algorithms

For most test objectives an automation of test case design is difficult to achieve. Thus, test case design usually has to be performed manually. Manual test case design, however, is time-intensive and susceptible to errors. The quality of the testing is heavily dependent on the performance of the single tester.

To increase the effectiveness and efficiency of the test, and thus to reduce the overall development costs for software-based systems, we require a test which is systematic and extensively automatable. For this reason, DaimlerChrysler Research works in the area of *evolutionary testing* [19]. The

aim of the work is to increase the quality of the tests and to achieve substantial cost savings in system development by means of a high degree of automation of test case design.

Evolutionary testing is a promising approach for fully automating test case design for various test aims. For instance, evolutionary tests can be used to systemize and automate the testing of non-functional properties and to generate test cases for conventional test methods. In this paper we shall concentrate on structural testing.

The only prerequisites for the application of evolutionary testing are an executable test object and its interface specification. For the automation of structural testing the source code of the test object must be available to enable its instrumentation.

In Section 2 we give a short overview of evolutionary testing. Section 3 describes the different aspects of structural testing. Our evolutionary test environment is presented in Section 4. A large number of real-world software modules have already been tested. The results are presented in Section 5. Our concluding remarks are set out in Section 6 along with our outlook for future research.

2 EVOLUTIONARY TESTING

Evolutionary testing is characterized by the use of metaheuristic search techniques for test case generation (see Fig. 1). The test aim considered is transformed into an optimization problem. The input domain of the test object forms the search space in which one searches for test data that fulfils the respective test aim. Additionally, a numeric representation of the test aim is necessary. This numeric representation is used to define objective functions suitable for the evaluation of the generated test data. Depending on which test aim is pursued, different objective functions emerge for test data evaluation. Section 4 describes objective functions for structural testing in detail.

Due to the non-linearity of software (if-statements, loops etc.) the conversion of test problems to optimization tasks mostly results in complex, discontinuous, and non-linear search spaces. Neighborhood search methods like hill climbing are not suitable in such cases. Therefore, metaheuristic search methods are employed, e.g. evolutionary algorithms, simulated annealing, or taboo search. In our work, evolutionary algorithms are used to generate test data because their robustness and suitability for the solution of different test tasks has already been proven in previous work, e.g. [14], [6], [19].

As we assume the reader to be familiar with evolutionary algorithms we shall only describe the interaction of the evolutionary algorithm with the other testing activities in this paper. Please refer to [18] and [17] for a longer description of evolutionary algorithms in the context of evolutionary testing.

Figure 2 presents the structure of evolutionary testing from the point of view of the evolutionary algorithm. The interaction with the other testing activities occurs during the evaluation of the individuals.

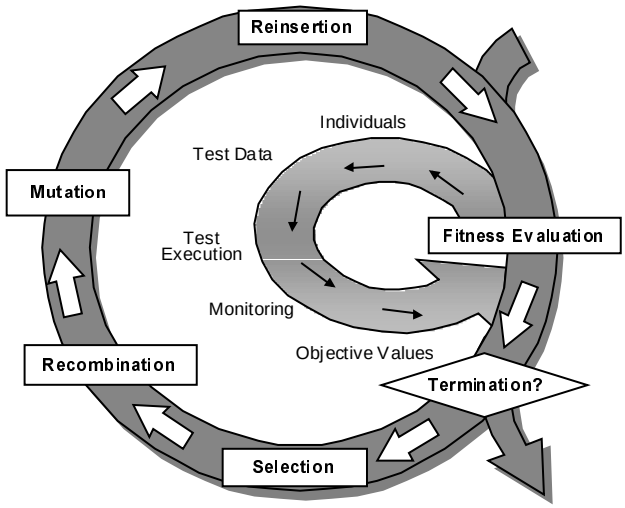


Fig. 2 Structure of Evolutionary Testing

Each individual within the population represents a test datum with which the test object is executed. For each test datum the execution is monitored and the objective value is determined for the corresponding individual. Next, population members are selected with regard to their fitness and subjected to recombination and mutation to generate new offspring. It is important to ensure that the test data generated are in the input domain of the test object. Offspring individuals are then evaluated by executing the test object with the corresponding test data. A new population is formed by combining offspring and parent individuals, according to the survival procedures defined. From now on, this process repeats itself until the test objective is fulfilled or another given termination criterion is reached.

3 STRUCTURAL TESTING

Structural testing is widespread in industrial practice and stipulated in many software-development standards, e.g. [13], [5], and [3]. The execution of all statements (statement coverage), all branches (branch coverage), or all conditions with the logical values True and False (condition coverage) are common test aims. Structural test methods are usually applied to unit tests. There are no enforced structure test criteria for integration tests or system tests.

The aim of applying evolutionary testing to structural testing is the automatic generation of a quantity of test data, which leads to the best possible coverage of the respective structural test criterion. In the case of statement testing, the goal of the test is to execute each program statement at least once. In the case of branch testing the empty branches also have to be executed. The test goals are based on the assumption that a

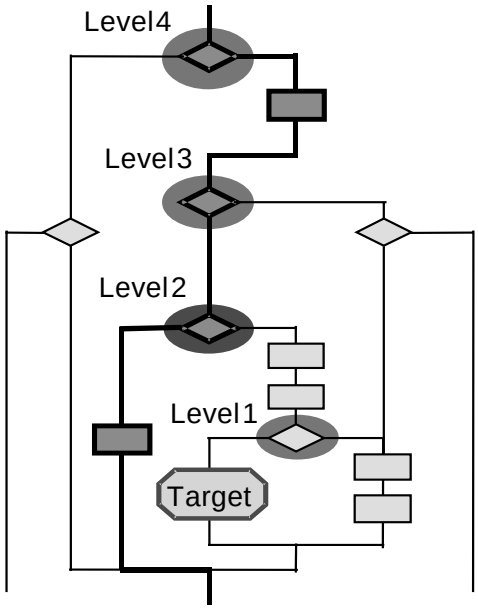


Figure 3. Approximation level calculation (objective function for structural testing)

test, which does not include each statement and all branches (of the system under test being executed) at least once, does not present a thorough check of the test object. Therefore, the overall goal of the test case design is to define a set of test data which guarantees that each statement or each branch is executed.

Whereas all previous work has concentrated on selected structural test criteria (statement-, branch-, condition and path test), our test environment has been developed to support all common control-flow and data-flow oriented test methods.

In order to search for the test data set the test is divided into partial aims. Each partial aim represents a program structure that requires execution to achieve full coverage, for example a certain statement or branch. For each partial aim, an individual objective function is formulated and a separate optimization is undertaken to search for a test datum executing the partial aim.

In order to direct the search toward program structures not covered, the objective function computes a distance for each individual that indicates how far away it is from executing the desired program structure. Individuals which are closer to the execution of the desired program structure will be selected as parents and combined to produce offspring individuals.

The objective functions of the partial aims consist of two components – the approximation level and the local distance calculation.

3.1 APPROXIMATION LEVEL CALCULATION

The approximation level supplies a figure for an individual that gives the number of branching nodes lying between program structures covered by the individual and the desired program structure. For this computation, only those branching nodes are taken into account that contain an outgoing edge that results in a miss of the desired program structure.

An example is given in Figure 3. The program graph contains four branching nodes which could result in a miss of the target node (representing the desired statement or branch). Each node is assigned the corresponding approximation level (level 4 to level 1). An individual that branches away from the target node at the first branching node attains a lower approximation level (level 4) than an individual which reaches level 3 etc. The figure shows the execution of an individual which misses the target node in the branching node with the approximation level 2. The individual passes the branching nodes in the levels 4 and 3 as desired but misses the target node at level 2.

3.2 LOCAL DISTANCE CALCULATION

The calculation of the local distance is performed in order to distinguish different individuals executing the same program path. For this, a distance to the execution of the other program path is calculated for the individual by means of the branching conditions in the branching node in which the target node is missed. Figure 4 illustrates this calculation.

For example, if a branching condition x==y needs to be evaluated as True to reach the target node, then the objective function may be defined as |x-y|. If an individual obtains the local distance 0, a test datum is found which fulfils the branching condition: x and y have the same value.

If a branching node contains multiple conditions the local distance is a combination of the local distances of each condition. For a node of the type a $\vee$ b the local distance of an

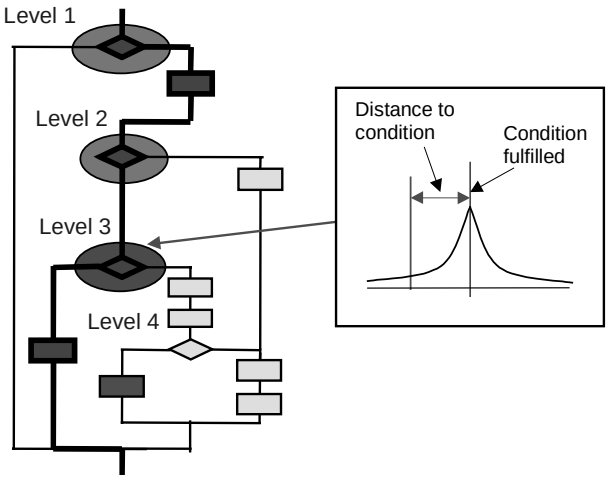


Figure 4. Local distance calculation (objective function for structural testing)

individual is obtained from the minimum value for each single predicate a and b. In the case of a $\wedge$ b the local distance of an individual is the result of the sum of the distances determined for each single predicate.

3.3 OBJECTIVE FUNCTION

The overall objective function value of an individual with respect to a certain partial aim is defined as the sum of its approximation level and its normalized local distance:

F(pa, i) = AL(pa, i) + (1 – LD(N(pa, AL(pa, i)), i),

- F(pa, i): objective value of individual i for the partial aim pa,
- AL(pa, i): highest approximation level of the individual i for the partial aim pa,
- N(pa, al): branching node with the highest approximation level for the partial aim pa,
- LD(n, i): normalized local distance of the individual i in branching node n.

An individual with an objective value of 0 leads to the passing of the desired program structure. This provides a natural termination criteria for the optimization of this partial aim.

Even though the evolutionary test works up only one partial aim after the other, the execution of a test datum usually leads to passing several partial aims. Thus, the test soon focuses on those program structures which are difficult to reach. After having worked up all partial aims, a minimal amount of test data is returned to the tester. This test data set leads to an execution of all reached partial aims.

4 EVOLUTIONARY TEST ENVIRONMENT

In order to automate test case design for different structural testing methods with evolutionary tests we have developed a tool environment which consists of six components:

- parser for the analysis of test objects,
- graphical user interface for the specification of the input domain of the test objects,
- instrumenter which captures program structures executed by the generated test data,
- test driver generator which generates a test bed running the test object with the generated test data,
- test control which includes the identification and administration of the partial aims for the test and which guarantees an efficient test by defining a processing order and storage of initial values for the partial aims,
- toolbox of evolutionary algorithms to generate the test data.

Fig. 5 presents the structure of the evolutionary test environment and shows the information exchange between these tools. The parser, interface specification, instrumenter and test driver generator constitute the test preparation. During test execution the test control and the evolutionary algorithm toolbox are employed.

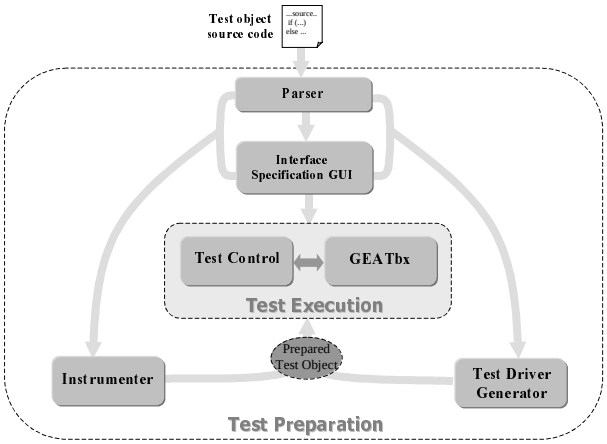


Figure 5. Components of the evolutionary test environment

4.1 PARSER

The parser identifies the functions in the source files which form the possible test objects. It determines all necessary structural information on the test objects. Control-flow and data-flow analyses are carried out for every test object. These analyses determine the interface, the control-flow graph, the contained branching conditions with their atomic predicates, as well as semantic information on the used data structures, e.g. the organization of user-defined data types.

4.2 GRAPHICAL USER INTERFACE FOR INTERFACE SPECIFICATION

To ensure efficient test data generation and to avoid the generation of inadmissible test data from the beginning, the tester may have to define the test object interface determined by the parser more precisely. For this, the developed tool environment provides a graphical user interface that displays the test objects and their interfaces as they have been determined by the parser.

The tester can limit the value ranges for the input parameters and enter logical dependencies between different input parameters. These will then be considered during test data generation. It is also possible to enter initial values for single or for all input parameters. As a result, test data of a previous test run or data of an already existing functional test, as well as specific value combinations for single input parameters, can be used as a starting point for test data generation (seeding).

4.3 INSTRUMENTER

The third component in the tool environment is the instrumenter that enables test run monitoring. In order to eliminate influences on program behavior the instrumentation has to take place in the branching conditions of the program. The

instrumentation of the branching conditions is always the same, independent of the selected structural test criterion.

The atomic predicates in the branching nodes of the test object are instrumented to measure the distances individuals are away from fulfilling the branching conditions (see Section 3.2). The instrumentation also provides information on the statements and program branches executed by an individual.

4.4 TEST DRIVER GENERATOR

The test driver generator generates a test bed that calls the test object with the generated individuals and returns the monitoring results provided by the execution of the instrumented test object to the test control. When the test object is called by the test driver, the individuals are mapped onto the interface of the test object. It is important that user specifications for the test object interface are taken into account. Individuals that do not represent a valid input are extracted and assigned a low fitness value.

4.5 TEST CONTROL

The most complex component of the evolutionary test environment is the test control. It is responsible for several difficult tasks: the management of the partial aims with their processing status, the collection of suitable initial values for the optimization of partial aims, and the recording of test data fulfilling partial aims.

The test control identifies partial aims for the selected structural test criterion. Partial aims are determined on the basis of the control–flow graph provided by the parser. The test control manages the determined partial aims and regulates the test progress. One after the other, the different partial aims are selected in order to search for test data with the evolutionary test. Independent optimizations are performed for every partial aim.

Although only one partial aim is considered for the optimization at a time, all individuals generated are evaluated with regard to all unachieved partial aims. Thus partial aims reached by chance are identified, and individuals with good objective values for one or more partial aims are noted and stored. Subsequent testing of these partial aims then uses the stored individuals as initial values (also compare [9]). This method is called seeding. It enhances the efficiency of the test because the optimization does not start with an entirely randomly generated set of individuals.

In order to calculate the objective values for the individuals the test control determines the program paths executed by every individual, on the basis of the data attained by test monitoring and the test object's control-flow graph. Objective values are then evaluated by applying the objective functions described in Section 3, that take into account the local distance measurement for the branching conditions as well as the approximation level.

The processing sequence for the partial aims that have not yet been attained is guided by the test control depending on the availability of suitable initial values. The partial aim for which the individuals with the best objective values are available is selected as the next one for the test. This ensures that the test quickly achieves a high coverage because partial aims which are difficult to execute or infeasible do not slow down the overall testing process. When no initial values are available, or several equally good initial values for different partial aims exist, then a breadth-first search is carried out. If the search fails to find a test datum for a partial aim it is marked as already processed and not fulfilled. During the remaining optimization process it is possible to reset this status if an individual with a better objective value for this partial aim is found accidentally. The partial aim is then targeted again for an additional test employing the attained values for initialization.

Once all partial aims have been processed the test is finished. The test data for the separate partial aims are then compiled and displayed with the obtained coverage. On this basis, the tester is able to check whether program structures that were not covered are infeasible, or whether the evolutionary test was not able to generate suitable test data.

In addition, the test control offers a simple application programming interface (API) to export test data found, and actual values for the output parameters of the test object, in order to support automatic test evaluation on the basis of test oracles. Moreover, it provides test and monitoring information for the visualization of the test progress.

4.6 GENETIC AND EVOLUTIONARY ALGORITHMS TOOLBOX GEATBX

We have applied the *Genetic and Evolutionary Algorithm Toolbox for Use with Matlab* (*GEATbx*) [10]. This is a very powerful tool that supports real and integer number representation of individuals as well as binary coding. Almost any hybrid form of evolutionary algorithm can be implemented, including genetic algorithms and evolution strategies. The toolbox offers a large number of different operators for the components of evolutionary algorithms, and also enables the application of sub-populations, migration and competition between sub-populations, and possesses extensive visualization functions for displaying the optimization state and progress. It is possible to specify admissible value domains for the parameters of an individual. The toolbox automatically ensures that these value domains are observed during the generation of individuals. Thus, the test driver only needs to check for dependencies between the single variables of an individual.

5 APPLICATION OF EVOLUTIONARY TESTING

Our tool environment has already been applied in experiments with typical real-world examples. Currently, our work concentrates on the automatic generation of test data for statement and branch tests, which has yielded excellent results. For all test objects a complete or very high coverage was achieved by the evolutionary test.

5.1 TEST OBJECTS

Table 1 presents a selection of examined test objects. To assess and compare the complexity of the software modules we report a number of software metrics. These figures are taken from a larger study examining the complexity of more than 40 different software modules [2].

The basic metric is lines of code. The cyclomatic complexity gives information on the test object's control flow. The nesting complexity assesses the nesting level and can indicate the difficulty in reaching a partial aim with respect to the control-flow. Myer's interval shows the complexity of certain branching conditions.

Asin() calculates the arcsin or arccos for the passed argument (1 double) and is a typical C library function.
Prototype: double asin(double arg);

Atof() is another typical C library function. It converts strings to the corresponding floating point value. *Atof()* contains several evaluations which check the input string for its validity. In the experiments the maximum string length was set to 10 characters in ASCII coding. Accordingly, the size of the search space is 255^{10}. For this test object each test datum (individual of the evolutionary algorithm) consists of 10 integer variables, each with a possible value of 1 to 255.
Prototype: double atof(char InStr[10]);

The *classifTria()* function is an implementation of the classic triangle classifier example used in a large number of testing papers. It is used in two different data type versions. The input domain is given either by three floating point values or by three integer values.
Prototype: void classifTria(double a, double b, double c);

In *powi()* a passed floating point value is raised to a passed integer value. The input consists of 1 double and 1 integer.
Prototype: double powi(double x, int nn);

Incbet() is a larger test function. It calculates the incomplete beta-integral of the passed argument (3 doubles).
Prototype: double incbet(double aa, double bb, double xx);

In the experiments the double values were bounded in $[-10^6, 10^6]$.

5.2 EVOLUTIONARY TESTING SETUP

Evolutionary testing was carried out using an evolutionary algorithm with the following configuration:

- linear ranking with a selective pressure of 1.7,
- selection by stochastic universal sampling,
- generation gap of 0.9,
- discrete recombination with a recombination rate of 1,
- real or integer valued mutation employing different mutation range for each subpopulation in the range $[0.1, 0.01, ..., 10^{-6}]$ and a mutation rate of (1/number of variables),
- regional population model dividing the population into subpopulations,
- migration between subpopulations every 20 generations in a complete net structure (5% migration rate),
- competition between subpopulations every 5 generations (division pressure of 3),
- maximal number of generation of 200.

The sizing of the subpopulations depends on the complexity of the software module under test. We employed 9 subpopulations with 100 individuals each. This number is relatively large and therefore more appropriate for the complex software modules (*powi()* and *incbet()*). In order to compare the results more easily we used this number for all the modules in all the experiments.

Structural testing exhibits a natural termination criteria. As soon as a partial aim is reached the corresponding optimization process can terminate. Thus, an upper bound for the number of generations is only defined if a partial aim can not be reached.

There is one straightforward mechanism for the dynamic adaptation of evolutionary testing in the context of our work which has already been successfully employed: the use of

Table 1 Metrics and number of partial test aims of the used test objects

metrics/modules	asin	atof	classifTria	powi	incbet
lines of code	13	36	41	51	159
cyclomatic complexity	4	16	14	15	23
Myer's interval	0	27	7	2	3
nesting complexity	4	32	17	19	43
no. of statement cover aims	10	40	30	36	58
no. of branch cover aims	12	56	42	49	79

multiple strategies and competing subpopulations. The use of multiple strategies (different mutation range for each subpopulation) leads to different search strategies: from a globally oriented search when employing a large mutation range to a very fine search when employing a small mutation range. Additionally, depending on the test process, the most successful strategies will be assigned more resources. This leads to an efficient distribution of resources during the whole optimization and a more robust search. For a longer discussion of competing subpopulations see [11].

5.3 COMPARISON OF EVOLUTIONARY AND RANDOM TESTING

We compare the results of evolutionary testing (ET) to those of random testing (RT) for all test objects. The availability of other means of comparison is very limited. One could also compare the results with those of an expert with good knowledge of the modules under test. However, this would involve a great deal of effort which would not be justifiable for real-world problems. We have carried out this kind of comparison for several modules in order to test temporal behavior [12]. In these cases RT performed nearly as well as the expert, whereas ET always proved itself to be better or at least as good as the expert.

The results of the experiments are presented in Table 2. The number of individuals generated for random testing was equivalent to that for evolutionary testing. Each experiment was repeated 10 times. The mean values for the respective results are presented.

The evolutionary test achieved full statement and branch coverage for the first 3 software modules in a very short time. Random testing was unable to reach full coverage for any of the software modules. The coverage values are much lower in general.

For the more complex software modules *powi()* and *incbet()* evolutionary testing reached high coverage values. However, full coverage (100%) was not reached. We are still investigating if these coverage values are the highest possible values (because of infeasible statements and branches). It is thus quite possible that for these test objects, the maximal possible coverage has been reached.

When we compare the results of evolutionary testing and random testing for these two modules the advantage of evolutionary testing is much more apparent. Especially for *incbet()*, the coverage of random testing is substantially lower than that of evolutionary testing.

This suggests, that evolutionary testing is currently the only sensible procedure of structural testing of large and complex software modules.

6 CONCLUDING REMARKS

The thorough test of embedded systems could include a number of demanding testing tasks. The test case design for various test objectives is difficult to master on the basis of conventional function-oriented and structure-oriented testing methods. Moreover, automation of test case design is problematic. Usually, test cases have to be defined manually.

The aim of the work presented in this paper is the automatic generation of test data for structural tests. For this, a tool environment has been developed that applies evolutionary testing to C programs. Test data are generated by means of evolutionary algorithms.

With evolutionary testing a new test method for testing embedded systems is provided, which enables the complete automation of test case design for various test objectives. The idea of evolutionary testing is to search for relevant test cases in the input domain of the system under test with the help of evolutionary algorithms. As described in this paper, evolutionary testing enables the complete automation for structural test case design.

Due to the full automation of the evolutionary testing, the system could be tested with a large number of different input situations. In most cases, more than several thousand test data sets are generated and executed within only a few minutes. The prerequisites for the application of evolutionary

Table 2 Results of statement and branch coverage for evolutionary testing (ET) and random testing (RT)

	asin	**atof**	**classifTria**	**powi**	**incbet**
statement cover	ET / RT	ET / RT	ET / RT	ET / RT	ET / RT
coverage [%]	100 / 50	100 / 61.5	100 / 13.3	90.7 / 77.8	87.9 / 8.6
no. of generations	18	82	172	855	1944
no. of individuals	15 048	66 481	139 476	689 510	813 204
testing time [s]	62	570	1225	7489	12339
branch cover	ET / RT	ET / RT	ET / RT	ET / RT	ET / RT
coverage [%]	100 / 50	100 / 61.8	100 / 11.9	83.7 / 73.5	72.2 / 7.6
no. of generations	16	74	206	1610	3224
no. of individuals	12 944	60 046	166 298	1 298 394	2 600 249

tests are few. Only an interface specification of the system under test is needed to guarantee the generation of valid input values. For structural testing the source code of the test object is also required. The evolutionary test is universally applicable because it adapts itself to the system under test.

In order to guarantee an efficient overall test, the test control of the evolutionary test environment evaluates every individual with regard to every partial aim that has not been reached. Partial aims reached purely by chance are thus identified immediately. Individuals suitable for one or more partial aims are noted, stored, and used as seeds at the optimization of these partial aims. The processing sequence of the partial aims is guided by the quality of the available initial values. In this way, the test quickly achieves the highest possible coverage. Experiments utilizing this strategy have proved successful, and the overall testing procedure has been accelerated considerably.

Evolutionary testing has already produced very good results in the application field. Therefore, evolutionary testing seems to have the potential to increase the effectiveness and efficiency of existing test processes. Evolutionary tests thus contribute to quality improvement and to reduction of development costs.

The users applying evolutionary testing in industrial practice can not be assumed to have any knowledge of evolutionary algorithms. Thus, the selection of evolutionary algorithms employed for the testing of a system needs to take place without the participation of the users. By means of extended evolutionary algorithms (Section 5.2), which combine global and local search procedures in several subpopulations, robust optimization results are obtained for a large number of different testing tasks.

Since research in the field of evolutionary computation is carried out intensively world-wide, further improvements of the search techniques can be expected in the future. Evolutionary testing could directly benefit from such improvements by incorporating new search techniques into test data generation, thus leading to a further increase in the effectiveness and efficiency of the tests.

At present, statement tests, branch tests, condition tests, and segment tests can be applied. Work on multiple-condition testing is drawing to a close. The test environment will also be extended for structural testing of object-oriented Java programs. Furthermore, a visualization component for observing the testing progress will be included and the distribution of tests to several computers will be supported.

References

[1] *Baresel, A.*: Automatisierung von Strukturtests mit evolutionären Algorithmen (Automation of Structural Testing using Evolutionary Algorithms). Diploma Thesis, Humboldt University, Berlin, Germany, 2000.

[2] *Buhr, K.*: Einsatz von Komplexitätsmaßen zur Beurteilung Evolutionärer Testbarkeit (Complexity Measures for the Assessment of Evolutionary Testability). Diploma Thesis, Technical University Clausthal, 2001.

[3] Capability Maturity Model for Software, Software Engineering Institute, Carnegie Mellon University.

[4] *Korel, B.*: Automated Test Data Generation. IEEE Transactions on Software Engineering, vol. 16 no. 8 pp.870-879, 1990.

[5] IEC 65A Software for Computers in the Application of Industrial Safety-Related Systems (Sec 122).

[6] *Jones, B. F., Eyres, D. E. and Sthamer, H.-H.*: A Strategy for using Genetic Algorithms to Automate Branch and Fault-based Testing. The Computer Journal, vol. 41, no. 2, pp. 98 – 107, 1998.

[7] *Grochtmann, M., and Wegener, J.*: Test Case Design Using Classification Trees and the Classification-Tree Editor CTE. Proceedings of Quality Week '95, San Francisco, USA, 1995.

[8] *Mathworks, The*: Matlab - UserGuide. Natick, Mass.: The Mathworks, Inc., 1994-1999. http://www.mathworks.com/

[9] *McGraw, G., Michael, C., Schatz, M.*: Generating Software Test Data by Evolution. Technical Report RSTR-018-97-01, RST Corporation, Sterling, Virginia, USA, 1998.

[10] *Pohlheim, H.*: GEATbx - Genetic and Evolutionary Algorithm Toolbox for Matlab. http://www.geatbx.com/, 1994-2002.

[11] *Pohlheim, H.*: Evolutionäre Algorithmen - Verfahren, Operatoren, Hinweise aus der Praxis. Berlin, Heidelberg: Springer-Verlag, 1999. http://www.pohlheim.com/eavoh/

[12] *Pohlheim, H. and Wegener, J.*: Testing the Temporal Behavior of Real-Time Software Modules using Extended Evolutionary Algorithms. in Banzhaf, W. (ed.): GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA: Morgan Kaufmann, p. 1795, 1999.

[13] RTCA/DO-178B Software Considerations in Airborne Systems and Equipment Certification.

[14] *Sthamer, H.-H.*: The Automatic Generation of Software Test Data Using Genetic Algorithms. PhD Thesis, University of Glamorgan, Pontyprid, Wales, Great Britain, 1996.

[15] *Tracey, N., Clark, J., Mander, K. and McDermid, J. (1998):* An Automated Framework for Structural Test-Data Generation. Proceedings of the 13th IEEE Conference on Automated Software Engineering, Hawaii, USA.

[16] *Watkins, A.*: A Tool for the Automatic Generation of Test Data Using Genetic Algorithms. Proceedings of the Software Quality Conference '95, Dundee, Great Britain, pp. 300-309, 1995.

[17] *Wegener, J.*: Evolutionärer Test des Zeitverhaltens von Realzeit-Sytemen (Evolutionary Testing of the Temporal Behaviour of Real-Time Systems). Shaker Verlag, 2001.

[18] *Wegener, J.; Baresel, A.; Sthamer, H.*: Evolutionary Test Environment for Automatic Structural Testing. Information and Software Technology, Special Issue devoted to the Application of Metaheuristic Algorithms to Problems in Software Engineering, vol. 43, pp. 841 – 854, 2001.

[19] *Wegener, J., and Grochtmann, M.*: Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing. Real-Time Systems, 15, pp. 275-298, 1998.

An Adaptive Genetic Algorithm for Multi Objective Flexible Manufacturing Systems

Abdulnasser Younes and Hamada Ghenniwa*and Shawki Areibi†
School of Engineering
University of Guelph
Guelph, Ontario, Canada N1G 2W1

Abstract

A large number of combinatorial problems are associated with manufacturing optimization[1]. The use of intelligent techniques in the manufacturing field has therefore been growing in the last decade. This paper presents a Genetic Algorithm solution for the manufacturing systems in general and flexible manufacturing in particular. In our implementation we have combined a Pareto-based approach with an adaptive weighted sum technique for tackling the multi-objective flexible manufacturing systems problem. Experimental results demonstrate that this approach is very effective for handling such complex systems.

1 INTRODUCTION

In recent years, distributed and open architectures are considered appropriate design approaches for systems in many environments, particularly, for flexible manufacturing systems (FMS). These systems consist of multiple-heterogeneous machines (machines robots and/or computers), where each machine is capable of performing a specific set of operations that may overlap with those of the other machines. The ultimate goal for designing these systems might be to maximize the FMS throughput[2]. However, several problems such as part type partitioning, assignment and sequencing must be solved before this goal can be achieved. The main focus of this paper is on modeling and solving the assignment problem of FMS. In FMS literature, the assignment problem, is sometimes dealt with as a flow management problem[3], other attempts are based on reducing the problem to a mathematical programming problem. Most combinatorial optimization problems in manufacturing optimization systems are NP-hard, i.e, there is no polynomial time algorithm that can possibly solve them. Heuristic methods are normally employed for the solution of these problems. A growing number of researchers have adopted the use of meta-heuristic techniques such as Simulated Annealing and Tabu Search for difficult combinatorial problems. Evolutionary computation methods are meta-heuristics that are able to search regions of the solution's space without being trapped in local optima. Multi-objective evolutionary algorithms have been recognized to be particularly suitable for solving flexible manufacturing systems because of their ability to exploit and explore multiple solutions in parallel, and the ability to find an entire set of Pareto-optimal solutions in a single run. This paper focuses on developing a model for a class of FMS at the part machine level and solving the problem using a Genetic Algorithm technique. The rest of the paper is organized as follows. Section 2 introduces the main concepts of flexible manufacturing systems. Section 3 presents a Genetic Algorithm solution for the FMS system. Results are introduced in Section 4. The paper concludes with some comments on how the Genetic Algorithm performed and possible future work.

2 BACKGROUND

Many models for the problem of assigning parts of different types to machines have been developed in the literature [4, 5], [6]. These models assume either parallel machines that are identical in capabilities but may differ in speed [2] or machines that are specialized [7, 6]. The goodness of an assignment is measured in terms of minimizing part transfer and balancing the work-load of the machines. These two objectives are lexicographically ordered, such that the

*The second author is in the Electrical Engineering Department at the University of Western Ontario.

†The work of the third author has been partially supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) operating grant (OGP 43417)

primary objective is minimizing part transfer and the secondary objective is balancing the work-load. The aim is to facilitate the creation of machine cells with minimum part transfer while maximizing the utilization of machines. These objectives are conflicting with each other. While minimizing part transfer tends to favor the assignment of the whole of a part to a single machine, balancing work-load tries to make the work-load distribution even among the machines. Thus satisfying both objectives seems a hard problem to solve [8].

2.1 FMS EXAMPLE

Consider a flexible manufacturing system consisting of three machines, M_1, M_2 and M_3, each of which is characterized by a set of operations, respectively, denoted by $\{O_1, O_2, O_3, O_4\}$, $\{O_3, O_4, O_5\}$, and $\{O_4, O_5\}$, where O_i denotes operation i. This system is needed to process two Part types P_1 and P_2. Part P_1 requires four operations denoted by $\{O_1, O_2, O_3, O_5\}$, while P_2 requires three operations denoted by $\{O_2, O_3, O_5\}$. There are several processing choices for this setting, such as:

- First Choice, for part P_1: $(O_1 \mapsto M_1, O_2 \mapsto M_1, O_3 \mapsto M_1, O_5 \mapsto M_2)$ i.e, assign machine M_1 to process O_1, O_2 and O_3, and assign M_2 to process O_5. For part P_2: $(O_2 \mapsto M_1, O_3 \mapsto M_1, O_5 \mapsto M_2)$ i.e assign machine M_1 to process O_2,O_3 and have M_2 process O_5.
- Second Choice, for part P_1: $(O_1 \mapsto M_1, O_2 \mapsto M_1, O_3 \mapsto M_2, O_5 \mapsto M_3)$ i.e, assign machine M_1 to process O_1, O_2, assign M_2 to process O_3, assign M_3 to process O_5. For part P_2: $(O_2 \mapsto M_1, O_3 \mapsto M_2, O_5 \mapsto M_3)$, i.e, have machine M_1 process O_2, M_2 process O_3 and have M_3 process O_5.

Looking back at the suggested choices we can notice that the first solution is biased towards part transfer objective function (i.e minimize transfer of parts between machines) where the total number of machines involved for P_1 were two machines and the same applies for $P2$. The number of operations performed by the machines are five, two and zero respectively i.e there is no balance in the operations performed by different machines. On the other hand the second choice indicates that the balance criteria is met (M_1 performs 3 operations, M_2 performs 2 operations and M_3 performs 2 operations), but the total number of part transfer has increased to four machines. What needs to be accomplished is minimizing part transfer as a primary objective and balancing the workload as a secondary objective instead of optimizing a single objective at a time.

2.2 MATHEMATICAL FORMULATION

The assignment problem can be formulated in terms of minimizing the part transfer and balancing the machine workload. In order to formulate the problem, the following notations are introduced:
i,l : machine index (i,l = 1,2,3,...,n_m)
j : part index (j = 1,2,3,...,n_p)
$\hat{k}_j$: is processing choice for part j (j = 1,2,3,....,n_p)
k_j : is the number of processing choices of P_j
$n_{ji\hat{k}_j}$: is the number of necessary operations required by Pj on M_i in processing choice $\hat{k}_j$, $1 \leq \hat{k}_j \leq k_j$
$t_{ji\hat{k}_j}$: is the work-load of machine M_i to process part P_j in processing choice $\hat{k}_j$.

$$x_{ji\hat{k}_j} = \begin{cases} 1 & \text{if } P_j \text{ requires } M_i \text{ in processing choice } \hat{k}_j \\ 0 & \text{otherwise} \end{cases}$$

$$q_{j\hat{k}_j} = \begin{cases} 1 & \text{if processing choice } \hat{k}_j \text{ is selected for part j} \\ 0 & \text{otherwise} \end{cases}$$

Using this notation, then the objective functions are:

1. Minimization of part transfer (by minimizing the number of machines required to process the part):

$$F_1 = min_{\hat{k}_j} \sum_{i=1}^{n_m} q_{j\hat{k}_j} x_{ji\hat{k}_j}, \forall_j \qquad (1)$$

2. Minimization of the number of necessary operations required from each machine over the possible processing choices:

$$F_2 = min_{\hat{k}_j} \sum_{i=1}^{n_m} q_{j\hat{k}_j} x_{ji\hat{k}_j} n_{ji\hat{k}_j}, \forall_j \qquad (2)$$

3. Load Balancing by minimizing the cardinality distance between the workload of any pair of machines:

$$F_3 = min_{\hat{k}_j} \sum_{j=1}^{n_p} q_{j\hat{k}_j} \sum_{i=1}^{n_m} \sum_{l=(i+1)}^{n_m} |x_{ji\hat{k}_j} t_{ji\hat{k}_j} - x_{jl\hat{k}_j} t_{jl\hat{k}_j}| \qquad (3)$$

The overall multi-objective mathematical model of FMS can be formulated as follows:

$$solve for \quad F_1, F_2, F_3$$

s.t.

$$\sum_{\hat{k}_j=1}^{k_j} q_{j\hat{k}_j} = 1$$

$$x_{ji\hat{k}_j} \in \{0,1\};\ q_{j\hat{k}_j} \in \{0,1\};\ n_{ji\hat{k}_j} \geq 1;\ t_{ji\hat{k}_j} \geq 0$$

However utilizing equations 1, 2 and 3 directly as a mathematical programming formulation is too complex. Therefore, for this class of problems, it is more practical to seek a heuristic solution rather than insisting on the optimal. The following section investigates finding such a solution using a Genetic Algorithm heuristic solution.

3 A GA ALGORITHM FOR FMS

Genetic Algorithms **(GA's)** are a class of optimization algorithms that seek improved performance by sampling areas of the parameter space that have a high probability for leading to good solutions [9]. The inherent characteristic of Genetic Algorithms demonstrates why Genetic search may be well suited to multiple-objective optimization problems. The basic feature of Genetic Algorithms is multiple directional and global search through maintaining a population of potential solutions from generation to generation. The population-to-population approach is useful when exploring Pareto solutions. It is important to refine several components such as the encoding method recombination method, fitness assignment, and constraint handling to obtain effective implementations to the given problem. Therefore, when considering how to adapt Genetic Algorithms to multiple-objective optimization problems we have to determine the fitness value of individuals according to multiple objectives. In our implementation we have combined a Pareto-based approach with an adaptive weighted sum technique for tackling the multi-objective flexible manufacturing systems problem.

3.1 GENETIC ALGORITHM MODULES

There are essentially four basic components necessary for the successful implementation of a Genetic Algorithm. At the outset, there must be a code or scheme that allows for a bit string representation of possible solutions to the problem. Next, a suitable function must be devised that allows for a ranking or fitness assessment of any solution. The third component, contains transformation functions that create new individuals from existing solutions in a population. Finally, techniques for selecting parents for mating, and deletion methods to create new generations are required. We will restrict our discussion on the first three modules for flexible manufacturing systems.

As seen in Figure 1, the chromosome representation is defined as a series of operations for all parts involved. Each gene in the chromosome represents the machine type that can possibly process a specific operation. The assignment of machine types to operations is made by randomly generating random numbers within the range of the total number of machines available. In Figure 1, we have a system consisting of 3 parts. Part P_1 requires 3 operations $\{O_2,O_4,O_6\}$ which can be processed by M_2, M_3 and M_1 respectively. The second part requires four operations $\{O_1,O_2,O_5,O_6\}$ which are assigned to Machines M_1 and M_2 and part P_3 requires two operations O_4 and O_6 which are handled by machines M_3 and M_1 respectively. The advantages of this representation scheme are the simplicity and the capability of applying standard operators. However an offspring may not be feasible and thus some special repair heuristics are used to modify the chromosomes to become feasible. Genetic Algorithms

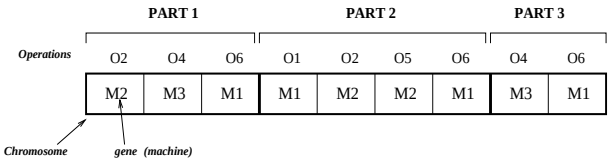


Figure 1: Chromosome Representation

work by assigning a value to each string in the population according to a problem-specific *fitness* function. For the flexible manufacturing problem, the evaluation function measures the goodness of any solution in terms of minimizing part transfer and balancing the workload of the machines.

For any part P_j , **part transfer** is $(\gamma_j - 1)$ with γ_j being the number of different machines assigned to part P_j. Thus, a normalized part transfer function for all the parts can be given as:

$$f_{trans} = \frac{\sum_{j=1}^{parts}(\gamma_j - 1)}{\Gamma}$$

where Γ is an upper bound for the total number of part transfer which is calculated as follows:
$\Gamma = \sum_{j=1}^{parts}(\phi_j - 1)$ and ϕ_j is the number of operations for part P_j.

Machine imbalance can be measured in terms of the deviation, δ_i, of the number of operations performed by machine M_i from the average. Thus a normalized measure of machine imbalance can be given by

$$f_{baln} = (\frac{\sqrt{SSE}}{\sum_{j=1}^{parts}\phi_j}) \times n_m$$

where, n_m is the total number of machines, SSE is the sum of the squared deviations, given by $SSE = \sum_{i=1}^{machines}\delta_i^2$. The fitness assignment strategy of our implementation uses a weighted based fitness function. For an individual χ_u, the score or fitness can be given

by:

$$Score(\chi_u) = (w_{tran} \times (1.0 - f_{trans})) + \frac{w_{baln}}{f_{baln}}$$

where w_{tran} and w_{baln} are the weights assigned to transfer and balance objective functions respectively. Operators in the reproduction module, mimic the biological evolution process, by using unary (mutation type) and higher order (crossover type) transformation to create new individuals. *Mutation* is simply the introduction of a random element, that creates new individuals by a small change in a single individual. When mutation is applied to a bit string, it sweeps down the list of bits, replacing each by a randomly selected bit, if a probability test is passed. On the other hand, *crossover* recombines the genetic material in two parent chromosomes to make two children. Crossover begins by randomly choosing a cut point K where $1 \leq K \leq L$, and L is the string length. The parent strings are both bisected so that the leftmost partition contains K string elements, and the rightmost partition contains $L-K$ elements. The child string is formed by copying the rightmost partition from parent P_T^1 and then the leftmost partition from parent P_T^2. Figure 2 shows an example of applying three types of

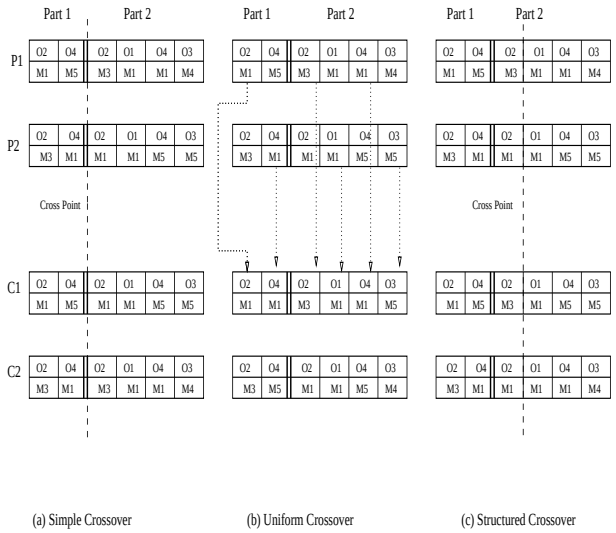


Figure 2: Crossover Operator

crossover operators implemented in this paper. In the simple crossover strategy the cut point(s) are set at the part delimiter such that a complete part is transfered from a parent to a child. In uniform crossover every other gene is received from a different parent as seen in Figure 2b. In the third crossover strategy, the operator begins by randomly choosing a cut point in the string (or multi-points in the chromosome) as described above (see Figure 2c). If all initial solutions are feasible then these crossover strategies lead to complete feasible solutions. Due to the mutation operator, some chromosome may become infeasible (i.e an operation assigned to a machine that cannot handle the operation). In this situation a simple heuristic technique is used to repair the chromosome by assigning the correct machine to the designated operation.

3.2 GA IMPLEMENTATION

In our weighted-sum approach we assign weights to each objective function and combine the weighted objectives into a single objective function as explained in Section 3.1. To fully utilize the power of the Genetic Algorithm we use several approaches: (i) fixed-weight approach, (ii) random-weight approach, and (iii) adaptive weight approach. In the fixed-weight

```
GENETIC ALGORITHM
1. Encode Solution Space
2.(a) set pop_size, max_gen, gen=0;
  (b) set cross_rate, mutate_rate;
3. Initialize Population.
4. For phase ≤ max_phases
    Adapt Weights of Obj Functions
    While max_gen ≥ gen
      Evaluate Fitness
      For (i=1 to pop_size)
        Select (mate1,mate2)
        if (rnd(0,1) ≤ cross_rate)
          child = Crossover(mate1,mate2);
        if (rnd(0,1) ≤ mutate_rate)
          child = Mutation();
        Repair child if necessary
      End For
      Add offsprings to New Generation.
      Save Best δ Solutions
      gen = gen + 1
    End While
5. Return best chromosomes.
```

Figure 3: A Genetic Algorithm for FMS

approach, weights are not changed during an entire evolutionary process. Weights are determined apriori to give selective pressure towards the part transfer objective function. In the random based implementation weights are randomly reset at each step in the selection procedure to give an even chance to all possible combinations. Finally, in the adaptive weight approach, weights are adjusted adaptively based on the current generation to obtain search pressure toward part transfer while balancing the work-load. Figure 3

shows a Genetic Algorithm implementation for flexible manufacturing systems. The algorithm begins with an encoding and initialization phase during which each string in the population is assigned a uniformly distributed random point in the solution space. In the first phase the system assigns a large weight to the first objective function (i.e w_{trans}) and the values of the objective functions f_{trans} and f_{baln} are calculated and set to f_{trans}^{phase1} and f_{baln}^{phase1} respectively. In the next few phases the weights of the objective functions are adjusted adaptively such that the value of f_{trans} is within a δ% tolerance of f_{trans}^{phase1}. Each iteration of the genetic algorithm begins by evaluating the fitness of the current generation of strings. A new generation of offspring is created by applying crossover and mutation to pairs of parents who have been selected based on their fitness (the function rnd(0,1) basically returns a random number between 0 and 1). The algorithm terminates after some fixed number of iterations.

4 RESULTS

The Genetic Algorithm code was developed on a SUN Sparc Ultra II workstation running Solaris 8. The code was written in C and compiled using GNU g++ version 2.95.2. Table 1 shows several benchmarks that have been used for evaluating the performance of the Genetic Algorithm. These benchmarks were randomly generated with different number of machines, parts and operations. In Table 1 the second column gives the total number of machines involved and the possible operations performed by each machine. The third column specifies the number of parts that need to be manufactured and the operation required by each part. The rest of the table gives the maximum operations to be performed and the average operations performed by each machine. Figure 4 presents the convergence rate of the Genetic Algorithm for the two objective functions, namely the part transfer and balance

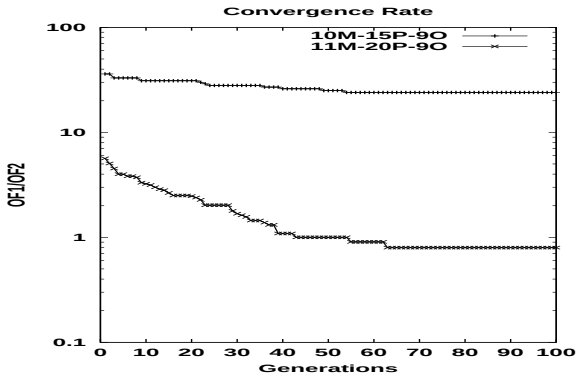


Figure 4: GA Convergence

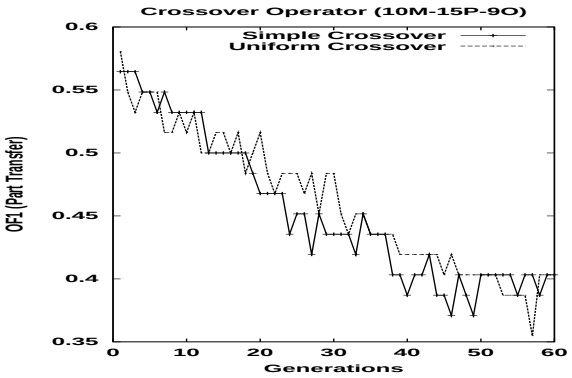


Figure 5: Performance of different Crossover methods

objectives. The figure clearly indicate that after 60 generations there is no improvement in solution quality. Figure 5 presents the performance of two crossover techniques proposed for this paper. The graph clearly indicates that uniform crossover performs better than simple crossover For small benchmarks the quality of solutions obtained from the three crossover operators are quite similar. As the benchmarks increase in size the performance of simple crossover and structured crossover deteriorates. Figure 6 shows the results obtained as a function of the mutation rate. The graphs clearly show that as we increase the mutation rate the two objective functions deteriorate. Mutation rates in the range of 1-5% give the best results. This is obvious since increasing the mutation rate beyond 5% leads to random walks of the solution space and results in unproductive wandering. Figure 7 plots the problem in the criterion space. It is evident from the figure that the search takes place in multiple directions versus a fixed direction as would be the case in a fixed weight approach. Table 2 shows the results obtained when running the Genetic Algorithm by optimizing each objective function separately. The Table is organized as

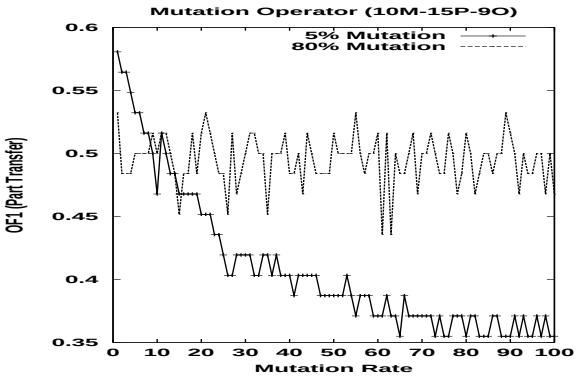


Figure 6: Affect of Mutation Rates

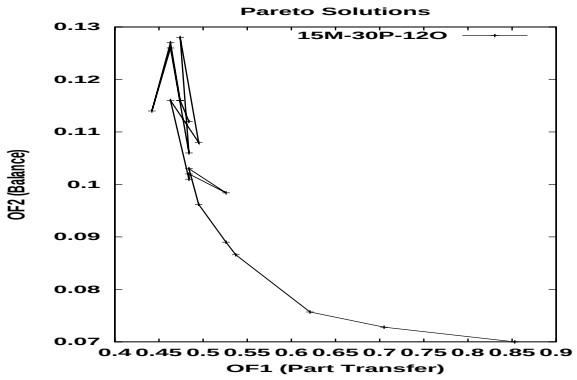


Figure 7: Objectives in Criterion Space

follows: The first column (Prob) indicates the name of the benchmark. The second column (ObjT) specifies the type of objective function being optimized where **PT** stands for part transfer function and **BAL** stands for machine work-load balance. The third and fourth columns specify the objective function values for part transfer and machine work-load balance respectively. The fifth column indicates the total number of machines involved in processing the parts (i.e part transfers involved). Finally, the sixth and seventh columns specify the number of machines involved in processing a specific part and the total number of operations performed by each machine respectively.

We would notice from the table that whenever we optimize the first objective function (i.e part transfer) the less the number of machines involved (i.e sixth column) and the excessive imbalance in the operations performed by the machines. Table 3 shows the results obtained when running the Genetic Algorithm by optimizing both objective functions together. It is clear from the results that both techniques are very competitive. As the size of the benchmarks increase the performance of the adaptive weight technique surpasses that for the fixed weight approach. For example in the 10 machine example the number of part transfer is reduced by two and in the last benchmark (15M30P12O) the systems achieves a reduction in part transfer and a better balance when using the adaptive approach over that based on fixed weights. We should notice that the largest benchmark used in this paper involves fifteen machines, thirty parts and twelve operations. Currently we are seeking real life benchmarks that involve a large number of parts and operations to quantify the computation ability of the Genetic Algorithm on such large benchmarks. We anticipate that the computational complexity will depend entirely on the population and generation size used to solve the problem. Running a Genetic Algorithm entails setting a number of parameter values. As the benchmarks increase in size we have to make sure that our algorithm has the capability of adaptively tuning the parameters to achieve robustness and good performance.

5 CONCLUSIONS

This paper presented a Genetic Algorithm solution for flexible manufacturing systems. The use of evolutionary computation methods for manufacturing optimization is expanding. The amount of work indicates that evolutionary computation methods have established themselves as a useful optimization technique in the manufacturing field, despite the fact that their theoretical foundation are still debated. Results obtained indicate that our Genetic Algorithm implementation achieves excellent results with respect to part transfer and balancing the work among the machines. Future work should compare this Genetic Algorithm implementation with other heuristic search approaches (possibly a memtic algorithm that combines a Genetic Algorithm with local search techniques) and the feasibility of integrating Genetic Algorithms with Multi Agent Systems. We also seek to include sequencing constraints and tools costs in the objective function in our future implementation.

References

[1] C. Dimopoulos and A. Zalzala, "Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions, and Comparisons", *IEEE Transactions on Evolutionary Computation*, vol. 4, n. 2, pp. 93–113, 2000.

[2] J. Blazewicz, K. Ecker, E. Pesch and G. Schmidt, *Scheduling Computer and Manufacturing Processes, 2nd Ed*, Springer, Berlin, New York, 2001.

[3] J. Chen and S. Ho, "Multi-Objective Evolutionary Optimization of Flexible Manufacturing Systems", in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1260–1268, San Francisco, California, 2001, Morgan Koffman.

[4] K. Baker, *Introduction to Sequencing and Scheduling*, Kluwer Academic Publishers, Boston, 1974.

[5] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, New York, 2000.

[6] S. French, *Sequencing and Scheduling*, ellis Horwood Limited, England, 1982.

[7] E. Coffman, *Computers and Job Scheduling*, John Wiley, New York, 1976.

[8] H. Ghenniwa and M. Kamel, "Minimizing Part Transfer and Balancing the Workload in Assigning Parts to Machines Having Overlapping Capabilities", *Journal of Flexible Manufacturing Systems*, 1998.

[9] V. Venkatasubramanian and I.P. Androulakis, "A Genetic Algorithm Framework for Process Design and Optimization", *Computers Chemical Engineering*, vol. 15, n. 4, pp. 217–228, 1991.

Prob	Mach	Part	Oper	Ops/Mach	
				MAX	$\overline{x}$
3M1P5O	3 (1,2,3,4);(3,4,5);(4,5)	1 (1,2,3,5)	5	4	2.3
3M2P5O	3 (1,2,3,4);(3,4,5);(4,5)	2 (1,2,3,5);(2,3,5)	5	4	3
5M2P7O	5 (1,2,7);(1,2);(2,3) (2,4,5),(1,4,6)	2 (7);(1,2,3,4,5,6)	7	3	2.1
5M5P6O	5 (2,3,4,5);(2,3,4,6); (1,2,3,4);(1,2,4,6);(1,2,5)	5 (1,2,3,4,5);(1,2,3,4,6) (1,2,5);(2,3,4);(2,3,4,6)	6	3	3.2
5M10P8O	5 (1,2,3);(4,5,6);(2,3,7) (5,6);(7,8)	10 (1);(4);(8);(2,3,4,5) (2,4,6,8);(4,6,7,8) (2,3,5,6,7);(1,2,4,5,7) (1,4,8);(3,5,6)	8	3	3.2
6M6P10O	6 (1,2,3);(4,5);(1,2,3,4,5) (6,7,8);(9,10);(6,7,8,9,10)	6 (1,2,3,4,5);(1,2,3) (4,5);(6,7,8,9,10) (6,7,8);(9,10)	10	5	3.5
7M7P7O	7 (7);(6);(5);(4);(3);(2);(1)	7 (1);(2);(3);(4);(5);(6);(7)	7	1	1
10M15P9O	10 (1,2,3);(1,4,6);(5) (1,2,3,4,5);(1,7,8,9) (1,2,7,8,9);(1,2,3,4,7,8,9) (5,7,8);(1,3,5);(2,4,6)	15 (1,2,3);(1,2,4);(1,2,5) (1,3,6,7);(1,4,7,8) (2,3,5,8,9);(4,5,6,7,9) (1,2,4,5,9);(1,3,5,7,9) (2,4,6,7,8,9);(1,2,3,4,5,6) (2,4,5,6,7,8);(1,2,3,4,5,6,7) (2,3,4,5,6,7,8) (1,2,3,4,5,6,7,8)	9	7	4.2
11M20P9O	11 (1,2);(3,4);(5,6);(7,8) (1,2,3);(3,4,5);(5,6,7) (7,8,9);(1,2,3,4) (5,6,7,8);(1,2,3,4,9)	20 (1,2,3);(1,2,4);(1,2,5) (3,6,7);(4,7,8);(5,8,9) (6,7,9);(4,5,9);(6,8,9) (5,7,9);(7,8,9);(3,4,5) (4,5,6);(6,7,8);(2,4,6) (1,3,5);(1,3,7);(1,3,9) (2,5,7,9);(1,3,5,7)	9	5	2.9
15M30P12O	15 (1,2);(3,4);(5,6);(7,8) (1,2,3);(3,4,5);(5,6,7) (7,8,9);(1,2,3,4) (5,6,7,8);(1,2,3,4,9) (9,10);(8,11,12);(8,9,10,11) (7,8,9,10,11,12)	30 (1,2,3);(1,2,4);(1,2,5) (3,6,7);(4,7,8);(5,8,9) (6,7,9);(4,5,9);(6,8,9) (5,7,9);(7,8,9);(3,4,5) (4,5,6);(6,7,8);(2,4,6) (1,3,5);(1,3,7);(1,3,9) (2,5,7,9);(1,3,5,7) (1,2,9,10);(2,4,8,11) (3,5,9,11);(4,6,10,12) (5,7,8,9,10);(2,3,8,9,10,11) (2,3,8,10,11,12);(1,2,3,8,10,11,12) (2,4,6,8,10,11,12);(1,2,3,4,5,6,7,8) (5,6,7,8,9,10,11,12)	12	6	3.2

Table 1: Benchmarks used as test cases

Prob	ObjT	Obj1	Obj2	T-M	P-P-M	O-P-M
3M1P5O	PT	0.3333	1.00	2	2	2,2,0
	BAL	0.6666	0.5773	3	3	2,1,1
3M2P5O	PT	0.4000	1.2909	4	2,2	4,2,1
	BAL	0.8000	0.5773	6	3,3	3,2,2
5M2P7O	PT	0.4000	1.0954	4	1,3	1,0,1,2,3
	BAL	0.6000	0.6324	5	1,4	1,1,1,2,2
5M5P6O	PT	0.1333	1.4142	7	2,2,1,1,1	2,4,6,3,5
	BAL	0.7333	0.0001	16	3,5,2,3,3	4,4,4,4,4
5M10P8O	PT	0.4761	2.4083	20	1,1,1,2,3,2,2,3,3,2	4,10,8,4,5
	BAL	0.7142	0.7745	25	1,1,1,3,4,2,3,5,3,2	6,7,5,7,6
6M6P10O	PT	0.000	3.9157	6	1,1,1,1,1,1	0,0,10,3,0,7
	BAL	0.5714	0.5773	14	3,2,2,3,2,2	3,3,4,4,3,3
7M7P7O	PT	-	2.4494	7	1,1,1,1,1,1,1	7,0,0,0,0,0,0
	BAL	-	0.0	7	1,1,1,1,1,1,1	1,1,1,1,1,1,1
10M15P9O	PT	0.3387	11.157	36	1,1,1,3,1,2,3,3,2,3, 3,3,3,3,4	0,3,0,35,0,11,21,0,1,6
	BAL	0.8225	0.9486	66	3,2,3,4,4,3,5,4,4,5, 5,5,6,6,7	8,9,7,8,7,8,8,7,7,8
11M20P9O	PT	0.3809	6.1864	37	1,1,2,2,2,3,2,2,2, 2,1,2,3,1,2,2,2,1,2,2	0,0,0,0,3,3,9,7,7,13,20
	BAL	0.9047	0.7977	58	2,2,2,3,3,3,3,3,2, 3,3,3,3,3,3,3,3,3,4,4	5,6,5,5,6,6,6,6,5,6,6
15M30P12O	PT	0.5052	6.6282	78	1,2,2,2,2,2,2,3,2,3, 2,1,2,2,2,2,2,2,4,2, 2,3,3,3,2,3,5,6,5,4	1,0,0,2,16,14,16,6,11 12,13,1,1,15,17
	BAL	0.853	1.88	111	3,3,2,2,3,3,3,3,3,3, 3,3,3,3,3,3,2,3,4,4, 4,4,4,4,4,4,,9,6,7,6	6,9,7,6,12,10,10,8,8 8,10,7,6,11,7

Table 2: Comparisons Based on Optimizing a Single Objective Function

Prob	ObjT	Obj1	Obj2	T-M	P-P-M	O-P-M
3M1P5O	FWA	0.333	1.000	2	2	2,2,0
	AWA	0.333	1.000	2	2	2,2,0
3M2P5O	FWA	0.800	0.577	6	3,3	3,2,2
	AWA	0.800	0.577	6	3,3	3,2,2
5M2P7O	FWA	0.400	0.894	4	1,3	1,0,2,2,2
	AWA	0.400	0.894	4	1,3	1,0,2,2,2
5M5P6O	FWA	0.733	0.0	16	3,5,2,3,3	4,4,4,4,4
	AWA	0.733	0.0	16	3,5,2,3,3	4,4,4,4,4
5M10P8O	FWA	0.619	0.774	23	1,1,1,3,4,2,2,4,3,2	6,7,6,7,5
	AWA	0.619	0.774	23	1,1,1,3,4,2,2,4,3,2	6,7,5,7,6
6M6P10O	FWA	0.142	0.577	8	2,1,1,2,1,1	3,4,3,3,4,3
	AWA	0.142	1.290	7	2,1,1,1,1,1	3,2,5,3,2,5
7M7P7O	FWA	-	2.449	7	1,1,1,1,1,1,1	7,0,0,0,0,0,0
	AWA	-	2.449	7	1,1,1,1,1,1,1	7,0,0,0,0,0,0
10M15P9O	FWA	0.370	4.868	38	2,2,2,3,2,2,3,3 2,2,2,3,3,4,3	1,9,2,18,5,10,12,4,9,7
	AWA	0.338	6.640	36	1,2,2,3,2,2,3,2 2,2,3,3,3,3,3	2,6,0,23,5,15,8,4,3,11
11M20P9O	FWA	0.380	3.965	36	1,1,2,2,2,2,2,2,2,2,1 2,2,1,2,2,2,1,3,2	1,1,2,0,5,5,9,10,9,10,10
	AWA	0.380	3.030	36	1,1,2,2,2,2,2,2,2,2,1 1,2,1,2,2,2,1,3,3	1,0,4,3,8,8,7,8,7,7,9
15M30P12O	FWA	0.453	7.04	73	1,1,2,2,2,2,2,2,2,2, 2,1,2,2,2,2,2,3,3,3, 3,4,3,3,3,2,4,3,4,4	2,0,0,1,18,15,18,5 9,6,13,0,5,20,13
	AWA	0.410	6.36	69	1,1,2,2,2,2,2,2,2,2, 1,1,2,2,2,2,2,2,3,3, 2,3,3,3,3,2,5,3,4,3	2,0,1,0,19,17,16,9 7,7,14,5,4,16,8

Table 3: Comparisons Based on Optimizing a Combined Objective Function

An Evolution Strategies Based Approach to Image Registration

Jian Zhang
EECS Dept.
Tulane University
New Orleans, LA 70118

Xiaohui Yuan
EECS Dept.
Tulane University
New Orleans, LA 70118

Bill P. Buckles
EECS Dept.
Tulane University
New Orleans, LA 70118

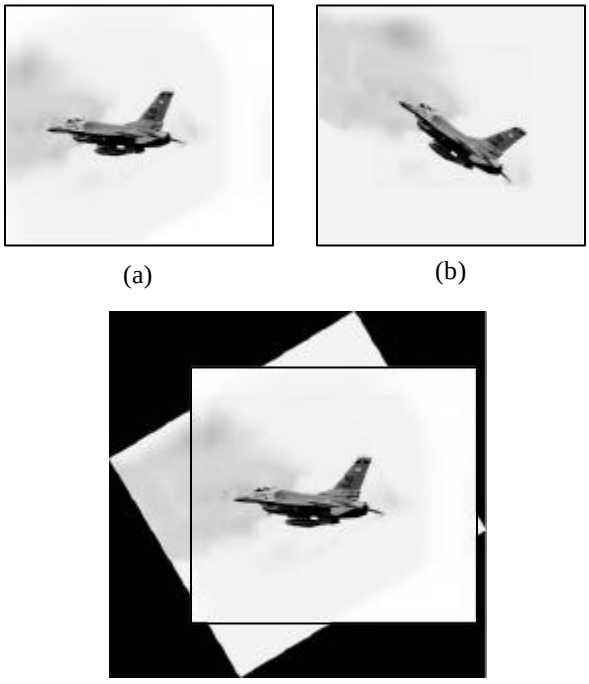
Abstract

An image registration approach based on Evolution Strategies is proposed. In image registration, an invariant reference needs to be established within each source image, which is unavailable in many cases. To solve this problem, feature configuration (which is defined as the cluster of feature vectors on an image representing homogeneous feature distribution,) is employed to describe the object inside the scene. Instead of finding the correspondence of the entire image, the spatial relationships of the feature configuration in every source image are discovered with Evolution Strategies (ES). While one approach, even this one, may not be suitable for every image domain, the ES approach has many advantages. Compared to some methods, it is computationally effective; compared to other, it is capable of discovering transformations of larger scope (e.g., greater rotation angles or translation distances etc.) The search structure we use is an ellipsoid. The results from various images prove it to be an efficient and effective method.

1 INTRODUCTION

A fundamental image-processing task, image registration matches two or more images such that features from each individual source are aligned against the same reference. Virtually all image understanding tasks, such as image fusion, object recognition etc., require image registration procedure as pre-processing. It is a particular important issue faced in almost all remote sensing domains. In medical imaging, a patient's cranial scan must be matched with medical atlas images as well as previous scans of the same patient. In Earth science, the extent of deforestation can be determined only if the present image can be compared to ones from previous time periods. These and many other examples exist that assert the need to put multiple images into pixel-by-pixel correspondence.

Intensive research has been devoted to find the most effective and efficient registration methods [2, 8, 14, 17]. Approaches proposed include control point based methods [13], frequency feature based methods [1, 16], mutual information based methods [9] etc. Figure 1 illustrates an image registration example. Two images are superimposed one on the other, as shown in Figure 1(c), based on the transformation discovered by the registration process.



(a) (b)

(c)

Figure 1. Image (a) and (b) are two source images focusing on the same object but containing different scene. (c) is the superimposed image after registration.

Image registration can be viewed as a combination of different choices of feature space, search space, searching strategy, and similarity measure [12]. The feature space extracts information from the source images, which provides a quantitative space for transformation. All possible transformations form a search space, such that given a pair of images a sequence of transformations can be found in the search space to put these images in correspondence. The searching strategy defines rules of

finding the next transformation. The registration accuracy is assessed by the similarity measure. The registration process proceeds iteratively by searching and applying transforms until the similarity measurement is satisfied.

Generally, a similarity measure is required to evaluate the accuracy of aligned image after transformation. During registration, the similarity measurement is iteratively computed and improved by adjusting the transformation. However, most similarity measurements are computationally expensive even on moderately sized images, which makes registration inefficient, especially when there is a large initial difference between images. Here we propose a fast, control point free and feature based image registration method, which is based on Evolution Strategies. The objective is to find the correspondence between two or more images having a spatial difference caused by rigid transformation.

The principal idea is to search for the correspondence between some specific feature configurations instead of the correspondence across the whole image. Given the specific feature description, Evolution Strategies is employed to find the feature configuration, defined in section 3.2, on all source images. A reference image is randomly selected thereafter, in which the feature configuration is located and is used for registering other images. However, the region defined by feature configurations may not enclose the same feature distribution. Therefore, a refinement process based on the reference image is followed to adjust the feature configuration such that similar feature enclosure is ensured. Finally, the transformation functions are determined by comparing the spatial characteristic of configurations, which is represented in the form of feature ellipse, against that of the reference configuration and are used to register those images.

The rest of this article is organized as follows. In section 2, a short review of Evolution Strategies is given. Section 3 presents the retrospective image registration problem and illustrates Evolution Strategies based image registration scheme. Experiments are illustrated in section 4. The paper is concluded with discussion in section 5.

2 ES SHORT REVIEW

Evolution Strategies (ESs) are algorithms that imitate the principles of natural evolution as a method to solve parameter optimization problem [4, 3, 18]

The goal of a parameter optimization problem $f: M \subseteq R^n \to R$, $M \neq 0$, where f is called *objective function*, is to find a vector $x^* \in M$ such that

$$\forall x \in M : f(x) \geq f(x^*) \tag{1}$$

where $f^* := f(x^*) > -\infty$ is called a *global minimum*; x^* is the *global minimum point*. M is the set of feasible points for a problem. In correspondence with global minimum, a *local minimum* $\hat{f} = f(\hat{x})$ is defined by the following condition:

$$\exists \varepsilon > 0 \quad \forall x \in M \quad : \quad \|x - \hat{x}\| < \varepsilon \Rightarrow \hat{f} \leq f(x) \tag{2}$$

Coexistence of global minimum and several local minima make optimization a non-trivial problem.

Each ES individual represents a vector within the domain of the objective function f. Each x_i, $i = 1, 2, \ldots, n$, is termed an *object variable* and is represented as a real value in the individual. Evolution Strategies is essentially randomized hill climbing, which makes it a non-deterministic optimization strategy. Hill climbing necessitates the resolution of two issues at each iteration -- (i) the direction to move and (ii) the distance (step size). These issues are resolved by embedding control variables into individual and an ES individual is, organized as object variables and control variables, illustrated below.

$$\{x_1, x_2, \ldots x_l ; \sigma_1, \sigma_2, \ldots, \sigma_m ; \alpha_1, \alpha_2, \ldots, \alpha_p\} \tag{3}$$

where the σ's and α's are control variables.

An object variable should be considered the mean of a normally distributed random variable. Under that interpretation, each σ_i is a standard deviation for an object variable. Thus $m \leq l$. (If $m < l$ then σ_m applies to all x_j, $m \leq j \leq l$.) Each α_i is a surrogate for the covariance of two object variables. α is organized as an upper triangular matrix as is a covariance matrix. (That is to say, $p = (2l-m)(m-l)/2$.) The correspondence between α_{ij}, $i, j \leq m$ and the covariance, c_{ij} is

$$\tan(2\alpha_{ij}) = \frac{2c_{ij}}{\sigma_i^2 - \sigma_j^2} \tag{4}$$

An interpretation of an ES organism is an l-dimensional jointly distributed normal variate with mean x and the standard deviation σ. The orientation of the distribution in l-space is determined indirectly by the covariance and directly by α.

The incorporation of control variables into the individual representation establishes a two-level self-learning process, since not only the object variable adapts according to the objective function, but also the control variables change with respect to the actual topological requirements. In other words, the control variables make up an internal model of the objective function, which is learned on-line during optimum seeking without an additional measure of fitness.

The ES algorithm is formulated in the language of biology as follows:

Step 1. A given population consists of m individuals. Each is characterized by its genotype consisting of n genes, which unambiguously determine the fitness for survival.

Step 2. By mutation and recombination operations, each individual parent produces l/m offspring on average, so that a total number of λ offspring individuals are available.

Step 3. Select the best of the offspring to form parents of the following generation and continue at Step 1.

3 IMAGE REGISTRATION WITH ES

Let I_1 and I_2 denote two image matrices, then image registration, under *Cartesian Coordinates*, can be expressed as:

$$I_1(x,y) = g(I_2(f(x,y))) \tag{5}$$

where function *f(.)* is a 2D spatial-coordinate transformation, i.e. *f(.)* maps two spatial coordinates, *x* and *y*, to new spatial coordinates *x'* and *y'*, and function *g(.)* is a 1-D intensity or radiometric transformation.

The registration problem is to find the optimal transform functions *f(.)* and *g(.)*, namely spatial and intensity transformation, so that the images are aligned under the same coordinates system. The intensity transformation *g(.)* is not always necessary, and if g(.) is need, a lookup table is usually sufficient [12].

3.1 RETROSPECTIVE REGISTRATION

Restrospective registration is required when an image is obtained without the benefit of a fiducial reference system, e.g. a battle field surveillance image or MRI cranial scan image. In this case, a reference is not included in the source image.

Let point (x, y) denotes the central point of the object and let the rotation angle be θ. In order to describe the scaling transformation, let' s denote a point in an image in homogeneous coordinates (x, y, s), with (x/s, y/s) being the corresponding *Cartesian* coordinates. Using the homogeneous coordinates, the transformation function *f(x,y)* has the following general form:

$$f_{\theta t_x, t_y}(x,y,s) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ s \end{bmatrix} \tag{6}$$

where θ determines the rotation angle. t_x/s and t_y/s are translations on horizontal and vertical directions. *s* is the scaling weight that also functions as the normalization factor in the homogeneous coordinates.

3.2 REGISTRATION WITH ES

To deal with retrospective registration, an image feature is employed in our scheme. Frequently used features include luminosity, texture, shape, etc. Usually, a picture contains at least one object, which can be distinguished from its background with a set of characteristic features. Such features are distributed in a certain region, i.e., the area defined by the shape of that object. That is, the co-occurrence of these features is only satisfied within the object. Hence, the area that concurrently contains a set of certain features is called the feature configuration. The *feature configuration*, Φ, is defined on image *I* such that a cluster of pixels constitute a close region *P* within which image features, $F_1, F_2, \ldots, F_n$, are uniformly distributed.

$$\begin{aligned} &\forall (x,y),(x',y') \in P \quad and \\ &|I^{F_1,F_2,\ldots,F_n}(x,y) - I^{F_1,F_2,\ldots,F_n}(x',y')| \le \varepsilon, \quad \varepsilon \to 0: \\ &\Phi^{F_1,F_2,\ldots,F_n}(x,y) = \left\{ I(x,y) \mid I(x,y) \in P, P \subseteq I^{F_1,F_2,\ldots,F_n} \right\} \end{aligned} \tag{7}$$

where $I^{F_1F_2,\ldots,F_n}(x,y)$ is the feature vector generated by applying feature filters onto image *I* and function |.| measures the distance of two vectors.

The rigid transformation does not change the feature of an image. That is, given the initial image I_1 and the transformation function *f*, the outcome image I_2 has the same feature as I_1. Thus, the feature configurations of these features in I_1 and I_2 are also spatially related with the same transformation.

$$\Phi_1^{F_1,F_2,\ldots,F_n}(x,y) = f(\Phi_2^{F_1,F_2,\ldots,F_n}(x,y)) \tag{8}$$

Because of its separability, a transformation function can also be expressed as follows:

$$I_2(x,y) = S(R(T(I_1(x,y),\Delta x,\Delta y),\Delta\theta),s) \tag{9}$$

where T(.), R(.) and S(.) denote translation, rotation and scaling respectively. Notice that the translation parameters Δx and Δy are defined with regard to the central point of the image. To prove equation (8), it is sufficient to prove that after translation and rotation, the distance between any two points in the feature configuration remains unchanged, while scaling enlarges the distance by scale of s. Due to space constraints, proofs are not included.

Yuan et al. reported successful feature identification using Evolution Strategies [18]. Inspired by the success of their work, Evolution Strategies is employed to search for the optimal transformation. During the search for image correspondence, instead of evaluating similarity measurements over the whole image, ES identifies the region from each source image that contains a homogeneous feature configuration. To capture the feature configuration, an ellipse structure is used to enclose the maximum homogeneous feature area. Here we call such an ellipse the feature ellipse.

Feature Ellipse

Feature ellipse, Λ, is the search structure used to enclose feature configuration $\Phi^{F_1,F_2,\ldots,F_n}$. Feature vectors inside the ellipse represent the same type of features. That is, it encloses only one type of feature configuration.

$$\begin{aligned} &\Lambda(x_0, y_0, a_1, a_2, \theta \mid \Phi^{F_1,F_2,\ldots,F_n}) \\ &\Rightarrow |I_\Lambda^{F_1,F_2,\ldots,F_n}(x,y) - I_\Lambda^{F_1,F_2,\ldots,F_n}(x',y')| \le \varepsilon \\ &and \quad \varepsilon \to 0 \end{aligned} \tag{10}$$

where $I_\Lambda^{F_1,F_2,\ldots,F_n}(x,y)$ and $I_\Lambda^{F_1,F_2,\ldots,F_n}(x',y')$ are any two different feature vectors enclosed by Λ.

A feature ellipse is determined by the coordinates of the center (x_0, y_0), the lengths of the major and minor axes a_1 and a_2, and the angle θ between the major axis and the horizontal line. These parameters are embedded into ES as objective variables and are organized as $(x_0, y_0, a_1, a_2, \theta)$. Figure 2 illustrates the structure of a feature ellipse. (Notice that ES also uses control variables denoted as σ' s that

establish a relationship with covariance as illustrated in equation (2) and directly determine the orientation of the distribution in *l*-space.)

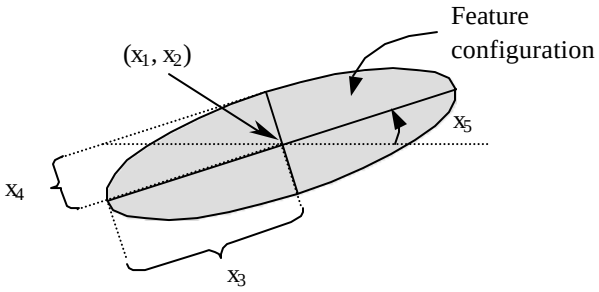


Figure 2. Encoding parameters of ellipse into objective variables in ES. Region inside the ellipse contains homogeneous configuration of certain feature

The optimum to the objective function of one ES application is a feature ellipse that maximizes its area under the constraint that only one feature configuration is included. Or in the other words, it minimizes the difference among the feature vectors inside the ellipse while enlarging its area.

Parameters Estimation

Given the feature description, ES is applied to all source images to find the feature configuration $\boldsymbol{F}_i$, $i=1,\ldots,n$. The feature configuration $\boldsymbol{F}_p$ located from one of the source images, which is randomly selected, is used as the reference for registering other images. This arbitrarily selected image is distinguished as the *reference image*. Figure 3 illustrates the diagram of ES based image registration.

Although the same feature description is used to guide the ES search, the outcome $\boldsymbol{F}_i$ usually does not give the same quantitative measurement, which appears as slightly different feature ellipses as shown in Figure 4(c) and 4(d). This is because of the probability-controlled randomness of reproduction process in ES. Therefore, a refinement process is followed. The refinement takes $\boldsymbol{F}_p$ as a reference and adjusts the feature configuration $\boldsymbol{F}_i$ of image I_i, such that the quantitative measurements, e.g. mean and variance, of each feature configuration, $\boldsymbol{F}_p$ and $\boldsymbol{F}_i$, match.

After searching, the feature configuration is reported as the parameters of an ellipse. The transformations f^i, $i=2, \ldots, n$, are determined by comparing the spatial parameters of ellipses with that of the reference.

4 EXPERIMENTS

In our experiments, the parent population size is chosen as 50 and the descendant population size is 300. These population sizes are used in both the initial search as well as the later refinement step. For the purpose of accelerating the search process, discrete recombination on object variables and panmictic intermediate recombination of control variables is preferred [21, 3, 22].

Moments are used in our experiments as the quantitative measurements of the feature configuration. Generally, only the first few moments are required to differentiate between signatures of clearly distinct shapes [7, 6].

$$m_n(v) = \sum_{i=1}^{K} (v_i - m)^n \, p(v_i) \tag{11}$$

The quantity m is recognized as the mean of v, which represents the gray level of an image, and m_2 as its variance. $p(v_i)$ is the normalized amplitude histogram at gray level v_i. Besides the moments, the *compactness of the ellipse*, which is the ratio of the major axis and the minor axis, is considered as another constraint of the feature ellipse. Given the match of the compactness measurement, the ratio of the axes from two feature ellipses exposes the scale factor of the image. Let the axes of two feature ellipses be (a_1, a_2) and (a_1', a_2'), where a_1, a_1' are the length of major axes and a_2, a_2' are the length of minor axes. The scaling factor is determined by the average of the ratio as shown below.

$$s = \frac{a_1/a_1' + a_2/a_2'}{2} \tag{12}$$

Figure 4 illustrates the process with sample images Child_1 and Child_2. An initial feature ellipse is randomly chosen for each source image, which are drawn and shown in Figure 4(a) and 4(b). After approximately 50 generations, both feature configurations are found, as shown in Figure 4(c) and 4(d).

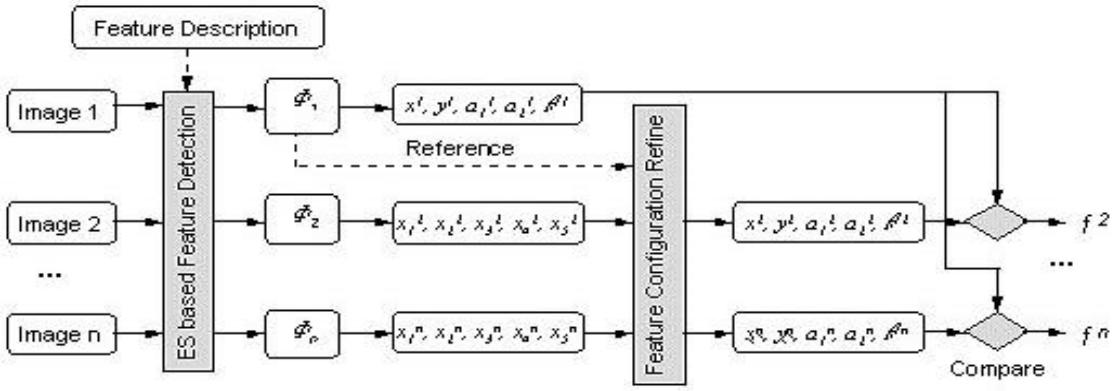
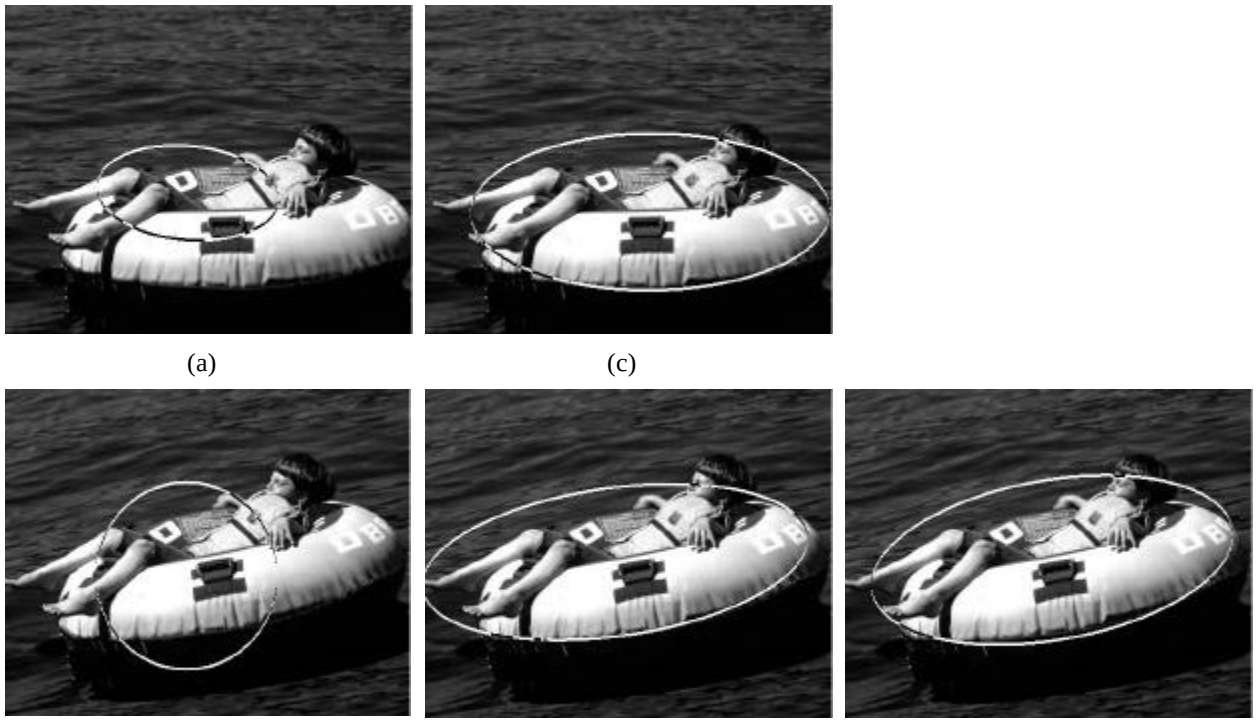


Figure 3. ES based image registration. Transformation function f^i represents the spatial relationship between image i and image 1, which is selected as the reference image.



(a) (c)

(b) (d) (e)

Figure 4. Registration stepwise images are shown. Image (a) and image (b) illustrate source images. Image (c) and (d) illustrate the feature configuration found by ES. Image (e) is the outcome of the refinement step.

Table 1. Experimental images and estimated transformation parameters. The noisy images are distorted with 20% noise.

Test Image		Rotation			Translation		
		Image 1	Image 2	$\Delta\theta$	Image 1	Image 2	$(\Delta x, \Delta y)$
Noise Free Image	Child	-0.3	-9.4	-9.1	(248, 220)	(228, 182)	(-20, -38)
	Plane	-2.1	28.9	31.0	(180, 211)	(231, 217)	(51, 6)
	Phone	44.5	75.0	30.5	(194, 201)	(173, 180)	(-21, -21)
	River	21.6	5.6	-16.0	(222, 211)	(243, 245)	(21, 33)
	Infrared	-5.1	-31.2	-26.1	(220, 184)	(215, 177)	(-5, -7)
	Lighthouse	86.0	75.5	-10.5	(171, 270)	(221, 278)	(50, 8)
Noisy Image	Child	-0.3	-9.6	-9.3	(235, 220)	(218, 176)	(-17, -44)
	Plane	-0.3	29.7	30.0	(184, 206)	(230, 212)	(46, 6)
	Phone	45.3	75.2	29.9	(193, 201)	(172, 180)	(-21, -21)
	River	24.4	8.1	-16.3	(217, 216)	(238, 245)	(21, 29)
	Infrared	-6.5	-33.6	-27.1	(225, 175)	(215, 173)	(-10, -2)
	Lighthouse	-3.6	-12.8	-9.2	(172, 265)	(221, 276)	(49, 11)

Although two ES process are running under the same feature description, regions covered by two feature ellipses are usually not identical. This is due to the non-deterministic search characteristic of the ES algorithm. Therefore, a refinement process is followed. In this experiment, Child_1 is chosen as the reference image. The mean and variance values of the area inside feature ellipse in image Child_1 are computed and used to refine the parameters of feature ellipse in Child_2. The refinement process uses ES optimization with one more constraint. That is, the difference between the quality measure of feature ellipse in Child_2 and the quality measure extracted from Child_1 is minimized and allow 1% error for mean, 0.1% error for variance and 1% error for the ellipse compactness.

Table 1 lists the estimated transformation on six pairs of test images. Each pair of images is of the same size. The first two columns give the condition and the name of images used in the experiments. The next three columns contain the

estimated rotation parameters, where results in column ‘ Image 1’ are the principal orientation of the feature ellipse located in the first image and column ‘ Image 2’ contains orientation parameters computed from the second image. The column $\Delta\theta$ lists the rotation difference between image pairs. The last three columns contain the translation estimations, which are coordinates of the central points of ellipses found in image pairs and the translation difference, Δx and Δy, between central points. Table 1 also contains the outcomes performed on the same experimental images, except each image is distorted with 20% noise. It is clear that the transformation parameters estimated under noise are very close to those computed with noise-free images.

Figure 5 illustrates two registered images, lighthouse and Child, given the transformation parameters provided in table 1. The registrations are accurate under the judgment of human perspective.

Figure 5. Sample registration results, lighthouse and child. Two source images are superimposed one on the other using the transformation matrix estimated with ES optimization.

Notice, the translation parameters Δx and Δy are not based on the center of the image but the central point of the feature ellipse. Therefore, when registering images, the translation and rotation are performed with regard to the center of the ellipse.

Table 2 illustrates the experimental result on scaling factor estimation. Notice that test images are resized to 80% of the original size. The second and the third columns are the axes’ lengths of the feature ellipses, which are formatted as (length of major axis, length of minor axis). The error estimation is below 0.02.

Table 2. Scaling factor estimation. The test images are resized by eighty percent.

Images	Axes (long, short)		Scale Factor
	100% image	80% image	
Child	(197.0, 83.3)	(172.0, 62.7)	0.813
Plane	(123.7, 38.4)	(95.3, 32.5)	0.808
Phone	(130.0, 55.7)	(103.9, 44.3)	0.797
River	(120.3, 24.7)	(93.7, 20.4)	0.802
Infrared	(181.8, 60.7)	(148.2, 47.1)	0.796
Lighthous	(114.9, 50.6)	(91.4, 40.9)	0.802

5 DISCUSSION AND CONCLUSION

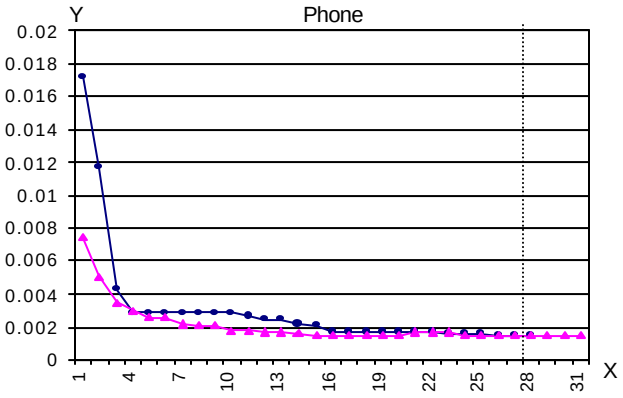
An Evolution Strategies based image registration approach is described in this article. Given the feature of an image remaining unchanged after rigid transformation, a search structure, the feature ellipse, is embedded into each ES individual. The objective function of ES achieves the minimization of differences between the quantitative measurements of feature configurations enclosed with feature ellipses while enlarging its area. The registration scheme contains three steps: search feature configurations under certain feature description, refining feature ellipses by minimizing quantitative measurements differences, and determining the transformation parameters.

Experiments have been performed on various kinds of images including nature scenes, military surveillance images etc. Promising results are also obtained under noisy circumstances. The experiments show the robustness of this approach, which is the result of two aspects. Firstly, the search is performed in the feature space, where the noise is reduced. Secondly, the optimization process, incorporating with the feature quantitative measurements, is insensitive to the noise. Even though the feature configurations found within the noise-free images and within noisy images are different, the transformation relationships discovered are almost identical.

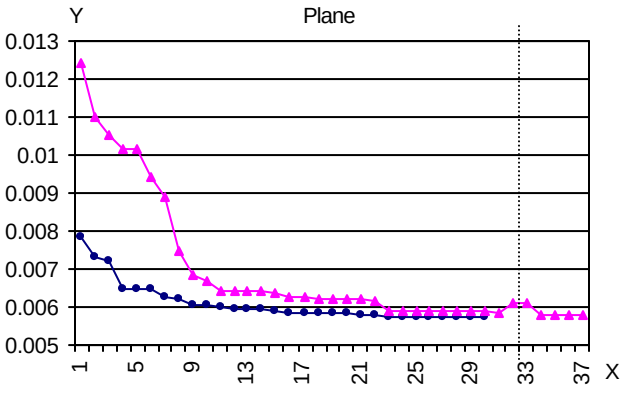
Moreover, since the feature comparison is performed inside a relatively small region, the feature ellipse, the computation expense is reduced. Figure 6 illustrates that the ratio of variance vs. area (enclosed by feature ellipse) changes with regard to the iterations. Figure 6(a) illustrates the optimization process with a Phone image. In the graph, solid circle line records the ratio changes of the reference image. The solid triangle line records the ratio changes of the companion image. (Remember, the reference image is simply an arbitrarily selected source image.) The graph is partitioned with a vertical doted line, where the left half illustrates the searching phase optimization and right half shows the refinement phase progress. Notice that during the first phase of registration, ES is applied individually on each image. Therefore the iterations used in searching are

different. In figure 6(b), which illustrates the optimization process with a Plane image, it is clear that the refinement process adjusts the feature ellipse in the companion image even closer to that of the reference image.

Experiments have been successful on images containing one object, which is distinguishable with a set of features. In cases where image contains more than one similar objects, due to the non-deterministic characteristic of ES, feature configurations representing different but similar objects may be found, e.g., the image shown in Figure 7. Obviously, it is hard to further distinguish among these objects. This difficulty may be overcome by ES with a niching strategy. Zhang et al. [19] described a niching embedded ES for multimodal function optimization, in which successful locating multiple optima is reported. Further study can be done for multi-object involved registration.



(a)



(b)

Figure 6. Variance reduces during searching and refinement. Axis X is the iteration number and axis Y is the ratio of variance vs. area enclosed within feature ellipse. In each graph, reference image is illustrated with solid circle line and the companion image is illustrated with solid triangle line.

(a)

(b)

Figure 7. Confusion of multiple similar objects in image registration. Two similar feature configurations are found yet corresponding to different objects.

Acknowledgments

Support for this work was provided in part by DoD EPSCoR and the Board of Regents of the State of Louisiana under grant F49620-98-1-0351.

References

[1] A.D. Calway, H. Knutsson and R. Wilson, "A Multiresolution Frequency Domain Algorithm for Fast Image Registration", 3rd Int'l Conf. On Visiual Search, April 1992

[2] Haim Schweitzer, "Optimal Eigenfeature Selection by Optimal Image Registration", Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'99), June 1999

[3] Thomas Bäck, "Evolutionary Algorithms in Theory and Practice", Oxford University Press, Oxford (1995)

[4] Thomas Bäck, Frank Hoffmeister, HansPaul Schwefel, "A Survey of Evolution Strategies", Proc. of the Fourth Int. Conf. on Genetic Algorithms, pages 2-9. Morgan Kaufmann, 1991

[5] H. Bulow, L. Dooley and D. Wermser, " Application of Principal Axes for Registration of NMR Image Sequences" , Pattern Recognition Letter 21 (2000) 329-336

[6] Xiaolong Dai, Siamak Khorram, " A Feature-Based Image Registration Algorithm Using Improved Chain-Code Representation Combined with Invariant Moments", IEEE Transactions on Geoscience and Remote Sensing, v37, n5, Sept. 1999

[7] Rafael C. Gonzalez and Richard E. Woods, " Digital Image Processing" , Addison-Wesley Publishing Company, 1992

[8] J. W. Hseih, H. Y.Liao, K. C. Fan Ming-Tat Ko, and Yi-Ping Hung, " Image registration using a new edge based approach" , Computer Vision Graphics and Image Processing, vol. 67, no. 2, pp. 112-130, August 1997

[9] Josien P. W. Pluim, J. B. Antoine Maintz, and Max A. Viergever, " Interpolation Artefacts in Mutual Information-Based Image Registration" , Computer Vision and Image Understanding 77, 211–232 (2000)

[10] J. West, M. Fitzpatrick, and et al., " Comparison and evaluation of retrospective intermodality brain image registration techniques" , Journal of Computer Assisted Tomography, 21:554-566, July/August 1997

[11] K. Rohr, R. Sprengel and H. S. Stiehl, " Incorporation of Landmark Error Ellipsoids for Image Registration Based on Approximating Thin-Plate Splines" , Proc. Computer Assisted Radiology and Surgery (CAR'97), Berlin, Germany, June 25-28, 1997

[12] Lisa Gottesfeld Brown, " A Survey of Image Registration Techniques" , ACM Computing Surveys, Vol. 24, No. 4, Dec. 1992

[13] L. M. G. Fonseca and C.S. Kenney, " Control Point Assessment for Image Registration" , XII Brazilian Symposium of Computer Graphic and Image Processing, pp.125-132, Oct. 1999

[14] M.E. Alexander, " A Fast Hierarchical Non-Iterative Registration Algorithm" , International Journal of Imaging Systems and Technology, Vol. 10, pp. 242-257 (1999)

[15] Raj Sharman, John M. Tyler and Oleg S. Pianykh, " A Fast and Accurate Method to Register Medical Images using Wavelet Modulus Maxima" , Pattern Recognition Letter 21 (2000) 447-462

[16] Stefan Kruger and Andrew Calway, " Image Registration using Multiresolution Frequency Domain Correlation" , British Machine Vision Conf., pp. 316-325, Sept. 1998

[17] W. Peckar, C. Schnorr, K. Rohr, and H. S. Stiehl, " Two-Step Parameter-Free Elastic Image Registration with Prescribed Point Displacements" , 9th Int. Conf. on Image Analysis and Processing, ICIAP'97, 1310, pp. 527- 534, 1997

[18] Xiaojing Yuan, Jian Zhang and Bill Buckles, " Multi-scale Feature Identification Using Evolution Strategies" , Submitted to International Journal on Artificial Intelligence Tools

[19] Jian Zhang, Xiaojing Yuan, Zhixiang Zeng, Bill P. Buckles, Cris Koutsougeras and Saud Amer. "Niching in an ES/EP Context", Proceedings of Congress on Evolutionary Computation, Washington D.C., July 1999.

[20] Qinfen Zheng and Rama Chellappa, " A Computational Vision Approach to Image Registration" , IEEE Transaction on Image Processing, v2, n3, July 1993

[21] H. P. Schwefel. " Collective Phenomena in Evolutionary Systems" , Preprints of the 31th annual Meeting of the International Society for General System Research, Budapest, V2, June 1987

[22] H. P. Schwefel. " Evolution and Optimum Seeking" , Wiley Interscience, John Wiley & Sons, New York, 1995.