GENETIC ALGORITHMS
Poster Papers

Keith Mathias, chair

Evolution of adaptive discretization intervals for rule-based genetic learning system

Jaume Bacardit
Enginyeria i Arquitectura La Salle (URL)
Psg. Bonanova 8, 08022-Barcelona,
Catalonia, Spain, Europe.
jbacardit@salleURL.edu

Josep M. Garrell
Enginyeria i Arquitectura La Salle (URL)
Psg. Bonanova 8, 08022-Barcelona,
Catalonia, Spain, Europe.
josepmg@salleURL.edu

Abstract

The traditional classifier rules evolved in genetic based machine learning (GBML) systems need a discretization process to handle problems with real-valued attributes. A good discretization procedure is needed to generate a solution with good accuracy because the alternative of a high number of simple uniform-width intervals is bad due to the big search space being hardly explorable in a reasonable time. There exist some good discretization algorithms, like the Fayyad & Irani method [Fayyad and Irani, 1993], but they fail in some problems.

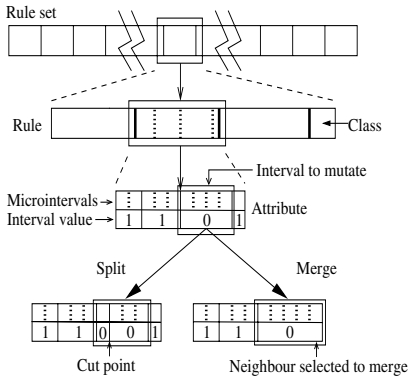
The work being summarized here deals with a rule representation with adaptive discrete intervals which split or merge through the evolution process, finding a robust and correct discretization intervals as the learning process is done with a reasonable computational cost.

Summary

This representation is used inside a Pittsburgh approach genetic classifier system derived from GABIL [DeJong et al., 1993], and the rule structure is taken directly from GABIL, using Conjunctive Normal Form (CNF) predicates.

In order to get a reasonable computational cost and also to bound the growth of the search space, some constrains have been introduced: (1) We define a fixed number of "low level" intervals which we call *micro-intervals*. (2) The adaptive intervals are built merging *micro-intervals*. When we split an interval, we select a random point in its *micro-intervals* to do the process. (3) When we merge two intervals, the value of the resulting interval is taken from the one which has more *micro-intervals*. (4) Finally, if both have the same number, the value is chosen randomly. The representations and split and merge operations are represented in figure 1

Figure 1: Adaptive intervals manipulated by merge and split operators.



We have integrated the split and merge operators inside the mutation operator, as it is the easiest part of the GA cycle to modify. Thus, we redefine the mutation operator adding p_{split} and p_{merge} as the probabilities of splitting or merging an interval which has been selected to mutate, instead of the classic bit-inversion operator.

The method presents a good performance and robustness.

References

[DeJong et al., 1993] DeJong, K. A., Spears, W. M., and Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13(2/3):161–188.

[Fayyad and Irani, 1993] Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029.

Influences of Clustering modifications on the performance of the Genetic Algorithm driven Clustering algorithm

Dirk Devogelaere, Marcel Rijckaert

K.U.Leuven, Chemical Engineering Department
De Croylaan 46, B-3001 Leuven,Belgium
Email: {dirk.devogelaere}@cit.kuleuven.ac.be
Tel: +16 32 23 68

SUMMARY

One way to look at basic modeling approaches is to split them up into mechanistic and data based models. A few years ago we developed our own data based model approach [1], called **G**enetic **A**lgorithm **d**riven **C**lustering (GAdC). The proposed methodology relies on semi-supervised clustering with a generative floating-point genetic algorithm and local learning. In this contribution we investigate the influence of clustering modification on the performance of the prediction of a real world application [2]. The task is the prediction of alga frequency distributions on the basis of the measured concentrations of the chemical substances, the global information concerning the season when the sample was taken, the river size and its flow velocity.

We deal with an evolutionary algorithm (EA) by implementing the GAdC as a generative floating-point genetic algorithm. An EA acts on a set of individuals. An individual is a representation of a point within the search space of the EA. In its simplest form, this individual is represented as a one-dimensional string of variables, called a chromosome. Each chromosome of the EA represents the coordinates of the cluster centers and a scaling factor for each dimension. If the dimensionality of the data is D, and there are K cluster centers, there will be D*K genes for the cluster centers and D genes for the scaling factors. The chromosome can be evaluated. This means that a certain fitness value (based on the objective value) is assigned to the individual depending on the problem at hand. In our case, the chromosome is decoded into a solution of the clustering. This solution is evaluated ("goodness of prediction") and the value is assigned to the chromosome in the EA.

The research in this paper is focused on how the performance of prediction is influenced by choosing a representative for the cluster centers. In all the variants, the genetic algorithm (GA) determines the cluster centers. By replacing the value determined by the GA in the non-empty clusters, we influence the mapping realized between the search space and the solution space of the GA. Different cluster centers in the search space might resolve to the same distribution of the cases over the different clusters, resulting in the same solution in the solution space. The EA is not aware of the similarity of these individuals. To overcome redundancy in the individuals' space, the EA has to update the offspring . Two variants for replacing the value proposed by the GA for the cluster centers were implemented. In the first variant (centroid), we replace the GA value by the mean value of the cluster calculated based on the elements in the cluster. In the second variant (closest) we replace the cluster center by the closest element of the cluster to the GA value. The variant without replacement is called "standard".

For each variant we predicted the outcome of all the seven algae distributions 30 times. For each algae distribution the mean squared error on the test set and the standard deviation were calculated depending on the variant used to update the cluster centers. Contrary to what was expected neither the centroid nor the closest variant performs better on the test set in general. Another way to present the results is plotting the error on the test set versus the ultimate fitness value obtained during training. There is a general tendency that training stops at lower fitness values in the case of the closest variant, while training stops at the higher fitness values in the case of the standard variant. As a consequence of the way of mapping the two variants "closest" and "centroid" cover a subspace of the solution space of the "standard" variant. This might indicate that less generations of the GA are necessary to achieve the same error on the test set. Secondly it might be important, as it is in training of neural networks, that training should be stopped at the right moment. If not, a kind of over training occurs and worse results on the test set are obtained. The results of a preliminary run indicate that the test error indeed seems to decrease but after obtaining a minimum at about 250 generations the error smoothly increases again as a function of the number of generations.

References

[1] D. Devogelaere, P. Van Bael, and M. Rijckaert (1999). Regression Through Genetic Algorithm driven Clustering, *Proceedings of the European Conference on Intelligent Techniques and Soft Computing (EUFIT),* September 7-10, Aachen, Germany.
[2] URL (1999) ftp.mitgmbh.de/pub/problem.zip

PRESERVING DIVERSITY IN CHANGING ENVIRONMENTS THROUGH DIPLOIDY WITH ADAPTIVE DOMINANCE

A. Şima ETANER-UYAR, A. Emre HARMANCI
Istanbul Technical University
Computer Engineering Department
TR80626 Maslak Istanbul, Turkey
{etaner, harmanci}@cs.itu.edu.tr

Genetic algorithms have been applied to a diverse field of problems with promising results. While most of these mainly address stationary problems, there's another group where the problem is dynamic, represented by a changing fitness function. These class of problems are characterized mainly by a need for a mechanism to adapt to the change. The best approach depends on the nature of the change in the fitness function. Different characteristics of changing fitness functions can be exploited in different ways to obtain an optimal solution. In dynamic environments, diversity plays an important role in genetic algorithm performance. The main approaches in coping with changing environments and preserving diversity are summarized in [Branke, 99]. One of these is the use of diploidy, however as shown in [Lewis, 98] diploidy alone is not sufficient and other modifications are needed. In this study a diploid representation for the individuals is used. Each individual consists of two chromosome strings, a string to represent the phenotype, a fitness value and an age value showing for how many generations the individual has survived. When determining the phenotype, a dominance map is applied to the two chromosome strings. In this implementation, a domination array composed of real numbers in $[0.0, 1.0]$, where each value shows the dominance factor of the allele 1 over the allele 0 for the corresponding location on the chromosomes, is used. The domination array evolves along with the individuals through the generations. The reproduction phase consists of mating pool determination, meiotic cell division with crossing over for gamete formation, mutation and the actual mating phase to form two new individuals. Offspring do not replace the parents and since population size is constant, individuals to survive into the next generation are determined by way of a fitness proportional method at the end of each generation. A possible replacement of aged individuals occurs at this stage where the aged individual may get replaced by a randomly created individual.

The main features of this algorithm contributing to preserving diversity are the use of a diploid representation with an adaptive domination mechanism, the use of a meiotic cell division in the reproduction phase and the use of a possible replacement policy of aged individuals. The algorithm used in this study is explained in greater detail in [Uyar, 99] by the same authors. To see the effects of each feature seperately, the algorithm is run with each feature either turned on or off. The results are obtained based on a variation of the 0-1 knapsack problem where the desired sum takes on random values at random intervals. The progression of the population diversity is observed through plotting the genotypic and phenotypic convergence rates of the population and tracking the algorithm's performance in following the change. It is shown in this study that different features of the chosen diploid algorithm address the issue of diversity from different aspects. The choice of which feature to use and which not to use depends mainly on defining the basic nature of the change in the environment and determining what each class of problem requires. At the time of submission, this is still a work in progress but the results obtained are promising.

References

[Branke, 99] BRANKE J. "Evolutionary Algorithms for Dynamic Optimization Problems - A Survey". Forschungsbericht No. 387, Institut fuer Angewandte Informatik und Formale Beschreibungsverfahren, Universitaet Karlsruhe. February 1999.

[Lewis, 98] LEWIS J., HART E., RITCHIE G. "A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems". PPSN'98, No. 1498 in LNCS. pp. 139-148. Springer 1998.

[Uyar, 99] UYAR S., HARMANCI E. "Investigation of New Operators for a Diploid Genetic Algorithm" in Proceedings of SPIE, Vol. 3812, pp. 32-43, 1999.

Genetic Algorithm Wrappers for Feature Subset Selection in Supervised Inductive Learning

William H. Hsu **Cecil P. Schmidt** **James A. Louis**

Laboratory for Knowledge Discovery in Databases, Kansas State University
234 Nichols Hall, Manhattan, KS 66506
{bhsu | cps4444 | jal8334}@cis.ksu.edu http://www.kddresearch.org

We derive a validation-based genetic algorithm for feature selection in supervised inductive learning, based upon the following loss functions:

1. **Inferential loss**: Quality of the model produced by an inducer as detected through inferential loss evaluated over a holdout validation data set $D_{val} \equiv D \setminus D_{train}$
2. **Model loss**: "Size" of the model under a specified coding or representation
3. **Ordering loss**: Inference/classification-independent and model-independent measure of data quality given only training and validation data D and hyperparameters $\boldsymbol{\alpha}$

$$f\left(\boldsymbol{\alpha}, D, \vec{\mathbf{I}}_{\mathbf{e}}\right) = a \cdot f_a\left(\boldsymbol{\alpha}, D, \vec{\mathbf{I}}_{\mathbf{e}}\right) + b \cdot f_b\left(\boldsymbol{\alpha}, D\right) + c \cdot f_c\left(\boldsymbol{\alpha}, D\right) \quad (1)$$

$$f_a^{BN}\left(\boldsymbol{\alpha}, D, \vec{\mathbf{I}}_{\mathbf{e}}\right) = 1 - \sqrt{\frac{1}{\sum_{X_i \in \mathbf{X} \setminus \mathbf{E}} a_i} \sum_{X_i \in \mathbf{X} \setminus \mathbf{E}} \sum_{j=1}^{a_i} \left(P'\left(x_{ij}\right) - P\left(x_{ij}\right)\right)^2} \quad (2)$$

$$f_a^{DT}\left(\boldsymbol{\alpha}, D\right) = 1 - \frac{m_{correct}}{m_{val}} \quad (3)$$

$$\textbf{where } m_{correct} \equiv h.classification - accuracy\left(D_{val}.select\left(\boldsymbol{\alpha}\right)\right)$$
$$h \equiv h_0.train\left(D_{train}.select\left(\boldsymbol{\alpha}\right)\right)$$
$$m_{val} \equiv \left|D_{val}\right|$$

$$f_b^{BN}\left(\boldsymbol{\alpha}, D\right) = 1 - \frac{\sum_{i=1}^{n} \left(a_i \cdot max\left(\prod_{j \in \mathbf{Pa}_{x_i}} a_j, 1 \right) \right)}{\prod_{i=1}^{n} a_i} \quad (4)$$

$$\textbf{where } a_i \equiv arity\left(X_i, B = \left(\chi, E, \Theta\right)\right)$$
$$\left(E, \Theta\right) = K2\left(\boldsymbol{\alpha}, D_{train}\right)$$

$$f_b^{DT}\left(\boldsymbol{\alpha}, D\right) = 1 - \frac{h.size(\,)}{s_{max}} \quad e.g., s_{max} = m$$

$$f_c^{DT}\left(\boldsymbol{\alpha}\right) = 1 - \frac{\left|\boldsymbol{\alpha}\right|}{n} \quad (5)$$

$$a + b + c = 1 \quad (6)$$

(7)

In related work on genetic wrappers for variable selection in supervised inductive learning, we adapted Equation (3) [HWRC02] from similar fitness functions developed by Cherkauer and Shavlik for decision tree pre-pruning and by Guerra-Salcedo and Whitley for connectionist learning [GW99]. This breadth of applicability demonstrates the generality of simple genetic algorithms as wrappers for performance tuning in supervised inductive learning.

In experiments using the UC Irvine Machine Learning Database repository, this system is shown to be competitive with search-based feature selection wrappers.

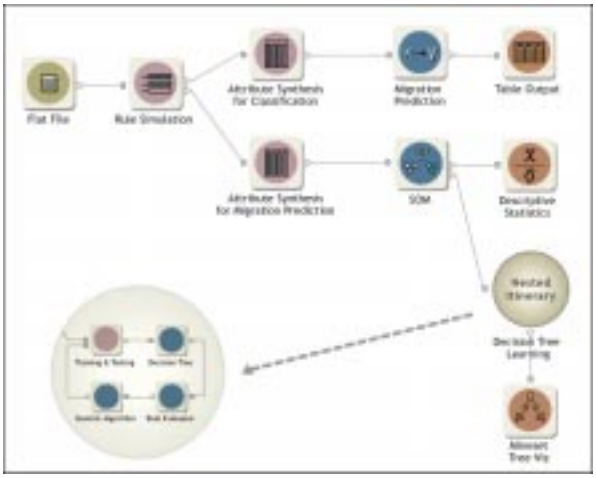


Figure 1. Itinerary for MLJ-CHC

Figure 1 illustrates a real-world application [HWRC02] – automobile insurance risk analysis – that uses the GA wrapper system (depicted in the lower-left inset). Preliminary results on this test bed also indicate that the system is competitive with search-based wrappers.

References

[Be90] D. P. Benjamin, editor. *Change of Representation and Inductive Bias*. Kluwer Academic Publishers, Boston, 1990.

[GW99] C. Guerra-Salcedo and D. Whitley. Genetic Approach to Feature Selection for Ensemble Creation. In *Proceedings of the 1999 International Conference on Genetic and Evolutionary Computation (GECCO-99)*. Morgan-Kaufmann, San Mateo, CA, 1999.

[HWRC02] W. H. Hsu, M. Welge, T. Redman, and D. Clutter. Constructive Induction Wrappers in High-Performance Commercial Data Mining and Decision Support Systems. *Knowledge Discovery and Data Mining*, Kluwer, 2002.

A Markov Chain Analysis of Fitness Proportional Mate Selection Schemes in Genetic Algorithm

Chien-Feng Huang
Center for the Study of Complex Systems, 4477 Randall Lab.
University of Michigan, Ann Arbor, MI 48109
cfhuang@fiore.physics.lsa.umich.edu

In the research of Genetic Algorithms (GAs), many models focus on problems where each individual's fitness is independent of others. In (Huang, 2002a), simple models that implement mate selection in GAs were introduced to model interdependent fitnesses of population members. They have been studied by the Schema Theorem and some empirical results. In this paper, I conduct a Markov chain analysis to further investigate these models, based on the Markov model developed by Nix and Vose (1992). Although such models quickly become unwieldy with increasing population size or string length, they provide important insights into how mate selection plays a crucial role in GA's search power, and thus serve as guidelines for studying more realistic problems.

In (Huang, 2002a), the implementation of the four mate selection schemes in the selection-for-mating step of a simple GA is:

During each mating event, a fitness-proportionate selection is run to pick out the first individual. Then the Hamming distances of all population members to the first individual are calculated. The actual mate of the first individual is chosen according to the following four different schemes:

Maximum Similar Mating (MSM): The population member whose Hamming distance is the smallest is selected for mating.

Proportional Similar Mating (PSM): The probabilities of population members being selected are *reversely* proportional to their Hamming distances.

Proportional Dissimilar Mating (PDM): The probabilities of population members being selected are proportional to their Hamming distances.

Maximum Dissimilar Mating (MDM): The population member whose Hamming distance is the largest is selected for mating.

As indicated in (Huang, 2002b), the way of calculating the probability of the second individual being selected as a parent is the only part that needs to be reconsidered in the Nix and Vose model. Please see (Huang, 2002b) for the detailed derivation for these four mate selection schemes.

With the modified Markov models, I use the GAFO (GAs being used for function optimization) idea developed by De Jong et al. (1994) to examine the effects of different mate selection schemes on GA's performance. As an example, I use the test function $f(y) = integer(y) + 1$. Due to the computational limitation, I use the simplest possible case, string length 2, to proceed the investigation.

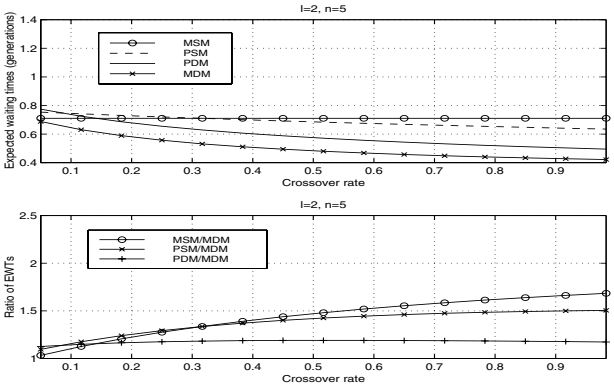


Figure 1: Interacting effects of mate selection and crossover.

Figure 1 shows the results obtained, based on crossover rates ranging from 0.05 to 1, mutation rate .1, and population size 5. The top plot is for the exact expected waiting times (EWTs) to the optimum and the bottom plot shows the ratios of the EWTs for the MSM, PSM, and PDM GAs to that for the MDM GA. One can see that MDM generally has the least EWTs than the other three. In particular, the dissimilar mating schemes demonstrate increasingly improved performance as crossover rate increases. This shows that both proper mate selection and crossover must operate together to enhance the power of simulating information exchange in GA's population.

Reference

Huang, C.-F. (2002a). A Study of Fitness Proportional Mate Selection Schemes in Genetic Algorithms. *Technical Report CSCS-2002-002*. Ann Arbor, MI: University of Michigan, Electrical Engineering and Computer Science.

Huang, C.-F. (2002b). A Markov Chain Analysis of Fitness Proportional Mate Selection Schemes in Genetic Algorithm *Technical Report CSCS-2002-003*. Ann Arbor, MI: University of Michigan, Electrical Engineering and Computer Science.

Nix, A. E. and Vose, M. D. (1992). Modelling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence #5*, pp. 79-88.

A Study of Fitness Proportional Mate Selection Schemes in Genetic Algorithms

Chien-Feng Huang
Center for the Study of Complex Systems, 4477 Randall Lab.
University of Michigan, Ann Arbor, MI 48109
cfhuang@fiore.physics.lsa.umich.edu

In the research of Genetic Algorithms (GAs), many models focus on problems where each individual's fitness is independent of others. In this paper, I introduce simple models to study mate selection in the context of GA. Allowing individuals to search for mates is an approach to model interdependent fitnesses of population members. This introduces another source of selection pressure, and the resulting GA hence forms a more complex system in which individuals' fitnesses depend on both the environment and other population members. I will show that mate selection plays a crucial role in GA's search power.

In this paper, four mate selection schemes in the selection-for-mating step of a simple GA are proposed as follows:

During each mating event, a fitness-proportionate selection is run to pick out the first individual. Then the Hamming distances of all population members to the first individual are calculated. The actual mate of the first individual is chosen according to the following four different schemes:

Maximum Similar Mating (MSM): The population member whose Hamming distance is the smallest is selected for mating.

Proportional Similar Mating (PSM): The probabilities of population members being selected are *reversely* proportional to their Hamming distances.

Proportional Dissimilar Mating (PDM): The probabilities of population members being selected are proportional to their Hamming distances.

Maximum Dissimilar Mating (MDM): The population member whose Hamming distance is the largest is selected for mating.

The testbed employed is an incompatible small royal road function IS_1 as shown in Table 1.

Table 1: Incompatible small royal road function IS_1.

s_1 = 11111**********; c_1 =10
s_2 = 00000**********; c_2 = 5
s_3 = *****11111*****; c_3 =10
s_4 = *****00000*****; c_4 = 5
s_5 = **********11111; c_5 =10
s_6 = **********00000; c_6 = 5
s_7 = **********11111; c_7 =10
s_8 = **********00000; c_8 = 5

This function involves a set of schemata $S = \{s_1, \ldots, s_8\}$ and the fitness of a bit string x is defined as

$$F(x) = \sum_{s \in S} c_s \sigma_s(x),$$

where each c_s is a value assigned to the schema s as defined in the table; $\sigma_s(x)$ is defined as 1 if x is an instance of s and 0 otherwise. In this function, the fitness of the global optimum string (20 1's) is $10 \times 4 = 40$.

The experiments performed here are based on one-point crossover rate 1, mutation rate 0.005, and population size 20 over 50 runs. Figure 1 shows the averaged best-so-far curves on function IS_1 for the four mate selection strategies.[1]

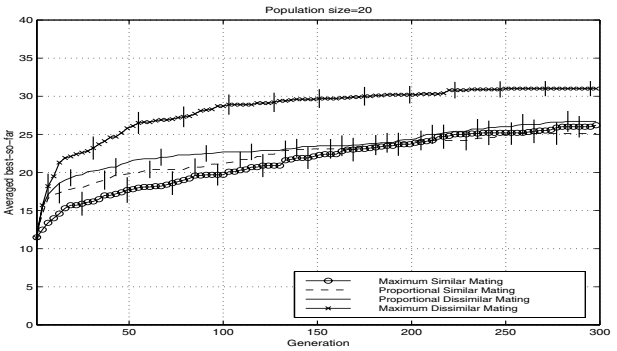


Figure 1: Best-so-far performance on IS_1.

We can see the maximum dissimilar mating results in better improvement than the other three. In (Huang, 2002) it is shown that, by suppressing hitchhiking and the founder effect, the maximum dissimilar mating retains more genetic variation in the population. The further exploration of the search space for the GA thus yields a better best-so-far performance.

Reference

Huang, C-F. (2002). *A Study of Mate Selection in Genetic Algorithms.* Doctoral dissertation. Ann Arbor, MI: University of Michigan, Electrical Engineering and Computer Science.

[1] The vertical bars overlaying the metric curves represent the 95-percent confidence intervals.

Incorporation of Fuzzy Preferences into Evolutionary Multiobjective Optimization

Yaochu Jin and Bernhard Sendhoff
Future Technology Research, Honda R&D Europe (D)
Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany
Email: {yaochu.jin, bernhard.sendhoff}@de.hrdeu.com

Abstract

A method for incorporating fuzzy preferences into evolutionary multiobjective optimization is proposed. Fuzzy preferences are converted into interval-based weights instead of single-valued crisp weights. The weight intervals are combined with the evolutionary dynamic weighted aggregation to obtain the preferred Pareto-optimal solutions.

Finding all Pareto-optimal solutions is not the final goal: a decision has to be made based-on users' preferences. Such preferences can usually be represented with the help of fuzzy logic. Before fuzzy preferences can be incorporated into multiobjective optimization, fuzzy preferences in inform of fuzzy relations are converted to a set of single-valued weights [1], during which a lot of information is lost. One more natural way to do this is to convert the fuzzy preferences into weights in intervals.

The weight intervals can then be combined with random weighted method (RWA) and dynamic weighted method (DWA) [2]. Suppose the maximal and minimal value of a weight is $\bar{w}$ and $\underline{w}$, the weights can be changed as follows:

$$w_1^i = \underline{w}_1 + (\bar{w}_1 - \underline{w}_1)\ \mathrm{rdm}(P)/P, \text{ (RWA) (1)}$$
$$w_1(t) = \underline{w}_1 + (\bar{w}_1 - \underline{w}_1)\ \sin(2\pi t/F)| \text{ (DWA).(2)}$$

In this way, the evolutionary algorithm can achieve a set of Pareto solutions reflected by the fuzzy preferences.

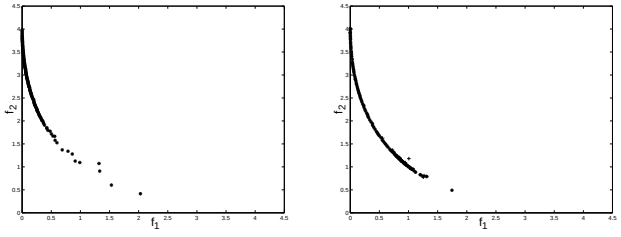
To illustrate how this method works, one example on two-objective optimization is presented in the following. In the simulations, we consider the following fuzzy preferences: "Objective 1 is more important than objective 2".

Then we can get the following preference matrix:

$$P = \begin{bmatrix} 0.5 & \delta \\ \gamma & 0.5 \end{bmatrix}, \quad (3)$$

where $0.5 < \delta < 1$ and $0 < \gamma < 0.5$. Therefore, we have $0.5 \leq w_1 \leq 1.0$, and $0.0 \leq w_2 \leq 0.5$.

Simulation are carried out on the Schaffer's function and the results are shown in Fig. 1



(a) (b)

Figure 1: Results on the Schaffer's function. a) RWA, b) DWA.

The main idea is to convert the fuzzy preferences into interval-based weights. With the help of the dynamic weighted aggregation method, preferred Pareto-optimal solutions can be obtained.

References

[1] D. Cvetkovic and I. Parmee. Use of preferences for GA-based multi-objective optimization. In *Proceedings of 1999 Genetic and Evolutionary Computation Conference*, pages 1504–1510, San Francisco, California, 1999. Morgan Kaufmann.

[2] Y. Jin, T. Okabe, and B. Sendhoff. Adapting weighted aggregation for multi-objective evolution strategies. In *Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization*, Leture Notes in Computer Science, pages 96–110, Zurich, March 2001. Springer.

Multi-Objective Bayesian Optimization Algorithm

Nazan Khan, David E. Goldberg, Martin Pelikan
Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{nkhan1, deg, mpelikan}@uiuc.edu

Recently, significant development in the theory and design of competent genetic algorithms (GAs) has been achieved. By competent GA we mean genetic algorithms that can solve boundedly difficult problems quickly, accurately, and reliably. However, most of the existing competent GAs focus only on single-objective optimization although many real-world problems contain more than one objective. Independently of the development of competent genetic algorithms, a number of approaches to solve such multiobjective problems have been proposed. However, there has been little or no effort to develop competent multiobjective operators that efficiently identify, propagate, and combine important partial solutions of the problem at hand.

This study makes an effort towards multiobjective competency by combining the best of both the worlds. Specifically, the study combines competent GAs with advanced techniques for finding and maintaining a diverse set of nondominated solutions defining the Pareto front. In particular, we combine the Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1999) with the nondominated sorting GA (NSGA-II) (Deb, Pratap, & Meyarivan, 2000). The resulting multi-objective Bayesian optimization algorithm (mBOA) incorporates the selection method of the NSGA-II into BOA.

The mBOA methodology can be described as follows: (1) Randomly generate n solutions and perform selection, (2) Build a probabilistic model (Bayesian network) of the promising solutions, (3) Sample new solutions using the Bayesian network, (4) Combine both the parent and the offspring population (5) Perform a non-dominated sorting and compute the crowding distance of the combined population. (6) Select the n *best* (based on the rank and crowding distance) solutions, and (7) Go to step (2) and repeat the process till some convergence criteria are satisfied. Further details on the proposed algorithm are given elsewhere (Khan, Goldberg, & Pelikan, 2002).

The proposed algorithm has been tested on an array of test functions which incorporate deception and loose-linkage and the results are compared to those of NSGA-II. Results indicate that mBOA outperforms NSGA-II on large loosely linked deceptive problems. A representative result on multiple interleaved trap-5 functions is shown in figure 1. It is clear from the figure that the mBOA converges to the actual front and maintains a good spread on it.

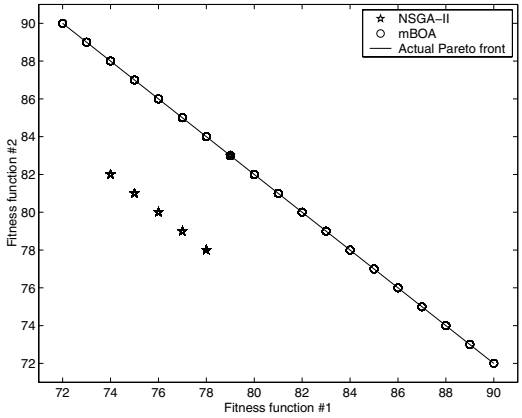


Figure 1: mBOA's performance on multiple interleaved trap-5 deceptive functions (Maximized)

References

Deb, K., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature*, *4*(1), 849–858.

Khan, N., Goldberg, D. E., & Pelikan, M. (2002). *Multi-objective Bayesian optimization algorithm* (IlliGAL Report No. 2002009). Urbana, IL: University of Illinois at Urbana-Champaign.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1*, 525–532.

A Hybrid Genetic Search for Circuit Bipartitioning

Jong-Pil Kim and Byung-Ro Moon
School of Computer Science & Engineering, Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{jpkim, moon}@soar.snu.ac.kr

Hypergraph partitioning is an important problem and has many applications including design automation of VLSI chips and multi-chip systems. In this paper, we proposed a hybrid genetic algorithm for partitioning a hypergraph into two disjoint graphs of minimum cut size.

The Fiduccia-Matheyses algorithm (FM) [1] is a representative iterative improvement algorithm for a hypergraph partitioning problem. But the quality of the FM is not stable. Kim and Moon [3] introduced lock gain as a primary measure for choosing the nodes to move. It uses the history of search more efficiently. Lock gain showed excellent performance for general graphs. We adapt the lock gain for hypergraphs within the framework of FM.

Lock gain was originally designed for the general graphs [3]. Thus, to apply the lock gain to the hypergraph bisection, some modification is necessary. We propose a new lock gain calculation method for the hypergraph bisection. Let's define $l_e(v)$ to be the lock gain of a node v due to the net e. $l_e(v)$ is obtained as the following. We assume that the node v is on the left side without loss of generality. L, R, L', and R' are defined to be #nodes on the left side, #nodes on the right side, #locked nodes on the left side, and #locked nodes on the right side, respectively.

If all nodes on the right side are locked and there is no locked node on the left side ($L > 0, \quad L' = 0, \quad R = R' > 0$), or if there exists locked nodes on the right side and there is no free node on the left side ($L = 1, \quad L' = 0, \quad R \geq R' > 0$), then $l_e(v) = 1$. On the other hand, if all nodes are on the left side and at least one node is locked ($L > L' > 0, \quad R = R' = 0$), or if all nodes on the left side except v are locked and there is no locked node on the right side ($L - L' = 1, \quad R > 0, \quad R' = 0$), then $l_e(v) = -1$. The lock gain $l(v)$ of the node v is defined to be $l(v) = \sum_{e \in N(v)} l_e(v)$ where $N(v)$ is a set of nets to which the node v is connected.

Table 1: Bipartition Cut Sizes

Circuits	GA		hMetis150	
	Average[1]	CPU[2]	Average[3]	CPU[2]
Test02	**88.18**	13.58	89.93	10.81
Test03	**58.00**	4.35	59.96	10.31
Test04	**51.28**	8.48	54.85	8.82
Test05	71.88	27.84	**71.44**	17.23
Test06	**63.04**	5.73	64.16	10.08
Prim1	53.66	1.33	**53.02**	5.28
Prim2	**148.87**	24.12	178.39	30.70

1. The average cut size of 100 runs
2. CPU seconds on Pentium III 866MHz
3. The average cut size of 100 runs, each of which is the best of 150 runs of hMetis

We tested the proposed algorithm on 7 ACM/SIGDA benchmarks. We examine the performance of the hybrid GA which uses the lock-gain based FM as a local optimization engine against a well-known partitioner hMetis [2]. Because the GA took roughly 150 times more than a single run of hMetis, it is not clear how critical the genetic search is to the performance improvement. Thus, hMetis150, that is a multi-start version of hMetis with 150 runs, was compared.

Table 1 shows the performance of the GA. On the average, the proposed GA performed best in 5 graphs among 7.

References

[1] C. Fiduccia and R. Mattheyses. A linear time heuristics for improving network partitions. In *19th IEEE/ACM Design Automation Conference*, pages 175–181, 1982.

[2] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proc. Design Automation Conference*, pages 526–529, 1997.

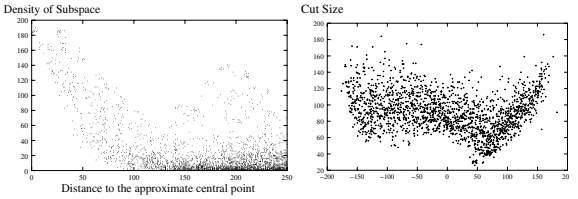
[3] Y. H. Kim and B. R. Moon. A hybrid genetic search for graph partitioning based on lock gain. In *Genetic and Evolutionary Computation Conference*, pages 167–174, 2000.

This work was supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University(SNU) and Brain Korea 21 Project. The RIACT at SNU provided research facilities for this study.

Visualization of the Fitness Landscape, a Steady-State Genetic Search, and Schema Traces

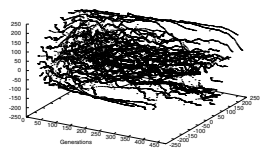
Yong-Hyuk Kim and Byung-Ro Moon
School of Computer Science & Engineering, Seoul National University
Shillim-dong, Kwanak-gu, Seoul, 151-742 Korea
{yhdfly, moon}@soar.snu.ac.kr

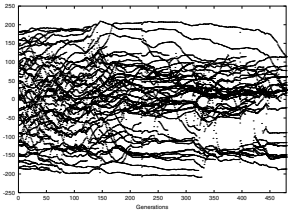
An NP-hard problem such as graph partitioning has a finite solution set and each solution has a cost. Although finite, the problem space is intractably large even for a small but nontrivial problem. A number of studies about the ruggedness and the properties of problem search spaces were done. Visualization is one of the most basic tools for studies of search spaces. A notable method for fitness landscapes is the plotting of fitness-distance correlation [1]. For genetic algorithm (GA) visualization, the most popular method is the fitness flow over time as in many GA papers. In this paper, we propose new visualization techniques primarily using Sammon's mapping. We analyze the problem space for graph partitioning more elaborately. We visualize the solutions associated with the genetic search. We also trace schemata.

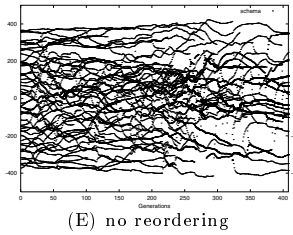


(A) local-optimum space

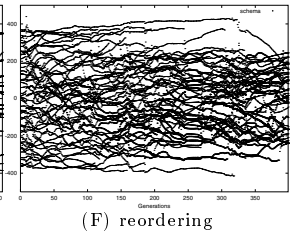
(B) local-optimum space


(C) search process: 3D


(D) search process: 2D



(E) no reordering



(F) reordering

We agree with the conjecture of Boese *et al.* [1] about the global convexity of local-optimum space but it is difficult to obtain further deduction. Figure (A) shows that the density of local optima near the center of the problem space is remarkably high. Interesting enough, one can also observe fairly high-density areas far from the center. It suggests the existence of "medium valleys."

Sammon's mapping [3] is a mapping technique for transforming a dataset from a high-dimensional input space onto a low-dimensional output space. The basic idea is to arrange all the data points on output space in such a way that minimizes the distortion of the relationship among data points. Sammon's mapping is a good visualization tool for the multi-dimensional dataset. The local-optimum space is a good candidate for Sammon's mapping. Figure (B) shows Sammon's mapping of the local-optimum space. The result visualizes the existence of valleys in more than one place.

Recently, Dybowski *et al.* [2] proposed a GA visualization method using Sammon's mapping. We extend their works. We make experiments with different distance measures, visualize the population space in 3-dimensional space, and provide a new technique for a steady-state GA to visualize the whole genetic search process. More detailed description is omitted by space limit. In the final experiments, we visualize the whole genetic search process of a steady-state GA. which generates only one offspring per iteration. It does not make fast change in population space. Hence, if previous positions are used for the initial positions of next generation Sammon's mapping, the positions change steadily over the generations. This makes it possible to visualize solutions over the genetic search process. Figure (C) and (D) show the visualization of the genetic search process.

It is important to preserve valuable schemata. The reordering heuristic transforms the shapes of valuable schemata to those advantageous for survival. We provide experiments for visualization of high-quality schema traces. Through the visualization combined with the techniques mentioned above, we observe the creation and extinction of schemata (see Figure (E) and (F)). Without reordering, despite of its early appearance, the schema did not spread all over the population as fast as the reordered version. On the other hand, the reordered version showed fairly stable preservation of the schema.

Our approach goes beyond those of Boese *et al.* [1] and Dybowski *et al.* [2]. Our approach will be also useful for other optimization problems.

This work was supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University (SNU) and Brain Korea 21 Project. The RIACT at SNU provided research facilities for this study.

References

[1] K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 15:101–113, 1994.

[2] R. Dybowski, T.D. Collins, and P. Weller. Visualization of binary string convergence by Sammon mapping. In *Fifth Annual Conference on Evolutionary Programming*, pages 377–383, 1996.

[3] J. W. Sammon, Jr. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.

Memetic Algorithms for Combinatorial Optimization Problems in the Calibration of Modern Combustion Engines

K. Knödler, J. Poland, and A. Zell
University of Tübingen,
Department of Computer Science
Sand 1, D - 72076 Tübingen, Germany
knoedler@informatik.uni-tuebingen.de

A. Mitterer
BMW Group Munich,
80788 München, Germany
alexander.mitterer@bmw.de

This work focuses on the high relevance of *memetic algorithms* (MAs, see e.g. [1] or [2]) in the calibration of modern combustion engines[1]. The MA used in this work is given by the following pseudo code:

```
begin
    for j := 1 to μ do
        i := Local-Search(generateSolution());
        add individual i to P;
    endfor;
    repeat
        for i := 1 to n_cross · μ
            select two parents i_a, i_b ∈ P randomly;
            i_c := Local-Search(Recombine(i_a, i_b));
            add individual i_c to P;
        endfor;
        for i := 1 to n_mut · μ
            select an individual i ∈ P randomly;
            i_m := Local-Search(Mutate(i));
            add individual i_m to P;
        end repeat;
        P := select(P);
        if P converged then P := mutateAndLS(P);
    until terminate=true;
end;
```

A random starting population is tackled by a local search operation, e.g. a *Monte Carlo Algorithm* (MC) in order to receive local optimum solutions. The numbers of individuals taken for crossover and for mutation are given by $n_{cross} \cdot \mu$ and $n_{mut} \cdot \mu$, respectively. After the convergence of the algorithm, all individuals but the best one are mutated and the MC is applied.

We studied three combinatorial optimization problems in the field of engine calibration. For transparency reasons, only the *final smoothing of maps defined by look-up tables* that was introduced in [4] is discussed here. Figure 1 visualizes a simple problem instance: there are three sets of candidates labeled with circles, diamonds, and squares. Every set itself defines a possible look-up table and hence a possible map. The idea is to mix up the sets in order to receive one final well defined look-up table with best smoothness properties. The mixing is noncritical, since the candidates at one grid point yield only slightly different engine behavior, e.g. fuel consumption or exhaust emission. We use

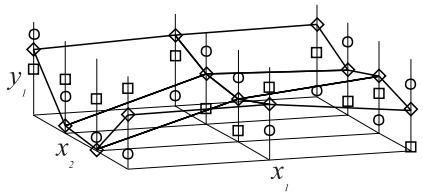


Figure 1: Candidates for a look-up table.

the variable alphabet encoding, i.e. each grid point j corresponds to one position of a chromosome that takes n_j values: $v = (v_j)_{j=1}^N \in \bigotimes_{j=1}^N \{1 \dots n_j\}$. Here, n_j is the number of candidates available at (x_1^j, x_2^j). This encoding allows standard mutation and crossover operations. We define the objective function $\phi(M) = \sum_{1 \le i < j \le n} neigh(i,j) \cdot |y^{(i)} - y^{(j)}|$ as smoothness of a map M where $neigh(i,j)$ is 1 for neighboring grid points, otherwise 0.

The consequent application of the MC (after each recombination and mutation) given by

```
repeat N times
    Choose j ∈ 1, ..., M randomly;
    Find all k ∈ 1, ..., n_j such that φ becomes minimal;
    Choose randomly one of these k;
end repeat;
```

significantly improves the regular genetic algorithm (GA). Beyond it, the MA outperforms a hybrid GA that uses the MC as mutation operation by up to 25%. Since the MC works locally it does not need complete fitness evaluations. However, to compensate higher computation times of the MA, the GAs were run with increased generation numbers and population sizes. In addition island parallel populations were used.

References

[1] P. Moscato, Memetic Algorithms: A Short Introduction, New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover, McGrawHill, London, 1999.

[2] P. Merz, Memetic Algorithms for Combinatorial Optimization Problems, PhD thesis, University of Siegen, 2000 (provided by the library)

[4] J. Poland et al., Evolutionary Search for Smooth Maps in Motor Control Unit Calibration, Stochastic Algorithms: Foundations and Applications (SAGA), 2001

[1] This research has been supported by the BMBF (grant no. 01 IB 805 A/1).

Using incremental evaluation and adaptive choice of operators in a genetic algorithm

Alexander Kosorukoff
Dept. of General Engineering
University of Illinois
Urbana-Champaign, IL 61801
alex3@illigal.ge.uiuc.edu

Abstract

The number of fitness evaluations determines the efficiency of a GA. This research suggests incremental evaluation which, if applicable, substantially reduces the complexity of evaluation of some individuals. Incremental evaluation and rational choice of operator based on utility maximization gives a two-fold reduction in computational cost needed to optimize the one-max function.

In this work, the problem "crossover vs mutation" is formulated as a decision problem using multiattribute utility theory (MAUT) (Keeney & Raiffa, 1993). The multilinear utility function of operators is based on ideas of Goldberg (1998). After assessing the utility function, MAUT is used to find a rational trade-off between crossover and mutation. The one-max function is the simplest function that can be evaluated incrementally and interesting results with it can motivate further research of more complex fitness functions.

Here we skip a detailed utility function assessment, and present only the final result:

$$U(F,I,S) = 0.06U(I) + 0.04U(S) +$$

$$+0.6U(F)U(I) + 0.3U(F)U(S)$$

where F is the number of applications per fixed cost, I is innovation rate, S is a scrap rate. According to this formula, utilities of crossover and mutation are $U_c \approx 0.09$ and $U_m \approx 0.36$. Apparently $U_c < U_m$, so the optimal strategy according to decision theory seems to always use mutation. However the experiments with only mutation produced inferior results. That is because utilities of operators change significantly during the run. However it is easy to recalculate the utilities in each generation. This gives us an adaptive algorithm that always selects an operator with maximumal estimated utility.

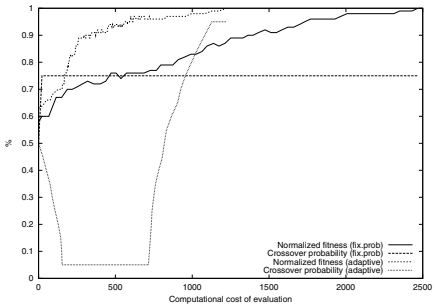


Figure 1: Comparison of adaptive sampling vs sampling with fixed probabilities

The utility of application of mutation operator is higher initially, however later the situation changes. It makes the adaptive algorithm change its preferred innovation operator (figure 1). Here we compare the average results: for fixed crossover probability $p_x = 0.75$ the average computational cost is 2568.30; for the adaptive algorithm, it is 1334.18.

This research proposes the rational choice of crossover and mutation. Genetic algorithm that chooses an operator with maximum utility has shown two-fold reduction in computational costs.

References

Goldberg, D. E. (1998). *The design of innovation: Lessons from genetic algorithms, lessons for the real world* (IlliGAL Report 98004). Department of General Engineering, UIUC.

Keeney, R. L., & Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value tradeoffs*. Cambridge, MA: Cambridge University Press.

A Hybrid Genetic Algorithm for Optimal Hexagonal Tortoise Problem

Seung-Kyu Lee, Dong-Il Seo, and Byung-Ro Moon
School of Computer Science & Engineering, Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{spin30, diseo, moon}@soar.snu.ac.kr

Hexagonal tortoise problem, which was first introduced by an old Korean mathematician Suk-Jung Choi (1646-1715) [1], is one of the most complicated tortoise problems. Since there has been no general heuristic to solve the hexagonal tortoise, an exhaustive search has been considered to be the only way for solving the problem to the optimum [4]. Recently a number of heuristics for the problem have been proposed [2], but they are too specific to be used for finding diverse instances.

The problem is to assign the consecutive numbers 1 through n to the vertices in a graph, which is composed of a number of overlapping hexagons, so that the sum of the numbers on each hexagon is the same. Figure 1 shows an optimal solution to the hexagonal tortoise problem with 30 numbers. The term "tortoise" comes from the fact that the overall shape of the graph resembles the *theca* of a turtle. We will take an optimization version of the problem that minimizes the difference among the hexagonal sums.

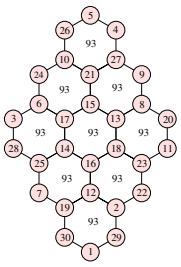


Figure 1: An optimal solution to the hexagonal tortoise with 30 numbers

We used a steady-state GA and encoded a solution as a matrix using hexagon-to-rectangle transformation

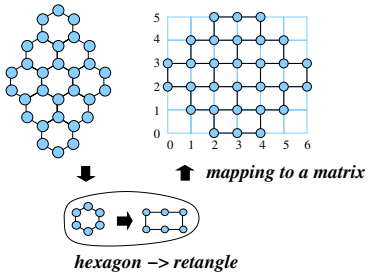


Figure 2: An example transformation

as shown in Figure 2. The fitness of a chromosome was given by the standard deviation of the hexagonal sums. The GA used the two-dimensional geographic crossover [3] and adopted an iterative improvement algorithm.

We attacked five problem instances with 30, 48, 70, 96, and 120 vertices, respectively. We denote by IGA the proposed GA with the iterative improvement. IGA found the optimal solutions up to the 70-number problems. Multi-Start performed much better than pure GA but was not comparable to IGA. The iterative improvement is a critical engine of both Multi-Start and IGA; but the performance was poor as its own. Pure GA showed the worst results among them.

References

[1] S. J. Choi. *Gusuryak* (a reprint). Sungshin Women's University Press, Seoul, Korea, 1983.

[2] Y. H. Jun. Mysteries of mathematics: The order of the universe hidden in numbers. *Dong-A Science*, 14(7):68–77, 1999.

[3] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *Proceedings of 6th International Conference on Genetic Algorithms*, pages 96–103, 1995.

[4] D. J. Kim and Y. H. Oh. Properties and solution-finding algorithm of Jisuguimundo (Turtle-shape Diagram). In *Proceedings of Korea Information Science Society*, volume 16, pages 405–408, 1989.

This work was supported by KOSEF through the Statistical Research Center for Complex Systems at Seoul National University(SNU) and Brain Korea 21 Project. The RIACT at SNU provided research facilities for this study.

Vehicle Routing Problem: Doing it the Evolutionary Way

Penousal Machado[1,2], Jorge Tavares[2], Francisco B. Pereira[1,2], Ernesto Costa[2]

[1]Instituto Superior de Engenharia de Coimbra, Quinta da Nora, 3030 Coimbra, Portugal
[2]Centro de Informática e Sistemas da Universidade de Coimbra, Polo II, 3030 Coimbra, Portugal
{machado, xico, ernesto}@dei.uc.pt, jast@student.dei.uc.pt

1 INTRODUCTION

The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem, which can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several costumer demands, find the set of routes with overall minimum route cost which service all the demands.

Due to the nature of the problem it is not viable to use exact approaches for large instances of the VRP. The application of evolutionary computation (EC) has achieved limited success. This led researchers to rely on hybrid approaches that combine the power of EC with the use of specific heuristics (e.g., (Thangiah, 1995)) or to simplify the problem (e.g., (Zhu, 2000)).

In this paper we present two EC approaches to the generic VRP. To our knowledge this is the first attempt to apply non-specific EC methods to this variant of the VRP. Our first approach uses a standard genetic algorithm (GA), whilst in the second we resort to a coevolutionary model.

2 GA APPROACH TO THE VRP

Our model can be viewed as an extension of the traditional GA approaches to the TSP problem.

We use a fixed size chromosome. Possible values for genes are: an integer representing a customer node or a special blank symbol. This blank symbol acts as a separator between routes. Since we don't know beforehand how many vehicles will be used in the optimal solution we set this number to the number of nodes divided by two.

As genetic operators we use: the partially mapped crossover (PMX) and the swap mutation operator.

Some routes in the chromosome may cause the vehicle to exceed its capacity. When this happens, we perform the following modification: the route that exceeds the vehicle capacity is split in several ones. These changes only occur at the interpretation level and, therefore, the information codified in the chromosome is not altered.

3 COEVOLUTIONARY APPROACH

We use two subpopulations: individuals from the 1st describe the size of each route; individuals from the 2nd specify the order by which the nodes are visited. Routes that exceed vehicle capacity are split.

Since we don't know in advance the optimal number of vehicles we the size of the individuals of the first subpopulation to the number of nodes divided by two. Maximum route length is also set to this value. As genetic operators, for that subpopulation, we use two-point crossover and uniform mutation.

The size of the individuals of the second subpopulation is fixed and equal to the number of nodes. Again, we use PMX crossover and swap mutation.

4 SYNOPSIS OF THE RESULTS

The results achieved are promising, as they show that EC techniques can deal in a satisfactory way with the problem. In particular, we showed that the inclusion of a simple, and non-specific, heuristic to generic EC techniques provides significant improvement of the results. The characteristics of our approach suggest that it shows good scalability, allowing their application to more complex instances of the problem, where more specific techniques may fail.

Acknowledgments

This work was partially financed by the Portuguese Ministry of Science and Technology under contract POSI/34493/SRI/2000.

References

Thangiah, S. R., Vehicle routing with time windows using Genetic Algorithms, Application and Book of Genetic Algorithms: New Frontiers, Volume II. Chambers, L. (ed), pp. 253-277, CRC Press, 1995.

Zhu, K., *A New Genetic Algorithm for VRPTW*, International Conference on Artificial Intelligence, Las Vegas, USA, 2000.

A Genetic Algorithm-Specific Test of Random Generator Quality

Mark M. Meysenburg, Dan Hoelting, Duane McElvain
Computer Science Dept.
Doane College
Crete, NE 68333

James A. Foster
Computer Science Department
University of Idaho
Moscow, ID USA 83844

Abstract

It has been shown in past research that pseudo-random number generator (PRNG) quality can impact the performance of simple genetic algorithms (GAs). However, standard empirical tests of random generator quality are not good predictors of when such impacts are likely to occur. In this paper, we introduce a new test of random generator quality, tailored to specific instances of a simple GA. This test has been shown to be a better predictor of GA performance impacts than standard empirical tests.

1 PRNG QUALITY AND GA PERFORMANCE

Past research has shown that the pseudo-random number generator (PRNG) chosen can impact the performance of evolutionary algorithms, and genetic algorithms (GAs) in particular. However, standard empirical PRNG quality evaluation tests are not able to fully predict when such impacts might occur.

In recent experiments, we evaluated PRNGs of varying quality levels using the Diehard test suite [Marsaglia, 1993]. We then used these PRNGs to drive a simple GA over a test suite of 42 problems [Meysenburg et al., 2002]. We found that Diehard gave false positive results: the suite predicted that the RANDU PRNG would cause unusual GA performance, but in fact such performance differences were not detected.

To correct this problem, we developed a PRNG quality test tailored specifically to the way our GA uses randomness. This test considers every decision made by the PRNG during a GA run, except for population initialization. The test uses a PRNG to make all of the decisions that would be made during an actual GA run, in the same order in which they would be made. Counts are kept of each possible outcome of every decision, and then compared to counts that would be expected from truly random sources.

We have found this new test to be more predictive for GA use than the standard Diehard suite. In particular, our new test was able to identify the PRNG that did cause unusual performance in our experiments, without the false positive behavior of the Diehard suite.

In conclusion, we feel that our new test of PRNG quality, tailored to the specific way that randomness is used in an actual GA, can be used by GA practitioners in order to choose PRNGs suited for their experiments.

Acknowledgments

This work is supported by the Doane College Cooper Undergraduate Research Program, the Initiative for Bioinformatics and Evolutionary STudies (IBEST) at the University of Idaho; by NIH NCRR grant 1P20RR016454-01; and by NIH NCRR grant NIH NCRR 1P20RR016448-01; and by NSF grant NSF EPS 809935.

References

[Marsaglia, 1993] Marsaglia, G. (1993). Monkey tests for random number generators. *Computers & Mathematics with Applications*, 9:1–10.

[Meysenburg et al., 2002] Meysenburg, M. M., Hoelting, D., McElvain, D., and Foster, J. A. (2002). How random generator quality impacts genetic algorithm performance. In *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Controlling Genetic Algorithms with Reinforcement Learning

James E. Pettinger

Dept. of Computer Science,
University of Exeter,
Exeter, EX4 4QT, UK.
J.E.Pettinger@exeter.ac.uk

Richard M. Everson

Dept. of Computer Science,
University of Exeter,
Exeter, EX4 4QT, UK.
R.M.Everson@exeter.ac.uk

Abstract

Here we present a hybrid system that uses a reinforcement learning agent to improve the performance of a genetic algorithm on the travelling salesman problem (TSP). The agent uses Q(λ) learning to estimate state-action utility values, which it uses to implement high-level adaptive control over the genetic algorithm. In this way the agent influences selection of both crossover and mutation operators as well as the selection of individuals for breeding, at every generation.

1 RL-GA MODEL

One of the weaknesses of genetic algorithms (GAs) is that there is a myriad of choices to be made in their implementation, such how frequently to crossover and mutate, how to determine which individuals will be selected for breeding, what operators to use for crossover and mutation, and so on. In study this we attempt to enhance the performance of a GA by using a reinforcement learning agent to learn to how adaptively control some of these aspects.

Our model itself contains two parts: a genetic algorithm and a reinforcement learning (RL) agent. The GA is capable of functioning independently but it is non-adaptive. The GA population itself forms the current state of the RL agent, which is represented using a set of state features. At each timestep the agent has certain actions available to it. Each action represents two things, the particular crossover or mutation operator the GA will use at the current timestep and the fitness class of the breeding pair or individual used with the selected operator. Each potential parent was classed as Fit (F) if it lay in the top 10% of the population and Unfit (U) if it did not. Thus, a crossover operator can be used with one of 4 types of parent pair, {FF, FU, UF, UU} and a mutation operator can be used with 2 types of individual, {F, U}. We use 3 crossover operators and 4 mutation operators – giving 20 composite actions in total.

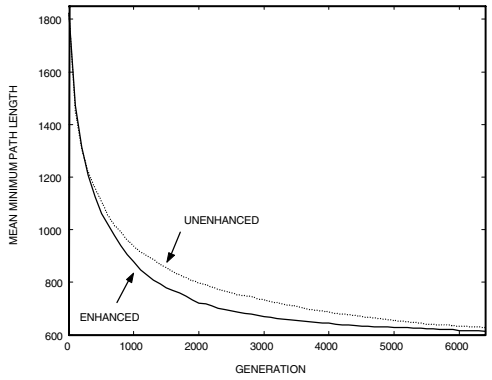


Figure 1 : Enhanced and Unenhanced GA Performance

Each action has a set of state-action utilites associated with it where each utility is an estimate of the reward that will be obtained if that action is used in a given GA state. An agent selects actions with a probability proportional to each state-action utility estimate. This scheme permits the agent to control both the genetic operator used and the particular individuals acted upon, dependant on the current population characteristics.

The system is trained by rewarding the agent after each action is taken. Reward is calculated by comparing the fitness of the parents and offspring. The better the offspring are, in comparison to the parents, the larger the reward and *visa versa*. This reward is used to update the utility of the action taken using Q(λ) learning. In this way actions that give larger rewards in a certain GA state are selected from that state with a higher probability. We then froze the learned utility estimates and applied the trained system to a test problem. Figure 1 shows the mean best solution across 50 test runs of a 40-city TSP for an RL enhanced GA and an unenhanced control GA.

2 CONCLUSIONS

Initial results on our test problem are encouraging with an RL-GA hybrid system outperforming an analogous non-adaptive GA system, on a 40-city TSP test problem. The hybrid system produces a better solution overall and learns faster. Results on a larger 100-city problem also show improvement.

An Integrated System for Phylogenetic Inference using Evolutionary Algorithms

Oclair Prado
Fundação CPqD
Campinas, SP, Brazil 13088-902

Fernando J. Von Zuben
DCA/FEEC/Unicamp
Campinas, SP, Brazil 13083-970

Abstract

This paper presents all the steps to reconstruct phylogenetic trees using an evolutionary algorithm. The fitness criterion is based on maximum likelihood and a toolbox is available. The codification, genetic operators and the optimization procedures involved are clearly described to guarantee reproducibility.

1 INTRODUCTION

Construction of phylogenetic trees is one of the most important problems in evolutionary research. According to Yoshikawa *et al.* (1999), a phylogenetic tree is a tree-structured graph that represents the evolutionary process of genes, and is constructed from sequential data (such as DNA sequences) obtained from several organisms, that will represent the leaves of the tree.

A difficulty here is the lack of information, because we do not have data from the common ancestors, and they must be inferred from the analysis of the current organisms.

Finding a good tree is an NP-Complete problem (Day, 1987), and the number of candidate trees may be calculated using the following formula:

$$\frac{(2n-3)!}{2^{n-2}(n-2)!} \qquad (1)$$

2 RESULTS AND DISCUSSION

Using DNA mitochondrial sequences of human, chimpanzee, gorilla, orangutan, and gibbon in the case of 5 leaves (Weir, 1996), Table 1 shows some comparative results obtained from three toolboxes developed to reconstruct trees: our software, called Phylogenetic Tree Project (PTP) (Prado & Von Zuben, 2001), PAML (Yang, 2000) and Phylip (Felsenstein, 1990). The results are close, but not equal, and it is expected to be this way, because each tool has its own features.

Since Maximum Likelihood method tries to maximize the probability of the data given a tree (Nei & Kumar, 2000), the best results in Table 1 are the ones obtained with Phylip. Phylip uses unrooted trees, so that less branches are involved. The final likelihood value is strongly affected by the number of branches. Anyway, the purpose here is not to obtain better results, when compared to other available toolboxes, but to attest the correct implementation of an evolutionary methodology (the reconstructed phylogenetic trees are almost identical), providing all the steps necessary to reproduce the results, what is not possible based on similar approaches in the literature (Matsuda, 1996).

Table 1: Comparative results using PTP, PAML and PHYLIP

TOOL	4 leaves	5 leaves
PHYLIP	-130.74914	-163.87052
PTP	-136.14495	-171.11214
PAML	-136.19721	-174.34676

References

Day, W.H.E, *Computational complexity of inferring phylogenies from dissimilarity matrice*, Bull. Math. Biol, 49:461-467, 1987.

Felsenstein, J. *PHYLIP Manual Version 3.3* University Herbarium, University of California, Berkeley, 1990.

Yoshikawa, T., Tabe, T., Kishinami, R., Matsuda, H. & Hashimoto, A. *On the Implementation of a Phylogenetic Tree Database*, Proc. of IEEE Pacific Conference on Communications, Computers, and Signal Processing, pp.42-45, August, 1999.

Matsuda, H. *Protein phylogenetic inference using maximum likelihood with a genetic algorithm*, Pacific Symposium on Biocomputing. World Scientific, London, pp. 512-523, 1996.

Nei, M. & Kumar, S. *Moledular Evolution and Phylogenetics*, Oxford University press, 2000.

Prado, O. & Von Zuben, F.J. The Phylogenetic Tree Project (PTP). Toolbox available at ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/oclair/. 2001.

Weir, B.S. *Genetic Data Analysis* II, Sinauer, Sunderland, MA, 1996.

Yang, Z. Phylogenetic *analysis by maximum likelihood (PAML)*, version 3.0. University College London, London, England, 2000.

A Genetic Algorithm for Improved Shellsort Sequences

Robert S. Roos **Tiffany Bennett** **Jennifer Hannon** **Elizabeth Zehner**

Department of Computer Science
Allegheny College
Meadville, PA 16335
rroos@allegheny.edu

Abstract

A genetic algorithm is used to find shellsort increment sequences that give good average-case performance on sequences of bounded size. Our approach applies a local improvement by removing a set of common factors from each increment sequence.

1 INTRODUCTION

A complete analysis of the average case behavior of shellsort has remained an unsolved problem in the theory of algorithms for forty years (Sedgewick, 1996). This work seeks to evolve increment sequences that sort in less time than known sequences. The authors know of only one other attempt to apply genetic algorithms to the generation of shellsort sequences (Simpson and Yachavaram, 1999).

2 METHOD

We use a steady-state genetic algorithm that tries to minimize the clock time needed to sort a particular file of one million elements. We permit at most one multiple of two and at most one multiple of three in any increment sequence. Our population size is 50; the number of generations is 100.

Crossover selection is according to fitness rank and is performed once in each generation. Mutation, consisting of adding a random integer to a randomly selected increment, is applied after a crossover, giving an effective mutation rate of .02.

On files of size half a million and one million, our sequence, {1, 8, 23, 131, 149, 155, 877, 2585, 5267, 13229, 72985, 91433}, outperformed the sequence of Simpson and Yachavaram (S/Y) and the sequence of Sedgewick that was used by S/Y.

Table 1 shows a representative set of data on several files of sizes 500000 and 1000000.

Table 1: Best/Median/Worst Sorting Times (ms)

FILE	OURS	S/Y
500000.0	721/724/737	749/750/752
500000.1	722/724/726	749/750/751
500000.2	721/724/725	749/751/752
1000000.0	1585/1606/1610	1677/1678/1678
1000000.1	1602/1603/1614	1678/1680/1681
1000000.2	1610/1613/1616	1678/1680/1682

Acknowledgments

This work was funded by a Collaborative Research Experience for Women in Undergraduate Computer Science and Engineering (CREW) grant, sponsored by the Computing Research Association's Committee on the Status of Women in Computing Research (CRA-W) in cooperation with National Science Foundation's Partnership for Advanced Computational Infrastructure's Education, Outreach and Training program.

References

Sedgewick, R. (1996). Analysis of shellsort and related algorithms. In *Algorithms—ESA '96, Fourth Annual European Symposium, Barcelona, Spain.* Lecture Notes in Computer Science 1136, Springer-Verlag, 1–11.

Simpson, R., and Yachavaram, S. (1999). Faster shellsort sequences: a genetic algorithm application. *Proceedings of the International Society for Computers and Their Applications(ISCA), April 7-9, 1999.* (First author's home page: http://cs.mwsu.edu/%7Esimpson/)

The Influence of Binary Representations of Integers on the Performance of Selectorecombinative Genetic Algorithms

Franz Rothlauf
University of Bayreuth, D-95440 Bayreuth/Germany, rothlauf@uni-bayreuth.de

1 Introduction

The discussion regarding the benefits of different binary representations of integers has a long tradition in Genetic and Evolutionary Algorithms (GEAs). However, besides the commonly used binary and gray encoding there are many other possible representations. We illustrate how the performance of selectorecombinative GAs depends on the used representation. We obtain three results: The choice of a proper binary representation is crucial for GEA success. When using selectorecombinative GAs the binary encoding can outperform the gray encoding. Encodings exist that are better than the gray or binary encoding.

2 Genotype-Phenotype and Phenotype-Fitness Mappings

When using a representation the fitness function f can be decomposed into

$$f_g(\boldsymbol{x}_g) : \Phi_g \rightarrow \Phi_p,$$
$$f_p(\boldsymbol{x}_p) : \Phi_p \rightarrow \mathbb{R},$$

where $f = f_p \circ f_g = f_p(f_g(\boldsymbol{x}_g))$, Φ_g is the genotypic and Φ_p is the phenotypic search space. The genotype-phenotype mapping f_g is the used representation. With a bitstring of length l we can represent 2^l different phenotypes. Therefore, the number of different representations is $2^l!$. For $l = 3$, there are $2^3! = 40\,320$ different representations. To reduce the number of different f_g we limit ourself to $l = 3$ and assume that $\boldsymbol{x}_g = 000 \in \Phi_g$ is always assigned to $x_p = 0 \in \Phi_p$. Then, for $l = 3$ the number of different representations is reduced to $(2^l - 1)! = 7! = 5040$.

f_p represents the fitness function and assigns a fitness value $f_p(\boldsymbol{x}_p)$ to every phenotype $\boldsymbol{x}_p \in \Phi_p$. For our investigation we use the easy problem $f_p(x_p) = x_p$.

3 Experimental Results

We concatenate 20 sub-problems of size $l = 3$. The fitness of an individual is calculated as the sum over the fitness of the 20 sub-problems. We use a selectorecombinative GA only using uniform crossover, tournament selection without replacement of size 2, and a population size $n = 20$. We performed 250 runs for each of the 5040 different genotype-phenotype mappings. A sub-problem is correctly solved if the GA is able to find the best solution $x_p = 7$. Figure 1 shows the distribution of the number of correctly solved sub-problems at the end of a GA run for all 5040 different types of genotype-phenotype mappings.

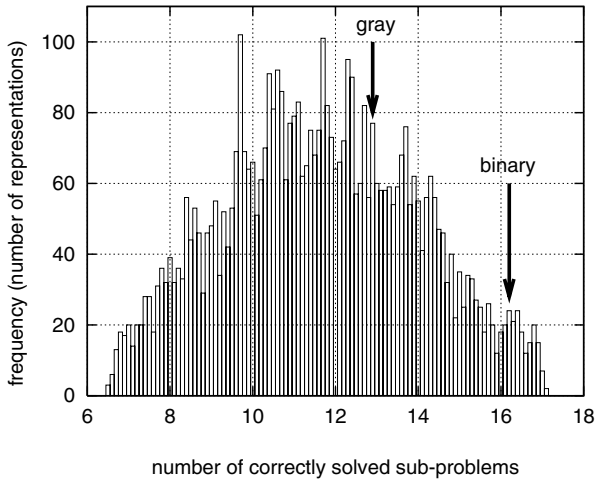


Figure 1: Experimental results

4 Conclusions

The results show that different genotype-phenotype mappings, that means assigning the genotypes $\boldsymbol{x}_g \in \{0,1\}^3$ in a different way to the phenotypes $x_p \in \{0,\dots 7\}$, strongly change the performance of GAs. Furthermore, for the considered easy problem the binary encoding (16.2 of the 20 sub-problems are correctly solved) performs better that the gray encoding (only 12.9 out of 20 sub-problems are correct). Finally, representations exist that outperform the binary encoding. If we can theoretically describe the properties of these encodings we can solve integer optimization problems more efficiently.

Genetic Algorithm Based Adaptive Control of an Electromechanical MIMO System

Ivan Sekaj
Dept. of Automatic Control Systems
FEI, Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, Slovak Republic

Martin Foltin
Dept. of Automatic Control Systems
FEI, Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, Slovak Republic

Michal Gonos
Dept. of Automatic Control Systems
FEI, Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, Slovak Republic

Abstract

An adaptive control algorithm design for an electromechanical system has been introduced. The adaptation mechanism covers the controlled system identification executed after each detection of the system dynamics behavior change and a genetic algorithm–based controller design procedure. As the moving-optimum optimization problem has a repetitive nature, the convergence rate of the genetic algorithm can be optimized using the proposed method, which exploits an archive of previous solutions.

1 ADAPTIVE CONTROL USING GA

The task was to design an adaptive control algorithm for an electro-mechanical multi-input multi-output (MIMO) system with two servomotors. Both subsystems (motors) have strong interactions and additionally, their dynamics changes with the changing mechanical load. The aim was to independently control the speed of each drive. To control this system, a four-controller structure with parameter adaptation has been proposed. The adaptation mechanism employs identification of the controlled MIMO system during its normal operation in the closed-loop using artificial neural networks (ANN). The ANN model identification follows after each detection of the controlled system dynamics change. The adaptation cycle is completed by the controller parameters computation performed by GA. In this moving-optimum problem the environment conditions repeat more or less periodically or the new conditions are similar to those, which already occurred in history. In order to timely respond to these environment changes, a sufficient convergence speed is an important requirement for the GA procedure. From this reason a new method for GA acceleration has been proposed. The method uses data, which are archived from previous solutions and which can be exploited under repeated or similar environment conditions.

2 ARCHIVE OF PAST SOLUTIONS

The principle of the proposed mechanism consists in repeating the procedure of saving the current solution and searching for past solutions after each reformulation of the optimization task. That means, that in the first phase, a standard GA finds an optimal (suboptimal) solution under the current conditions and saves the best solution into the archive. The archive has the form of a matrix, where each row is a record (string) of a solution under some environment conditions. When any new environment change occurs, such a solution is searched in the archive, which best meets the new criterion. This string (or several strings) is copied into the new population of the GA and the process continues. Another possible way is to use the best string only in the reinitialization phase of the new GA run. Anyway, the obtained results have shown, that the exploitation of the archive considerably accelerates the next GA convergence.

The described GA-based adaptation approach is robust and powerful and can be used also in many other application areas.

References

Y.Mitsukura, T.Yamamoto, M.Kaneda, K.Fujii (1999). Evolutionary Computation in Designing a PID Control System. *IFAC World Congress, Beijing, 5th-9th july 1999, P.R.China, pp. 497-502*

P.Ošmera, J.Roupec, R.Matoušek. (2000). Genetic Algorithms with Sexual Reproduction. *SCI 2000, Orlando, USA, pp. 306-311*

K.S.Narendra, S.Mukhopadhyay (1994). Adaptive control of nonlinear multivariable systems using neural networks. *Neural Networks 7, 737-752*

Parametric Study to Enhance Genetic Algorithm's Performance when Using Transformation

Anabela Simões

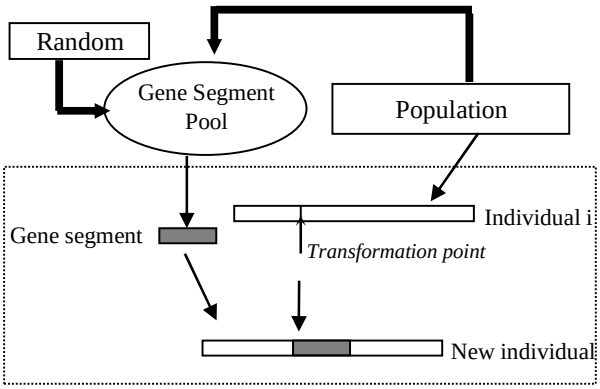
Department of Informatics and Systems Engineering
Coimbra Polytechnic
Quinta da Nora, 3030 Coimbra,
Portugal
abs@isec.pt

Ernesto Costa

Centre for Informatics and Systems of the
University of Coimbra
Polo II, Pinhal de Marrocos, 3030 Coimbra
Portugal
ernesto@dei.uc.pt

Extended Abstract

Transformation is a biologically inspired genetic operator that, when incorporated in the standard Genetic Algorithm replacing crossover, can promote diversity in the population (Simões and Costa 2001). The computational mechanism mimics the biological process and consists in the capacity of the individuals to absorb fragments of DNA from the environment. These gene segments are then reintegrated in the individuals' genome (see figure).



We have done an extensive empirical study carried to determine the best parameter setting to use with transformation in order to enhance the GA's performance. These parameters include the gene segment length, the replacement rate (percentage of individuals of the previous population used to update the gene segment pool), and the mutation and transformation rates.

The tests were made in two domains: function optimization (minimization) and combinatorial optimization (maximization). The chosen functions were: Ackley, Griewangk, Rastrigin and Schwefel and the 0/1 Knapsack problem (0/1 KP) for combinatorial optimization.

The tables below show the results obtained with, and without, the best parameters settings, for the case of function optimization and for the 0/1 knapsack.

Parametric Study				**Random Choice of Parameters**			
Segment length=5 Replacement Rate=90% Transformation Rate= 70% Mutation Rate=0.0%				Segment length=random Replacement Rate=90% Transformation Rate= 70% Mutation Rate=0.1%			
Nº evals->	*50000*	*100000*	*200000*	*Nº evals->*	*50000*	*100000*	*200000*
Ackley	2.678	0.044	**0.002**	Ackley	3.128	0.300	0.002
Griewangk	0.001	0.000	**0.000**	Griewangk	0.010	0.003	0.001
Rastrigin	8.290	0.821	**0.001**	Rastrigin	38.401	18.828	6.540
Schwefel	0.147	0.031	**0.008**	Schwefel	36.212	0.475	0.077

Parametric Study			**Random Choice of Parameters**		
Segment length=5 Replacement Rate=50% Transf. Rate= 90% Mutation Rate=0.0%			Segment length=random Replacement Rate=90% Transf. Rate= 70% Mutation Rate=0.1%		
Pop size->	*50*	*100*	*Pop size->*	*50*	*100*
50 items	204.60	**204.90**	50 items	197.30	197.80
100 items	442.50	**444.47**	100 items	413.00	408.40
250 items	**955.20**	954.60	250 items	838.50	834.87
500 items	**1926.87**	1910.00	500 items	1666.20	1669.07

As we can see, choosing the GA parameters with some criteria, the results obtained were quite better than the results achieved in our initial work.

Acknowledgments

This work was partially financed by the Portuguese Ministry of Science and Technology under the Program POSI.

References

A. Simões, E. Costa (2001). *On Biologically Inspired Genetic Operators: Using Transformation in the Standard Genetic Algorithm*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), pp. 584-591, San Francisco, USA, 7-11 July, Morgan Kaufmann Publishers, 2001.

Using GAs to Deal with Dynamic Environments: A Comparative Study of Several Approaches Based on Promoting Diversity

Anabela Simões

Department of Informatics and Systems Engineering
Coimbra Polytechnic
Quinta da Nora, 3030 Coimbra,
Portugal
abs@isec.pt

Ernesto Costa

Centre for Informatics and Systems of the
University of Coimbra
Polo II, Pinhal de Marrocos, 3030 Coimbra
Portugal
ernesto@dei.uc.pt

Extended Abstract

Most approaches used in EAs for problems in which the environment changes from time to time, try to preserve the diversity of the population. One of those approaches applies a new biologically inspired genetic operator called transformation (Simões & Costa, 2001). We tested two EAs using transformation (**TGA** and **ETGA**) and two other classical approaches: random immigrants (**RIGA**) and hypermutation (**HMGA**). The comparative study was made using the dynamic 0/1 Knapsack optimization problem.

Transformation is a process that modify certain bacteria (and occasionally other cells as well) which, when grow in the presence of killed cells, take up foreign DNA from that cells and acquire characters encoded by it. We incorporate transformation into the standard genetic algorithm as a new genetic operator that replaces crossover, and call the new algorithm **TGA**. The foreign DNA fragments, consisting of binary strings of different lengths, will form a gene segment pool and will be used to transform the individuals of the population. **ETGA**, is a enhanced version of TGA, where the parameters for transformation were optimized.

The **0/1 knapsack problem** is defined as follows: given a set of n items, each with a weight $W[i]$ and a profit $P[i]$, with $i = 1, ..., n$, the goal is to determine which items to include in the knapsack so that the total weight is less than some given limit (C) and the total profit is as large as possible. When the weight limit can change over time, we have the **dynamic** version. We used three types of changes in the capacity of the knapsack: periodic changes between two values (C1=104 and C2=60) and between three values (C1=60, C2=104 and C3=80) and non-periodic changes between 3 different capacities (C1=60, C2=80 and C3=104).

To evaluate the performance of the algorithms we used two measures: **accuracy**, the difference between the value of the current best individual in the population of "just before change" generation and the optimum value averaged over the entire cycle; **adaptability**, the difference between the value of the current best individual of each generation and the optimum value averaged over the entire cycle. The tables bellow show the results obtained with periodic changes among three values.

Accuracy	**TGA**	**ETGA**	**HMGA**	**RIGA**
Cycle = 30	5.27	1.45	**0.70**	1.08
Cycle = 100	2.53	**0.18**	0.36	0.22
Cycle = 200	1.27	**0.02**	0.42	0.15
Cycle = 300	0.78	**0.01**	0.49	0.22

Adaptability	**TGA**	**ETGA**	**HMGA**	**RIGA**
Cycle=30	7.63	3.53	**1.84**	2.74
Cycle=100	4.59	**1.30**	2.18	1.16
Cycle=200	3.10	**0.61**	1.39	0.65
Cycle=300	2.40	**0.42**	1.35	0.58

We can conclude that, in general, the approach based on transformation is a good candidate to be used in situations where the environment is dynamic, particularly in cases where the cycle length is greater than 30.

Acknowledgements

This work was partially financed by the Portuguese Ministry of Science and Technology under the Program POSI.

References

A. Simões and E. Costa (2001). *On Biologically Inspired Genetic Operators: Transformation in the Standard Genetic Algorithm*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), San Francisco, USA, July 2001.

Modified Linkage Learning Genetic Algorithm for Difficult Non-Stationary problems

Abhishek Singh
Dept. of Civil and Env. Engg.
University of Illinois at Urbana-Champaign
Urbana, IL 61820
asingh8@uiuc.edu

David E. Goldberg
Dept. of General Engg.
University of Illinois at Urbana Champaign
Urbana, IL 61820
deg@uiuc.edu

Ying-Ping Chen
Dept. of General Engg.
University of Illinois at Urbana Champaign
Urbana, IL 61820
ychen21@uiuc.edu

Abstract

The linkage learning genetic algorithm (LLGA) proposed by Harik (Harik 1997), evolved tight linkage in a bid to solve difficult problems. This paper extends this work to difficult *non-stationary* problems. The probabilistic expression mechanism of the LLGA is akin to the dominance and polyploidy found in nature. This redundancy of gene expression found in the LLGA was found to benefit the adaptation of the solution to dynamic fitness landscapes. However as the LLGA converges to tight linkage the available diversity decreases considerably. In this study it was found that by allowing disruption of tightly linked structures (with low probability) through the crossover operator and introducing explicit mutation and diploidy, much improvement could be gained in solving non-stationary deceptive problems without relearning the linkage evolved over the run.

1 NON-STATIONARY OPTIMIZATION USING MODIFIED LLGA

Our non-stationary problem consisted of *k*-bit trap functions and TMMPs (truncated massively multimodal problems) (Harik, 1997). To introduce non-stationarity an intermediate, periodically changing, XOR mask was used to change the expression of the genotype, in effect changing the fitness landscape for a given genotype.

The original LLGA performed satisfactorily on this problem when the fitness change was rapid, however it converged to tightly linked building blocks when the change was slow. The diversity in the LLGA introduced through the redundancy in the extended probabilistic mechanism (EPE-2) (Harik, 1997) was useless as an adaptive memory because as the LLGA progressed, building blocks clustered together making it increasingly difficult to express alternate solutions when fitness changed. This was primarily due to the non-disruptive crossover operator (that chose crossover points from non-coding regions) used by Harik. Allowing the crossover points to be within a 'coding region' allowed the shielded BBs to be expressed (analogous to the dominance change in traditional diploid GAs), and the new optima to emerge. This was at the cost of linkage, since every time crossover broke the linked structure the LLGA had to relearn the linkage. To tackle this an explicit diploidy scheme was introduced wherein each chromosome had a repressed partner (diploid) that could be expressed with a low rate. This significantly improved the performance of the LLGA (Figure 1), showing the importance of diploidy in non-stationary optimization.

The performance over concatenated traps and TMMPs was not as good. This was attributed to the sequential nature of linkage learning in LLGA, which caused a time scale problem with the fitness change due to which only partial BBs could be discovered in one epoch.

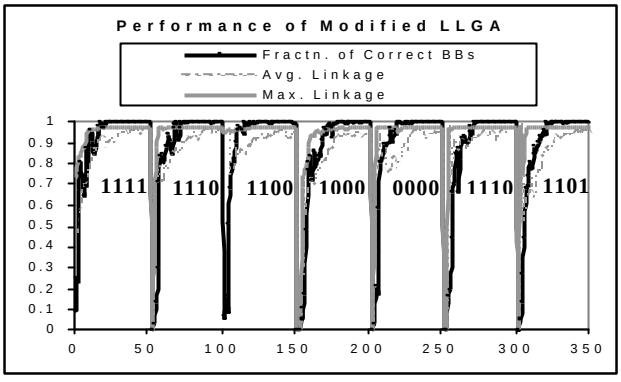


Fig 1. Performance of the modified LLGA (with changing optima in bold). The maximum linkage values are consistently high. In addition, the time for discovering the new optima decreases.

Reference

Harik, G R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*. Doctoral dissertation, University of Illinois, Urbana-Champaign. IlliGAL Report Number 97005.

Bi-directional circular linked lists in fitness caching

Tapio Tyni and Jari Ylinen
KONE Corporation, R & D
P.O. BOX 677
05801 Hyvinkää, FINLAND

Abstract

Caching the chromosomes with bi-directional circular linked lists reduces the number of fitness evaluations in the GA's providing a built-in bookkeeping property to store and maintain only the chromosomes referenced from the current active region of the problem search space. Keywords: GA, Caching, Epochal evolution.

1 PRINCIPLE AND TEST CASE

Caching is a commonly used technique in the computer science. It has been applied also with the GA's, e.g. in [Kra99]. Instead of the linear tables or linked lists, the bi-directional circular linked lists in the context of GA's are able to reduce the memory space requirements of the cached chromosomes by exploiting the epochal behaviour of the evolutionary search. While the GA's proceed towards the solution of the problem the population has typically periods - epochs - when it wanders around a local extreme point. During these epochs, the number of unique chromosomes that standard mutation and crossover can produce is limited. Only the chromosomes belonging to the present region of the search space need to be cached. Arranging the hash table overflow areas as illustrated in the Figure 1 yields to the built-in behaviour of overwriting the older, obsolete chromosomes in the cache memory with the chromosomes from the current, active region of the search space. Extra bookkeeping to manage data is not needed. The evolution of the fitness value in the Figure 2b illustrates the epochal behaviour of the search with a simple string search problem. The contour map in the Figure 2a shows the age of the chromosomes in the cache memory. When a new epoch begins, the earlier cached chromosomes are no more valid and the new chromosomes from the active search space region conquer the cache. Good examples occur during the generations 36-62 and 63-100. The theoretical probability to find a chromosome from the cache with the actual hits is also shown in the Figure 2a [TY99]. The savings in computational load and memory space are often crucial in the embedded real world real time applications. There the presented method may offer one alternative to obtain the tough goals of an online system.

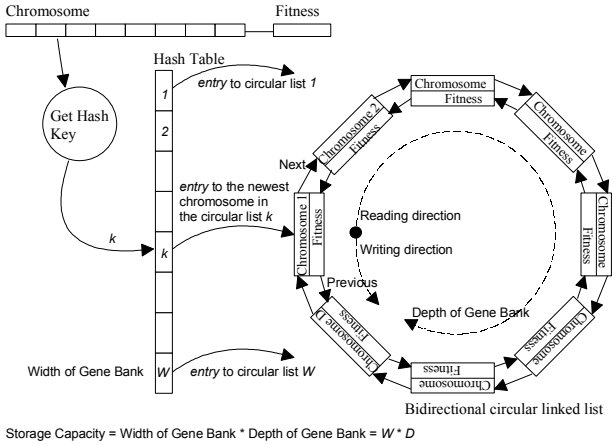


Figure 1. Illustration of the cache organisation

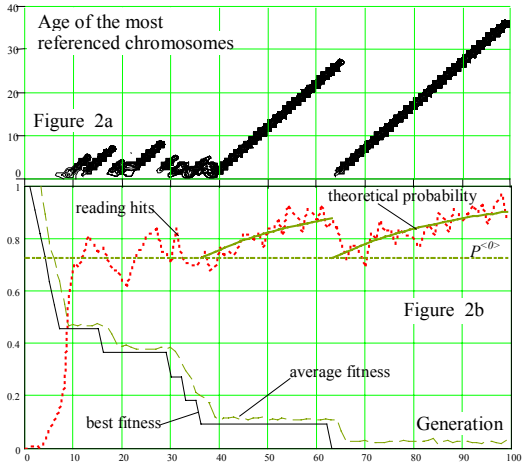


Figure 2. A simple string search test problem

References

[Kra99] Jozef Kratica: *Improving Performances of the Genetic Algorithm by Caching*, Computers and Artificial Intelligence, Vol. 18, No. 3, pp. 271-283, 1999.

[TY99] Tyni T., Ylinen J.: *Improving the Performance of Genetic Algorithms with a Gene Bank*. Proceedings of EUROGEN99, Report A2/1999, University of Jyväskylä, Finland, pp. 162-170, 1999.

Application of numerical optimization technique based on real-coded genetic algorithm to inverse problem in biochemical systems

Takanori Ueda
Graduate School of Bioresource and Bioenvironmental Sciences
Kyushu University
Fukuoka 812-8581, Japan
ueda@brs.kyushu-u.ac.jp

Nobuto Koga
Dept. of Biochem. Eng. & Sci.
Kyushu Institute of Technology
Fukuoka 820-8502, Japan
nob@brs.kyushu-u.ac.jp

Isao Ono
Faculty of Eng.
The University of Tokushima
Tokushima 770-8506, Japan
isao@is.tokushima-u.ac.jp

Masahiro Okamoto
Graduate School of Bioresource and Bioenvironmental Sciences
Kyushu University
Fukuoka 812-8581, Japan
okahon@brs.kyushu-u.ac.jp

Abstract

Real-coded Genetic Algorithms (RCGA) attract attention as numerical optimization methods for nonlinear systems. One of the crossover operators for RCGA called unimodal normal distribution crossover (UNDX) has shown good performance in optimization of various functions including multi-modal ones and benchmark functions with epistasis among parameters (Ono and Kobayashi, 1997). The UNDX generates new population lie on some ponds or along some valleys in order to focus the search on promising areas from a viewpoint of searching efficiency. Especially when the function has epistasis among parameters, namely valleys that are not parallel to coordinate axis, the UNDX can efficiently optimize it. Simple GA is one of the well-known generation alternation models, however, it has two problems. One is early convergence in the fast stage of search and the other is evolutionary stagnation in the last stage of it. A new generation alternation model called minimal generation gap (MGG) was proposed to overcome the above problems (Sato et al., 1997, Ono et al., 2000). The MGG has all advantages of convention models and the ability of avoiding the early convergence and suppressing the evolutionary stagnation.

Here, we applied the combination method (MGG+UNDX) to the optimization of real-valued parameters in the form of the S-system that is a type of power-law formalism and is suitable for description of organizationally complex systems such as gene expression networks (Maki et al., 2001) and metabolic pathways (nonlinear biochemical systems). The S-system is based on a particular type of ordinary differential equation in which such component processes are characterized by power-law functions as follows:

$$\frac{d}{dt}X_i = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}} \qquad (1)$$

where n is the total number of state variables or reactants (X_i), i, j ($1 \le i,\ j \le n$) are suffixes of state variables. The terms g_{ij} and h_{ij} are interactive effectivity of X_j to X_i. The first term represents all influences that increase X_i, whereas the second term represents all influences that decrease X_i. In a reaction network context, the non-negative parameters α_i and β_i are called relative inflow and outflow of reactant X_j, and real-valued exponents g_{ij} and h_{ij} are referred to as the interrelated coefficients between reactants X_j and X_i. Since the S-system is a formalism of ordinary nonlinear differential equation, the system can easily be solved numerically by using a numerical calculation program to be customized specifically for this structures. However, when an adequate time-course of relevant state variable is given, a set of parameter values α_i, β_i, g_{ij} and h_{ij}, in many cases, will not be uniquely determined, because it is highly possible that the other sets of parameter values will also show a similar time-course. Therefore, even if one set of parameter values that matches the observed time-courses is obtained, this set is still one of the best candidates that explain the observed time-courses. Our strategy is to explore and exploit these candidates within the immense huge searching space of parameter values. In the results, MGG+UNDX showed superiority in searching eficiency to the simple GA as the precision of the optimization becomes higher.

Acknowledgement

This work was partially supported by the Grants-in-Aid for Scientific Research on Priority Areas (C), “Genome Information Sciences” (No.12208008) from the Ministry of Education, Culture, Sports, Science and Technology in Japan (MO).

LCGA : Local Cultivation Genetic Algorithm for Multi-Objective Optimization Problems

Shinya Watanabe
Graduate School of Engineering,
Doshisha University
1-3 Tatara Miyakodani,Kyo-tanabe,
Kyoto, 610-0321, JAPAN

Tomoyuki Hiroyasu
Faculty of Engineering,
Doshisha University

Mitsunori Miki
Faculty of Engineering,
Doshisha University

Abstract

In this paper, a new genetic algorithm for multi-objective optimization problems is introduced. That is called "Local Cultivation GA (LCGA)". LCGA has a neighborhood crossover mechanism in addition to the mechanism of GAs that had proposed in the past researches. As compared with SPEA2, NSGA-II, and MOGA, LCGA is the robust algorithm which should find the Pareto optimum solution. Since LCGA is easy to implement to parallel computer as a master-slave model, the reduction of calculation cost can be expected.

1 Local Cultivation GA

We develop a new algorithm that is called Local Cultivation Genetic Algorithm (LCGA). LCGA has a neighborhood crossover mechanism in addition to the mechanisms of GAs that had proposed in the past researches. The following mechanisms are included in LCGA.

1) Preservation mechanism of the excellent solutions
2) Reflection mechanism of the preserved excellent solutions
3) Cut down (sharing) method of the preserved excellent solutions
4) Assignment method of fitness function
5) Normalization mechanism of values of each object

In LCGA, the exploitation factor of the crossover is reinforced. In the crossover operation of LCGA, a pair of the individuals for crossover is not chosen randomly, but individuals who are close each other are chosen. Because of this operation, child individuals that are generated after the crossover may be close to the parent individuals. Therefore, the precise exploitation is expected.

2 Numerical Examples

To discuss the effectiveness of the proposed method, LCGA was applied to test functions and results were compared to the other methods; those are SPEA2, NSGA-II and MOGA.

Figure 1 shows the derived Pareto solutions of KUR. These are the results of 10 trials. In this figure, LCGA derived better solutions than the other methods. Several other experimental results also show the similar tendencies. Through the numerical examples, the following points became clear.

- In all the test functions, LCGA derived better solutions than the other methods. From this result, it can be noted that the neighborhood crossover acts to derive the good solutions.
- On the other hand, the other methods can get good solutions in particular test function. Therefore, it can be concluded that LCGA is a robust method to find Pareto optimum solutions.

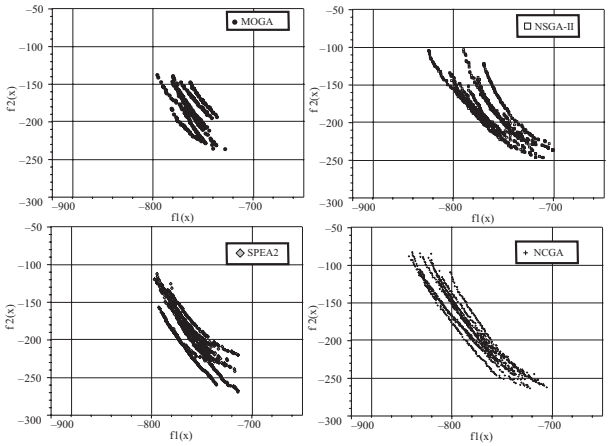


Figure 1: Derived Pareto individuals(KUR)

The Proportional Genetic Algorithm Representation

Annie S. Wu
School of EECS
University of Central Florida
Orlando, FL 32816-2362
aswu@cs.ucf.edu

Ivan Garibay
School of EECS
University of Central Florida
Orlando, FL 32816-2362
igaribay@cs.ucf.edu

1 Introduction

We have developed a genetic algorithm (GA) with a new representation method which we call the proportional GA (PGA). The PGA representation focuses on the idea that it is the content rather than the order of the encoded information that matters. As a result, the PGA representation is based on multisets rather than permutations. We extend the idea of location independent representations to a generally usable encoding for integer and floating point numbers. Specifically, the PGA assigns one or more unique characters to each parameter or component of a solution. The value of a parameter is determined from the proportion of the characters assigned to that parameter as compared with the total number of characters in the individual, or from the relative proportions of the assigned characters of that parameter. Thus, characters that exist are "expressed" and, consequently, interact with other expressed characters. Characters that do not exist are "not expressed" and simply do not participate in the interactions of the expressed characters. This representation may be further extended with the addition of *non-coding* characters that are not associated with any parameters. These non-coding regions are beneficial for fine tuning purposes.

We tested three variations of the PGA in our experiments. PGA1 is specialized for resource allocation problems. PGA2 and PGA3 may be applied to the general class of problems in which one is searching for a vector of values. Length restrictions limit our discussion here to PGA2 and PGA3. Full details about the PGA are available in [1].

2 Representation details

A consequence of the PGA's set-based representation is that the two following example individuals encode the same information.

```
AccBdDeeEbAbBDEccaAAAEebbEEECCDbbbABCDEedcbaAAddbA
AAAAAAAAAaaBBBbbbbbbbbbCCCccccDDDDddddEEEEEEEeeee
```

The following two tables show how PGA2 and PGA3, respectively, would decode the example individuals into expressed values. In both cases,
Expressed value $= V_{i,min} + pct(V_i) \times (V_{i,max} - V_{i,min})$.
V_{min} and V_{max} are predefined constants.

Value V	V_{min}	V_{max}	# *positive_char*(V)	# *negative_char*(V)	*pct*(V)	Expressed value
V_1	0	10	9 A's	2 a's	$9/(9+2)$	8.18
V_2	0	10	3 B's	9 b's	$3/(3+9)$	2.5
V_3	0	10	3 C's	5 c's	$3/(3+5)$	3.75
V_4	0	10	4 D's	4 d's	$4/(4+4)$	5.0
V_5	0	10	7 E's	4 e's	$7/(7+4)$	6.36

PGA2: Character assignment and expressed values.
$pct(V_i) = \frac{positive\_char(V_i)}{positive\_char(V_i)+negative\_char(V_i)}$

Value V	V_{min}	V_{max}	# *positive_char*(V)	# *negative_char*(V)	*pct*(V)	Expressed value
V_1	0	10	9 A's	2 a's	2/9	2.22
V_2	0	10	3 B's	9 b's	3/9	3.33
V_3	0	10	3 C's	5 c's	3/5	6.0
V_4	0	10	4 D's	4 d's	4/4	1.0
V_5	0	10	7 E's	4 e's	4/7	5.71

PGA3: Character assignment and expressed values.

$$pct(V_i) = \begin{cases} \frac{positive\_char(V_i)}{negative\_char(V_i)} & \text{if } positive\_char(V_i) \\ & < negative\_char(V_i) \\ \frac{negative\_char(V_i)}{positive\_char(V_i)} & \text{otherwise} \end{cases}$$

References

[1] A. S. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm. *Genetic Programming and Evolvable Hardware*, 2002. In press.

Climbing Unimodal Landscapes with Neutrality: A Case Study of the OneMax Problem

Tina Yu

ChevronTexaco Information Technology Company
San Ramon, CA 94583, U.S.A.
tiyu@chevrontexaco.com
http://www.improvise.com

Julian Miller

University of Birmingham
Birmingham, B152TT, U.K.
j.miller@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~jfm

Abstract

We investigate fitness neutrality in a Simple Evolutionary Algorithm (SEA) and in a neutrality-enabled evolutionary system using the OneMax problem. The results show that with the support of limited neutrality, SEA is less effective than our system where a larger amount of neutrality is supported. In order to understand the role of neutrality in evolutionary search of this unimodal landscape, we have created a theoretical framework that gives the number of gene changes under different levels of neutrality. The interim results of this theoretical work are also presented.

1 SUMMARY

OneMax was devised by Ackley as one of the benchmark problems to test the generality of his stochastic iterated genetic hill climbing search algorithm. This problem has a simple formula: for a binary string x of length l, the problem is to maximize:

$$\sum x_i,\ x_i \in \{0, 1\}^l.$$

There is only one optimum solution in this problem: the binary string with all its bit values equal to 1. This unimodal search space is most suitable for hill climbing search algorithms. As expected, he reported that hill climbing methods outperform genetic and simulated annealing search methods on this problem.

Although not an ideal problem to demonstrate the strength of Genetic Algorithms (GAs) , OneMax has been used by many researchers to study different aspects of GAs because of its simplicity. For example, (Bäck, 1992) studied optimal mutation rates and their interaction with selection and self-adaptation on this problem, while (Giguere and Goldberg, 1998) used this problem to investigate the sampling size and selection scheme in GAs. The purpose of these studies was to gain understanding of how GAs perform search, not to assess the search ability of GAs against other methods.

In this work, we investigate solving the OneMax problem using an evolutionary system that supports neutrality. The goal of this study is twofold:

- To assess the generality of the neutrality-enabled evolutionary system.
- To understand the search behavior of the neutrality-enabled evolutionary system on this problem.

Previously, we have devised an evolutionary system whose genotype representation contains extra inactive genes. Mutations acting on these inactive genes have a neutral effect on the genotype's fitness. However, neutral mutations can change the dynamics of the evolutionary search process. When applied to a Boolean function and four needle-in-haystack problems, the results show that higher search success rates were obtained when a higher amount of neutral mutations were permitted during search (Yu and Miller, 2001; Yu and Miller, 2002). These results encourage us to test the system on different kinds of problems to assess its generality. OneMax is chosen because its unimodal landscape has not been investigated previously.

The search behavior of the neutrality-enabled evolutionary system has been analyzed quantitatively using the ratio between active and inactive gene changes. Previously, we conducted experiments for empirical study of these ratios. In this work, we attempt a theoretical approach by formulating these ratios mathematically. Although this is a challenging task, our interim results indicate that it is achievable for the simple OneMax landscape.

References

Yu, T. and Miller, J. (2001). Neutrality and the evolvability of Boolean function landscape. *In Proceedings of the Fourth European Conference on Genetic Programming*. Pages 204-217.

Yu, T. and Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. In *Proceedings of the Fifth European Conference on Genetic Programming*. Pages 13-25.