European Graduate Student Workshop on Evolutionary Computation

1st European Workshop, EvoPhD 2006
Budapest, Hungary, April 10–12, 2006
Proceedings

Chairs: Mario Giacobini and Jano van Hemert

Preface

Evolutionary computation involves the study of problem-solving and optimization techniques inspired by principles of evolution and genetics. As any other scientific field, its success relies on the continuity provided by new researchers joining the field to help it progress. One of the most important sources for new researchers is the next generation of PhD students that are actively studying a topic relevant to this field. It is from this main observation the idea arose of providing a platform exclusively for PhD students.

From the perspective of a PhD student, it is important to work on an exciting topic that is of interest to other researchers in the field. Contact with these researchers early in the study can help in guiding the course by focusing on what other researchers deem important. Furthermore, those contacts are important for the social network of the PhD student, which will become vital for choosing the step following the PhD study.

We would like to stress that the work presented in the proceedings is not that of regular papers. Instead it is a collection of works where each student has described the project of his/her PhD thesis, and has outlined his/her achievements and goals of the PhD study. Moreover, a description is given of the desired feedback.

This volume contains the proceedings of EvoPhD 2006, the First European Graduate Student Workshop on Evolutionary Computation. It was held in Budapest, Hungary on 10–12 April 2006, as part of the EvoWorkshops 2006, which again runs jointly with the 9th European Conference on Genetic Programming and the 6th European Conference on Evolutionary Computation in Combinatorial Optimization.

We would like to acknowledge the work performed by the Program Committee, who has made sure, with a rigorous reviewing process, that this proceedings contains scientific works of a high quality. This year, seven manuscripts were submitted of which six were accepted for publication in the proceedings and, subsequently their authors have presented their work at the workshop. Each manuscript was reviewed independently by three reviewers, which gave insightful comments on the work as presented, but also on the topic itself and the proposed work and goals.

We are grateful to Franz Rothlauf for his moral support and helpful comments. Last, but not least, we thank Jennifer Willies for her administration of and efforts in both the conferences and the workshops, and especially for making this proceedings available on paper here today.

March 2006

Mario Giacobini and Jano van Hemert
Program Chairs

EvoPhD 2006

Organization

EvoPhD 2006 is organised as one of the workshops of the EvoWorkshops 2006, which runs in conjunction with EuroGP 2006 and EvoCOP 2006.

Organising Committee

EvoPhD Chairs:	Mario Giacobini University of Lausanne, Switzerland
	Jano van Hemert University of Edinburgh, United Kingdom
EvoWorkshops chair:	Franz Rothlauf University of Mannheim, Germany
Local Chair:	Anikó Ekárt Hungarian Academy of Sciences, Hungary
Publicity Chair:	Steven Gustafson University of Nottingham, UK

Program Committee

Enrique Alba	Steven Gustafson	Peter Ross
Anne Auger	Bill Langdon	Franz Rothlauf
Stefano Cagnoni	Tom Lenaerts	Conor Ryan
Mathieu Capcarrère	Evelyne Lutton	Giovanni Squillero
Pierre Collet	JJ Merelo	Christine Solnon
Ernesto Costa	Julian Miller	Marco Tomassini
Aniko Ekart	Daniele Radicioni	Leonardo Vanneschi
Jens Gottlieb	Günther Raidl	Sébastien Verel

Table of Contents

An Evolutionary Approach for Neural Model Optimization

Antonia Azzini

Università degli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
via Bramante 65, I-26013, Crema, Italy
azzini@dti.unimi.it

Abstract. Neuro-genetic systems are biologycally inspired computational models that use evolutionary algorithms (EAs) in conjunction with neural networks (NNs) to solve problems. The PhD research work presented in this paper describes an evolutionary approach to the joint optimization of neural network structure and weights which can take advantage of backpropagation as a specialized decoder. The approach is successfully applied to a real-world engine fault diagnosis problem and to a classification application of brain waves in the context of brain-computer interfaces.

1 Introduction

The evolutionary approach that implements the conjunction of evolutionary algorithms (EAs) with neural networks (NNs) is a more integrated way of designing ANNs. Neuro-genetic systems have become a very important topic of study in recent years since they allow all aspects of NN design to be taken into account at once and do not require expert knowledge of the problem. Much research has been undertaken on the combination of EAs and NNs. Through the use of EAs, the problem of designing a NN is regarded as an optimization problem. Some EAs have implemented search over the topology space, or a search for the optimal learning parameters. Some others focus on weight optimization: these can be regarded as alternative training algorithms, and in this case the evolution of weights assumes that the architecture of the network must be static. The main drawback using traditional gradient descent techniques such as backpropagation (BP) [10], lies in the trapping in local minima. Usually, a gradient descent algorithm is used to adapt the weights based on a comparison between the desired and actual network response to a given input stimulus. All training pairs, each consisting of input and desired output values, are forming a multi-dimensional error surface, getting stuck in local minima when moving across a rugged error surface. A common error surface has many local minima usually not meeting the desired convergence criterion. For this reason, rather than adapting weights based on local improvement only, EAs evolve weights based on the whole network fitness. Several works in this direction have been carried out by Yang and colleagues in [12] and by Zalzala and colleagues in [8]. The design

of an optimal NN architecture can be formulated as a search problem in the architecture space, where each point represents an architecture. As pointed out by Yao [13], given some performance (optimality) criteria, e.g., minimum error, learning speed, lower complexity, etc., about architectures, the performance level of all these forms a surface in the design space. Determining the optimal architecture design is equivalent to finding the highest point on this surface. Stanley and Miikkulainen in [11] also presented a neuro evolutionary method through augmenting topologies (NEAT). An interesting area of evolutionary NNs is the simultaneous evolution of different aspects of a NN. One of the most important is the combination of architecture and weight evolution, as presented in EPNet by Yao et al [14], and in work presented by Leung and colleagues [6]. The advantage of combining these two basic elements of a NN is that a completely functioning network can be evolved without any intervention by an expert.

2 Research

In this research we outline an approach to the design of NNs based on EAs, whose aim is both to find an optimal network architecture and to train the network on a given data set. The approach is designed to be able to take advantage of the backpropagation (BP) algorithm if that is possible and beneficial; however, it can also do without it. The basic idea is to exploit the ability of the EA to find a solution close enough to the global optimum, together with the ability of the BP algorithm to finely tune a solution and reach the nearest local minimum. This research was primed by an industrial application for the design of neural engine controllers, with particular attention to reduced power consumption and silicon area occupation. This application is described in Section 4. Then, another real-world application, that considers brain wave signal processing, has been successfully applied. In this case a classification algorithm in the analysis of P300 Evoked Potential is considered. It is described in Section 6.

2.1 Future Plans and Study

Further work on this problem during PhD research will include an in-depth study for parameters tuning and data set up, in order to improve the accuracy of classification. Furthermore we will apply our approach to a financial problem, related to financial market trends, currency exchange and industrial market. The aim of this approach is to model the mutual relationship among several financial instruments. For instance we are now investigating the relationship among the Dow Jones Industrial Average and a number of other market indices, including foreign exchange rates, market segments, and commodity prices. We are also investigating novel crossover operators for ANNs. As indicated in [7] there has been some debate in the literature about the opportunity of applying crossover to ANN evolution, based on disruptive effects that it could make into neural model. Our idea is to implement a kind of vertical crossover, defining a *merge-operator* between the topologies and weights matrices of two parents in order to

create the offspring. I am in the third and last year of the PhD curriculum in Computer Science of the University of Milan. At the end of each year of PhD course, all research works are taken into account and evaluated by an academic commission in order to make a critical analysis of the work and to give a positive outcome. I think that one of the major drawbacks of this research area is the lack of open positions in Italian universities in this field. Therefore I have to take into serious consideration the possibility of looking for a post-doctoral position abroad.

3 The Neuro-Genetic Approach

A peculiar aspect of our approach is that we do not use BP as some genetic operator, as is the case in some related work [2], the EA optimizes both the topology and the weights of the networks; BP is optionally used to decode a *genotype* into a *phenotype* NN. Accordingly, it is the genotype which undergoes the genetic operators and which reproduces itself, whereas the phenotype is used *only* for calculating the genotype's fitness. Concerning the kind of neural nework model, we restrict our attention to Multi-Layer Perceptrons (MLPs), a specific subset of the feedforwards NNs with a layer of input neurons, a layer of one or more output neurons and zero or more "hidden" (i.e., internal) layers of neurons in between; neurons in a layer can take inputs from the previous layer only.

3.1 The Evolutionary Algorithm

A key objective of our work is the design and the evolution of a suitable NN architecture. Some problem-specific parameters of the algorithm are the cost α of a neuron and β of a synapsis used to establish a parsimony criterion for the network architecture; a *bp* parameter, which enables the use of BP if set to 1, and other parameters like probability values used to define topology, weight distribution and *ad hoc* genetic operators.

The overall evolutionary process can be described by the following pseudo-code:

1. Initialize the population, either by generating new random individuals or by loading a previously saved population.
2. Create for each genotype the corresponding MLP, and calculate its mean square error (mse), its cost and its fitness values.
3. Save the best individual as the best-so-far individual.
4. While not termination condition do
   (a) Apply genetic operators to each network.
   (b) Decode each new genotype into the corresponding network.
   (c) Compute the fitness value for each network.
   (d) Save statistics.

The initial population is seeded with random networks initialized with different hidden layer sizes, using two exponential distributions to determine the number of hidden layers and neurons for each individual, and a normal distribution to determine the weights and bias values. For all weights matrices and bias we also define matrices of variance, that will be applied in conjunction with evolution strategies in order to perturb network weights and bias values. Variance matrices will be initialized with matrices of all ones. In both cases, the maximum size and number of the hidden layers is neither pre-determined, nor bounded, even though the fitness function may penalize large networks.

3.2 Representation

Each individual is encoded in a structure that maintains basic information on the net as illustrated in Table 1. The values of all these parameters are affected by

Table 1. Individual Encoding.

Element	Description
l	Lenght of the topology string, corresponding to the number of layers.
topology	String of integer values that represent the number of neurons in each layer.
$\mathbf{W}^{(0)}$	Weights matrix of the input layer neurons of the network.
$\mathbf{Var}^{(0)}$	Variance matrix of the input layer neurons of the network.
$\mathbf{W}^{(i)}$	Weights matrix for the ith layer, $i = 1, \ldots, l$.
$\mathbf{Var}^{(i)}$	Variance matrix for the ith layer, $i = 1, \ldots, l$.
b_{ij}	Bias of the jth neuron in the ith layer.
$\mathbf{Var}b_{ij}$	Variance of the bias of the jth neuron in the ith layer.

the genetic operators during evolution, in order to perform incremental (adding hidden neurons or hidden layers) and decremental (pruning hidden neurons or hidden layers) learning. Note that the use of the *bp* parameter defines two different types of genetic encoding: if no BP-based network training is employed, we have a *direct encoding*, in which the network structure is directly translated into the corresponding phenotype; otherwise, we have an *indirect encoding* of networks, where the phenotype is obtained by the training of an initial (embryonic) network using BP.

3.3 Fitness

We adopt the convention that a lower fitness means a better NN, mapping directly to the objective function of our problem in a cost minimization problem. Therefore, the fitness is proportional to the value of the mse and to the cost of the considered network. It is defined as

$$f = \lambda kc + (1 - \lambda)\text{mse}, \tag{1}$$

where $\lambda \in [0, 1]$ is a parameter which specifies the desired trade-off between network cost and accuracy, k is a constant for scaling the cost and the mse of the network to a comparable scale, and c is the overall cost of the considered network, defined as

$$c = \alpha N_{hn} + \beta N_{syn}, \tag{2}$$

where N_{hn} is the number of hidden neurons, and N_{syn} is the number of synapses. The mse depends on the *Activation Function*, that calculates all the output values for each single layer of the neural network. In this work we use the *Sigmoid Transfer Function*. The rationale behind introducing a cost term in the objective function is that we seek for networks which use a reasonable amount of resources (neurons and synapses), which makes sense in particular when a hardware implementation is envisaged. To be more precise, two fitness values are actually calculated for each individual: the fitness f, used by the selection operator, and a test fitness $\hat{f}$. Following the commonly accepted practice of machine learning, in our approach, the problem data are partitioned into three sets:

- *training set*, used to train the network;
- *test set*, used to decide when to stop the training and avoid overfitting;
- *validation set*, used to test the generalization capabilities of a network.

It is important to stress that no thikness is given to these dataset definitions in the literature. Now, $\hat{f}$ is calculated according to Equation 1 by using the mse over the test set. When BP is used, i.e., if $bp = 1$, $f = \hat{f}$; otherwise ($bp = 0$), f is calculated according to Equation 1 by using the mse over the training and test sets together.

3.4 Genetic Operators

The genetic core of the algorithm is described by the following pseudo-code:

1. Select from the population (of size n) $\lfloor n/2 \rfloor$ individuals by truncation and create a new population of size n with copies of the selected individuals.
2. For all individuals in the population:
   (a) Perform crossover.
   (b) Mutate the weights and the topology of the offspring.
   (c) Train the resulting network using the training and test sets if $bp = 1$.
   (d) Calculate f and $\hat{f}$.
   (e) Save the individual with lowest $\hat{f}$ as the best-so-far individual if the $\hat{f}$ of the previously saved best-so-far individual is higher (worse).
3. Save statistics.

The *selection* strategy used by the algorithm is truncation: starting from a population of n individuals, the worst $\lfloor n/2 \rfloor$ (with respect to f) are eliminated. The remaining individuals are duplicated in order to replace those eliminated. Finally, the population is randomly permuted.

As indicated in [7] there has been some debate in the literature about the opportunity of applying crossover to ANN evolution, based on disruptive effects

that it could make into neural model. In this approach two ideas of crossover are independently implemented: the first is a kind of single-point crossover with different cutting points; the second implements a kind of vertical crossover, defining a *merge-operator* between the topologies and weight matrices of two parents in order to create the offspring.

Single-Point Crossover: it is a kind of single-point crossover, where cutting points are extracted for each parent, since the genotype lenght is variable. Furthermore, the genotypes can be cut only in 'meaningful' places, i.e., only between one layer and the next: this means that a new weight matrix has to be created to connect the two layers at the crossover point in the offspring. These new weight matrices are initialized from a normal distribution, while corresponding variance matrices are setted to matrices of all ones. This kind of crossover is shown in Figure 1.

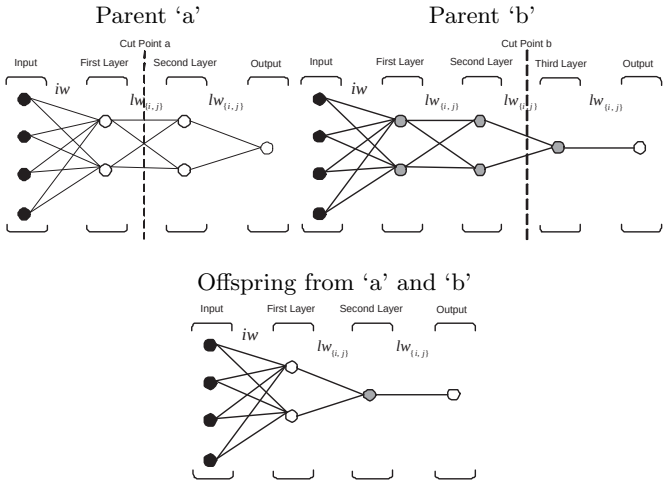


Fig. 1. Single-Point Crossover Representation.

Vertical Crossover: the second type of crossover operator is a kind of 'vertical' crossover and it is implemented as shown in Figure 2. Once the new population has been created by the *selection* operator, two individual are chosen for coupling and their neural structures are compared. If there are some differences in the topology length l, the hidden layer insertion mutation operator will be applied to the shortest neural topology in order to obtain individuals with the same number of layers. Then a new individual will be created, the child of the two parents selected. The neural structure of the new individual is created by adding the number of neurons in any hidden layer of each parent, excepted for input and output layer (they are the same for each neural network). The new input-weights matrix $\mathbf{W}^{(0)}$ and the relative variance matrix $\mathbf{Var}^{(0)}$ are respectively obtained by appending the matrix of the second parent to the matrix of the

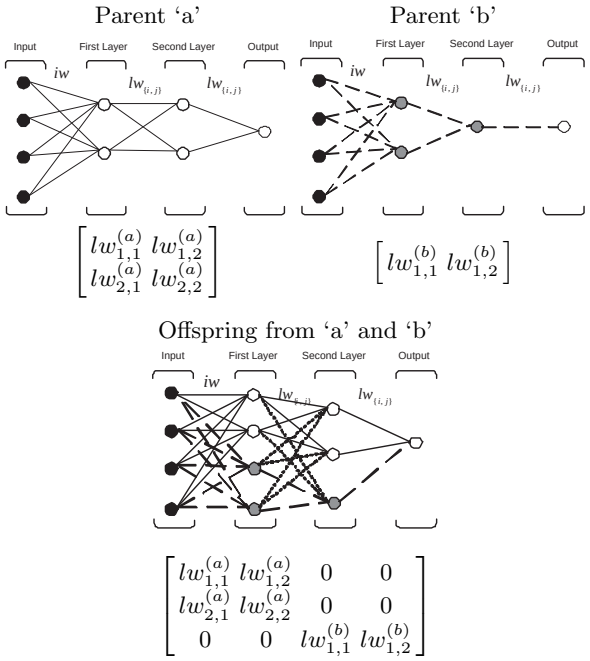


Fig. 2. Merge-Crossover Representation.

first parent. Then, the new weight matrix $\mathbf{W}^{(i)}$ and the corresponding variance matrix $\mathbf{Var}^{(i)}$ for each hidden layer of the new individual are respectively defined as the block diagonal matrix of the matrix of the first parent and the matrix of the second parent. Bias values and corresponding variance matrices of two parents are concatenated in order to obatin the new values for the new biases b_{ij} and variances $\mathbf{Var}(b_{ij})$. All the weights of the inputs to the new output layer will be set to the half of the corresponding weights in the parents. The rationale of this choice is that, if both parents were 'good' networks, they would both supply the appropriate input to the output layer; without halving it, the contribution from the two subnetworks would add and yield an approximately double input to the output layer. Therefore, halving the weights helps to make the operator as little disruptive as possible.

Two types of *mutation* operators are used: a general random perturbation of weights, applied before the BP learning rule, and three *mutation* operators which affect the network architecture. The weight mutation is applied first, followed by the topology mutations, as follows:

1. Weight mutation: all the weight matrices $\mathbf{W}^{(i)}$, $i = 0, \ldots, l$ and the biases are perturbed by using variance matrices and evolutionary strategies applied to the number of synapses of the entire neural network N_{syn}. This mutation is implemented by the following equation:

$$W_j^{(i)} \leftarrow W_j^{(i)} + N(0,1) \cdot \mathrm{Var}_j^{(i)}$$
$$\mathrm{Var}_j^{(i)} \leftarrow \mathrm{Var}_j^{(i)} \cdot e^{\tau' N(0,1) + \tau N(0,1)}$$

with $\tau' = \frac{1}{\sqrt{2N_{syn}}}$ and $\tau = \frac{1}{\sqrt{2\sqrt{N_{syn}}}}$.
After this perturbation has been applied, neurons whose contribution to the network output is negligible are eliminated, basing on threshold. In this work two different kinds of threshold are considered and alternatively applied to the weight perturbation. The first is a fixed threshold, simply defining a ϵ parameter, setted before execution. The following pseudocode is implemented in mutation operator by applying a comparison between that parameter and all weight matrices values.

for $i = 1$ **to** $l - 1$ **do**
if $N_i > 1$
  for $j = 1$ **to** N_i **do**
    if $||W_j^{(i)}|| < \epsilon$
      delete the jth neuron

where N_i is the number of neurons in the ith layer, and $W_j^{(i)}$ is the jth column of matrix $\mathbf{W}^{(i)}$. This solution presents the drawback that the fixed threshold value ϵ could be difficult to set for different real-world application. A solution to this problem has been implemented in this approach by defining a variable threshold. In this case the new threshold is defined, depending on a norm (in this case L_∞) of the weight vector for each node, and the relevant average and standard deviation of the norms of the considered layer. This task is carried out according to the following pseudo-code:

for $i = 1$ **to** $l - 1$ **do**
if $N_i > 1$
  for $j = 1$ **to** N_i **do**
    if $||W_j^{(i)}|| < (\mathrm{avg}_k(||W_k^{(i)}||) - r \cdot \mathrm{stdev}_k(||W_k^{(i)}||))$
      delete the jth neuron

where N_i is the number of neurons in the ith layer, $W_j^{(i)}$ is the jth column of matrix $\mathbf{W}^{(i)}$, and r is a parameter which allows the user to tune how many standard deviations below the layer average the contribution of a neuron must be before it is deleted. In this solution the settings of r parameter is only for tuning standard deviation and corresponding variances are not so invasive in mutation.

2. Topology mutations: these operators affect the network structure (i.e., the number of neurons in each layer and the number of hidden layers). In particular, three mutations can occur:

(a) Insertion of one hidden layer: with probability p^{+}_{layer}, a hidden layer i is randomly selected and a new hidden layer $i-1$ with the same number of neurons is inserted before it, with $\mathbf{W}^{(i-1)} = \mathbf{I}(N_i)$ and $b_{i-1,j} = b_{ij}$, with $j = 1, \ldots, N_i = N_{i-1}$, where $\mathbf{I}(N_i)$ is the $N_i \times N_i$ identity matrix.

(b) Deletion of one hidden layer: with probability p^{-}_{layer}, a hidden layer i is randomly selected; if the network has at least two layers and layer i has exactly one neuron, layer i is removed and the connections between the $(i-1)$th layer and the $(i+1)$th layer (to become the ith layer) are rewired as follows:

$$\mathbf{W}'^{(i-1)} \leftarrow \mathbf{W}'^{(i-1)} \cdot \mathbf{W}'^{(i)}.$$

Since $\mathbf{W}^{(i-1)}$ is a row vector and $\mathbf{W}^{(i)}$ is a column vector, the result of the product of their transposes is a $N_{i+1} \times N_{i-1}$ matrix.

(c) Insertion of a neuron: with probability p^{+}neuron, the jth neuron in the hidden layer i is randomly selected for duplication. A copy of it is inserted into the same layer i as the (N_i+1)th neuron; the weight matrices are then updated as follows:

i. a new row is appended to $\mathbf{W}^{(i-1)}$, which is a copy of jth row of $\mathbf{W}^{(i-1)}$;

ii. a new column $W^{(i)}_{N_i+1}$ is appended to $\mathbf{W}^{(i)}$ where $W^{(i)}_{j} \leftarrow \frac{1}{2} W^{(i)}_{j}$ and $W^{(i)}_{N_i+1} \leftarrow W^{(i)}_{j}$.

The rationale for halving the output weights from both the jth neuron and its copy is that, by doing so, the overall network behavior remains unchanged, i.e., this kind of mutation is neutral.

All three topology mutation operators are designed so as to minimize their impact on the behavior of the network; in other words, they are designed to be as little disruptive (and as much neutral) as possible. Table 2 lists all the parameters of the algorithm, and specifies the default values that they assume in this work.

4 Fault-Diagnosis Application

Every industrial application requires a suitable monitoring system for its processes in order to identify any decrease in efficiency and any loss. A generic information from an electric power measurement system, which monitors the power consumption of an electric component, can be usefully exploited for sensorless monitoring of an AC motor drive, whose input terminals are the only accessible points to it. The proposed approach involves the analysis of the signal – the load current – through wavelet series decomposition. The decomposition results in a set of coefficients, each carrying local time-frequency information. An orthogonal basis function is chosen, thus avoiding redundancy of information and allowing for easy computation. This problem has been already approached with a neuro-fuzzy network, whose structure was defined *a priori*, trained with BP.

Table 2. Parameters of the Algorithm.

Symbol	Meaning	Default Value
n	Population size	60
seed	Previously saved population	none
T	Maximum time for simulation	∞
G	Maximum number of generations	∞
bp	Backpropagation selection	0/1
p^{+}_{layer}	Probability to insert a hidden layer	0.1
p^{-}_{layer}	Probability to delete a hidden layer	0.1
p^{+}_{neuron}	Probability to insert a neuron in a hidden layer	0.05
p_{cross}	Probability to crossover	0
r	Parameter for use in weight mutation to decide neuron elimination	1.5
h	Mean for the exponential distribution	3
N_{in}	Number of network inputs	*
N_{out}	Number of network outputs	*
α	Cost of single neuron	2
β	Cost of single synapsis	4
λ	Desired tradeoff between network cost and accuracy	0.5
k	Constant for scaling cost and mse in the same range	10^{-5}

*) depends on the problem.

5 Experiments and Results

In our approach, both the network structure (topology) and the weights have to be determined through evolution at the same time. We have to look for networks with 8 input attributes, corresponding to the wavelet coefficients for each level of wavelet decomposition, and 1 output, which we shall interpret as an estimate of the fault probability: zero thus means a fault is not expected at all, whereas one is the certainty that a fault is about to occur. The data used for learning have been obtained from a Virtual Test Bed (VTB) model simulator of a real engine. Several settings of five parameters (namely, bp, the population size n, and the three mutation probabilities relevant to structural mutation, $p^{\pm}_{\text{layer}}$ and p^{+}_{neuron}) have been explored in order to assess the robustness of the approach and to determine an optimal set-up: the results are summarized in Table 3. All runs were allocated the same fixed amount of neural network executions, to allow for a fair comparison between the cases with and without backpropagation. Ten runs were executed for each setting, of which we report the average and standard deviation for the best solutions found. A first comment can be made regarding the size of the population. In most cases we observe that the solutions found with a larger population are better than those found with a smaller population. With $bp = 1$, 15 settings out of 27 give better results with $n = 60$, while with $bp = 0$, 19 settings out of 27 give better results with the larger pupulation. In addition, we observe that, for this problem, there is a clear tendency for the runs

using backpropagation ($bp = 1$) to consistently obtain better quality solutions (lower average fitness and smaller standard deviation).

Table 3. Experimental results for the engine fault diagnosis problem.

setting	p^+_{layer}	p^-_{layer}	p^+_{neuron}	$bp = 1$				$bp = 0$			
				$n = 30$		$n = 60$		$n = 30$		$n = 60$	
				avg	stdev	avg	stdev	avg	stdev	avg	stdev
1	0.05	0.05	0.05	0.11114	0.0070719	0.106	0.0027268	0.14578	0.013878	0.13911	0.0086825
2	0.05	0.05	0.1	0.10676	0.003172	0.10606	0.0029861	0.1434	0.011187	0.13573	0.013579
3	0.05	0.05	0.2	0.10776	0.0066295	0.10513	0.0044829	0.13977	0.014003	0.13574	0.010239
4	0.05	0.1	0.05	0.10974	0.0076066	0.10339	0.0036281	0.14713	0.0095158	0.13559	0.011214
5	0.05	0.1	0.1	0.1079	0.0067423	0.10696	0.0050514	0.14877	0.010932	0.13759	0.014255
6	0.05	0.1	0.2	0.10595	0.0035799	0.10634	0.0058783	0.14321	0.0095505	0.1309	0.012189
7	0.05	0.2	0.05	0.10332	0.0051391	0.10423	0.0030827	0.14304	0.014855	0.13855	0.0089141
8	0.05	0.2	0.1	0.10723	0.0097194	0.10496	0.0050782	0.13495	0.015099	0.13655	0.0079848
9	0.05	0.2	0.2	0.10684	0.007072	0.1033	0.0031087	0.14613	0.010733	0.14165	0.013385
10	0.1	0.05	0.05	0.10637	0.0041459	0.10552	0.0031851	0.13939	0.013532	0.13473	0.0085242
11	0.1	0.05	0.1	0.10579	0.0050796	0.10322	0.0035797	0.13781	0.0094961	0.13991	0.012132
12	0.1	0.05	0.2	0.10635	0.0049606	0.10642	0.0042313	0.13692	0.017408	0.13143	0.012919
13	0.1	0.1	0.05	0.10592	0.0065002	0.10889	0.0038811	0.13348	0.009155	0.1363	0.013102
14	0.1	0.1	0.1	0.10814	0.0064667	0.10719	0.0054168	0.13785	0.013465	0.13836	0.0094587
15	0.1	0.1	0.2	0.10851	0.0051502	0.11015	0.0055841	0.14076	0.01551	0.13994	0.011786
16	0.1	0.2	0.05	0.10267	0.005589	0.10318	0.0085395	0.1396	0.0098416	0.13719	0.016372
17	0.1	0.2	0.1	0.10644	0.0045312	0.10431	0.0041649	0.13597	0.012948	0.14091	0.014344
18	0.1	0.2	0.2	0.10428	0.004367	0.10613	0.0052063	0.14049	0.013535	0.13665	0.011426
19	0.2	0.05	0.05	0.10985	0.0059448	0.10757	0.0045103	0.13486	0.0079435	0.14068	0.013874
20	0.2	0.05	0.1	0.10593	0.0048254	0.10643	0.0056578	0.13536	0.0112	0.12998	0.013489
21	0.2	0.05	0.2	0.10714	0.0043861	0.10884	0.0049295	0.13328	0.0087402	0.1314	0.0088282
22	0.2	0.1	0.05	0.10441	0.0051143	0.10789	0.0046945	0.13693	0.0096481	0.13456	0.012431
23	0.2	0.1	0.1	0.1035	0.0030094	0.1083	0.0031669	0.13771	0.015971	0.13939	0.0092643
24	0.2	0.1	0.2	0.10722	0.0048851	0.1069	0.0050953	0.13204	0.010325	0.1378	0.01028
25	0.2	0.2	0.05	0.10285	0.0039064	0.1079	0.0045474	0.14062	0.012129	0.14005	0.011195
26	0.2	0.2	0.1	0.10785	0.008699	0.10768	0.0061734	0.14171	0.008802	0.13877	0.0094973
27	0.2	0.2	0.2	0.10694	0.0052523	0.10652	0.0050768	0.14216	0.015659	0.13965	0.015732

We can observe that there is a clear trend for the runs using backpropagation ($bp = 1$) to consistently obtain better quality solutions (lower average fitness and smaller standard deviation). In all cases, the relative standard deviation is sufficiently small to guarantee finding a good solution in a few runs. A comparison with the results obtained in [3] for a hand-crafted neuro-fuzzy network did not reveal any significant difference. This is an extremely positive outcome, given the expert time and effort spent in hand-crafting the neuro-fuzzy network, as compared to the practically null effort required to set up our experiments. On the other hand, the amount of required computing resources was substantially greater with our approach.

6 Application to Brain-Wave Analysis

The second application consider a binary classification of brain waves in the context of Brain Computer Interfaces (BCI). BCI represent a new communication

option for those suffering from neuromuscular impairment that prevents them from using conventional input devices, such as mouses, keyboards, joysticks, etc. This new approach has been developing quickly during the last few years, thanks to the increase of computational power and the availability of new algorithms for signal processing that can be used to analyze brain waves. BCI systems appear as a possible and sometimes unique mode of communication for people with severe neuromuscular disorders like spinal cord injury or cerebral paralysis. Exploiting the residual functions of the brain, may allow those patients to communicate. The human brain has an intense chemical and electrical activity, partially characterized by peculiar electrical patterns, which occur at specific times and at well-localized brain sites. All of that is observable with a certain level of repeatability under well-defined environmental conditions. These simple physiological issues can lead to the development of new communication systems.

6.1 Problem Description

One of the most utilized electrical activities of the brain for BCI is the so-called P300 Evoked Potential. This wave is a late-appearing component of an Event Related Potential (ERP) which can be auditory, visual or somatosensory. It has a latency of about 300 ms and is elicited by rare or significant stimuli, when these are interspersed with frequent or routine stimuli. Its amplitude is strongly related to the unpredictability of the stimulus: the more unexpected the stimulus, the higher the amplitude. This particular wave has been used to make a subject chose between different stimuli [5, 4]. The general idea of Donchin's solution is that the patient is able to generate this signal without any training. This is due to the fact that the P300 is the brains response to an unexpected or surprising event and is generated naturally. Donchin developed a BCI system able to detect an elicited P300 by signal averaging techniques (to reduce the noise) and used a specific method to speed up the overall performance. Donchin's idea has been adopted and further developed by Beverina and colleagues of ST Microelectronics [1]. We have applied the neuro-genetic approach described in Section 3 to the same dataset of P300 evoked potential used by Beverina and colleagues for their approach on brain signal analysis based on support vector machines.

6.2 Experiments and Results

The dataset provided by Beverina and colleagues consists of 700 negative cases and 295 positive cases. The feature are based on wavelets, morphological criteria and power in different time windows, for a total of 78 real-valued input attributes and 1 binary output attribute, indicating the class (positive or negative) of the relevant case. A positive case is one for which the response to the stimulus is correct; a negative case is one for which the response is incorrect. In order to create a balanced dataset of the same cardinality as the one used by Beverina and colleagues, for each run of the evolutionary algorithm we extract 218 positive cases from the 295 positive cases of the original set, and 218 negative cases from the 700 negative cases of the original set, to create a 436

bp	training				test			
	false positives		false negatives		false positives		false negatives	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
0	93.28	38.668	86.14	38.289	7.62	3.9817	7.39	3.9026
1	29.42	14.329	36.47	12.716	1.96	1.4697	2.07	1.4924

Table 4. Error rates of the best solutions found by the neuro-genetic approach with and without the use of backpropagation, averaged over 100 runs.

case training dataset; for each run, we also create a 40 case test set by randomly extracting 20 positive cases and 20 negative cases from the remainder of the original dataset, so that there is no overlap between the training and the test sets. This is the same protocol followed by Beverina and colleagues. For each run of the evolutionary algorithm we allow up to 25,000 network evaluations (i.e., simulations of the network on the whole training set), including those performed by the backpropagation algorithm. 100 runs of the neuro-genetic approach with different parameters settings were executed with $bp = 0$ and $bp = 1$, i.e., both without and with backpropagation.

Due to the way the training set and the test set are used, it is not surprising that error rates on the test sets look better than error rates on the training sets. That happens because, in the case of $bp = 1$, the performance of a network on the test set is used to calculate its fitness, which is used by the evolutionary algorithm to perform selection. Therefore, it is only networks whose performance on the test set is better than average which are selected for reproduction. The best solution has been found by the algorithm using backpropagation and is a multi-layer perceptron with one hidden layer with 4 neurons, which gives 22 false positives and 29 false negatives on the training set, while it commits no classification error on the test set. However, the results obtained by the neuro-genetic approach without any specific tuning of the parameters, appear promising in comparison with results obtained by Beverina and colleagues with support vector machines [9]. To provide a reference, the average number of false positives obtained by Beverina and colleagues with support vector machines are 9.62 on the training set and 3.26 on the test set, whereas the number of false negatives are 21.34 on the training set and 4.45 on the test set [9].

7 Achievements and Feedback

We illustrated an evolutionary approach to the joint design of neural network structure and weights which can take advantage of BP as a specialized decoder. The results obtained on the fault diagnosis application compared well against alternative approaches based on the conventional training of a predefined neuro-fuzzy network with BP and they shown how the algorithm is somewhat robust w.r.t. the setting of its parameters, i.e., its performance is little sensitive of the fine tuning of the parameters. In the second application of Brain-Wave Analy-

sis, the comparison with a mature approach, based on support vector machines [1], shows that our approach has some potential, even though, unsurprisingly, it does not attain the same levels of accuracy. An important issue still unresolved regards the the efficiency and the robustness of this approach even when input data are affected by uncertainty depending on errors introduced by some measurement instrumentations. From this Doctoral Consortium I hope to obtain valuable feedback about my research work, and suggestions for future directions. It could be interesting and useful to know what is the relevance of these techniques in real problems with respect to other techniques and what could be further improvements to this approach, with particular attention to genetic operators and parameter's settings here implemented. I also expect to make contacts with other researchers with similar interests in this field.

References

1. F. Beverina, G. Palmas, S.Silvoni, F. Piccione, and S. Giove. User adaptive bcis: Ssvep and p300 based interfaces. *PsychNology Journal*, 1(4):331–354, 2003.
2. P. A. Castillo, J. Carpio, J. J. Merelo, A. Prieto, V. Rivas, and G. Romero. Evolving multilayer perceptrons. *Neural Processing Letters*, 12(2):115–127, 2000.
3. L. Cristaldi, M. Lazzaroni, A. Monti, and F. Ponci. A neurofuzzy application for ac motor drives monitoring system. *IEEE Transactions on Instrumentation and Measurement*, 53(4):1020–1027, August 2004.
4. E. Donchin, K.M. Spencer, and R. Wijesinghe. The mental prosthesis: assessing the speed of a p300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, 8(2):174–179, June 2000.
5. L.A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalogr Clin Neurophysiol.*, 70(6):510–523, December 1988.
6. F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):54–65, January 2003.
7. V. Maniezzo. Genetic evolution fo the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, January 1994.
8. P. Mordaunt and A.M.S. Zalzala. Towards an evolutionary neural network for gait analysis. In *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 2, pages 1238–1243, 2002.
9. Giorgio Palmas. Personal communication, November 2005.
10. D. E. Rumelhart, J. L. McClelland, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
11. K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
12. B. Yang, X.H. Su, and Y.D. Wang. Bp neural network optimization based on an improved genetic algorithm. In *Proceedings of the IEEE First International Conference on Machine Learning and Cybernetics*, pages 64–68, November 2002.
13. X. Yao. Evolving artificial neural networks. In *Proceedings on IEEE*, pages 1423–1447, 1999.
14. X. Yao and Y.Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, May 1997.

Evolution of Small-World Networks as a support for Cellular Automata Computation

Christian Darabos

INFORGE - Information Systems Dept., University of Lausanne, Switzerland
{Christian.Darabos, Marco.Tomassini}@unil.ch

Abstract. We study an extension of cellular automata to arbitrary interconnection topologies for the majority and the synchronization problems. By using an evolutionary algorithm, we show that small-world type network topologies consistently evolve from regular and random structures without being designed beforehand. These topologies have better performance than regular lattice structures and are easier to evolve, which could explain in part their ubiquity. Moreover, we show experimentally these graph topologies are much more robust in the face of random faults than lattice structures for these problems.

1 Introduction to the Research Area

In recent years there has been substantial research activity in the study of networks. These latter, which can be formally described by the tools of graph theory, are a central model for the description of many phenomena of scientific, social and technological interest. Typical examples include the Internet, the World Wide Web, social acquaintances, electric power networks, neural networks, and many others. The pioneering studies of Watts and Strogatz [18, 17] have been instrumental in initiating the movement, and they have been followed by many others in the subsequent years. Their key observation was that most real networks, both in the biological world as well as man-made structures, have mathematical properties that set them apart from regular lattices and from random graphs, which were the two main topologies that had been studied until then. In particular, they introduced the concept of *small-world networks*, in which most pairs of vertices seem to be connected by a short path through the network. The existence of short paths between any pair of nodes has been found in networks as diverse as the Internet, airline routes, neural networks, metabolic networks, among others. The presence of short paths is also a characteristic of random graphs, but what sets these real networks apart is a larger *clustering coefficient* than that of random graphs having a comparable number of nodes and links. The clustering coefficient roughly represents the probability that two nodes that are neighbors of a third one, are also neighbors of each other, which means that there is more local structure in these networks than in plain random graphs.

Another type of networks which also differs from both the regular and the random ones, the *scale-free* graphs, is more typical of real-world networks; they are fully described in [1], which is a very readable and complete account of modern network theory.

The topological structure of a network has a marked influence on the processes that may take place on it. Regular and random networks have been thoroughly studied from this point of view in many disciplines. In computer science, for instance, variously connected networks of processors have been used in parallel and distributed computing [9], while lattices and random networks of simple automata have also received a great deal of attention [4, 7]. On the other hand, due to their novelty, there are very few studies of the computational properties of small-world networks. One notable exception is Watts' book [17] in which cellular automata (CAs) computation on small-world networks is examined in detail. Another recent study for CAs on small worlds appears in [12]. However, one important piece of the puzzle is missing in these works. How such networks could arise in the first place? In the works cited above the graphs are generated by a prescribed algorithm. In our opinion, the question of how these networks could emerge in the fist place is an interesting yet unanswered one. In contrast, for scale-free networks there exist several models that account for their genesis [1], although all of them make some hypothesis as to how new nodes become linked to existing ones. For example, *preferential attachment* posits that the likelihood of connecting to a node depends on the node's degree: high-degree nodes are more likely to attract other nodes.

2 Research and Study

2.1 Goals

The work presented here is an appetizer to a more thorough study of the behavior of CAs on different network topologies. The long term ambition is to simulate and study real-world like interactions whilst keeping the advantages, flexibility and simplicity of CAs. Namely mostly biological phenomena that take on more complex network structures, such as protein-cell interactions.

The need to study the evolution and properties of complex networks of dynamical processes is rapidly becoming a key factor in order to be able to master the complexity of social, biological, and technological systems. In this project we will focus on metabolic networks as an important case study. Using networks of cellular automata as a model, our aim is to develop tools that will allow us to investigate the dynamical properties of such real networks. The analysis of simulation data will be useful to forecast the behavior of such networks under the influence of perturbations, such as those present in diseases and in the use of pharmaceutical drugs. However, the models and the tools developed should prove of more general value, since all these networks share a number of structural properties which, in turn, influence their behavior.

2.2 Current Status

Current studies are toy examples of more complex behavior. We have conducted three main studies along the lines proposed in the introduction (Section 1) of this document that follow from Watt's past studies [17].

1. An attempt to find how and what kind of networks might come to be selected consisted of letting an artificial evolutionary process find "good" network structures according to a predefined fitness measure, leaving the fine details of the wiring to the evolutionary algorithm. We take as a prototypical problem the *majority classification* problem which is the same that Watts discusses in [17] as a useful first step. This will also allow us to compare the products of artificial evolution with Watts' results.
2. We have thoroughly studied the performance of the above described evolved networks on both the *majority classification* and *synchronization* task and compared to Watt's and other's work.
3. We have compared the robustness of our newly evolved structures to the artificially evolved rules CAs in [10, 11], using the *Hamming* distance as a measurement for the capacity of both to resist to random transient faults.

2.3 Future Planning

Future work include:

- A study of the co-evolution of the structure of our CAs and the nodes update rules of our generalized CAs.
- A comparison of the results obtained on evolved *Small-World* networks and different, better known, structures, notably hand-constructed *Scale-Free* structures using the Barabási-Albert [1] model.
- Further down the road our work will drift towards biological networks as more concrete study cases will be required. At that time, we hope to have built a team around this project composed of both computer scientists and biologists and try to make everyone proficient in both fields.
- On a more personal note, the ways I envisage the future are multiple. Having had a business experience prior to my Ph.D. studies, I could see myself going back to business in an R&D oriented company or department. On the other hand, I am currently developing a strong taste for academic research and depending on the results of this project and the opportunities ahead of me, I might want to carry out more research on the subject or on related ones. The important factor for me is to be able to relocate, move abroad in a completely new and exciting environment.

2.4 Study

Ph.D. studies at the University of Lausanne are conducted over four years of research plus a possible fifth one for the redaction and defense of the work.

Starting in October 2005, an 18-month Bologne-like doctoral school has been added to the beginning of the cursus. This goes along a study plan put together by the student and his supervisor and include reading of articles, participation to seminars, summer schools and courses and publications. I am currently in my first year of the Ph.D. cursus. In addition to the academic part, all Ph.D. students have a didactical workload as teaching assistants for their supervisor and the classes they teach.

3 Results

3.1 Methodology

CAs are dynamical systems in which space and time are discrete. A standard CA consists of an array of cells, each of which can be in one of a finite number of possible states. Here we will only consider boolean automata for which the cellular state $s \in B = \{0, 1\}$. The regular cellular array (lattice) is d-dimensional, where $d = 1, 2, 3$ is used in practice. In one-dimensional lattices a cell is connected to r local neighbors (cells) on either side, where r is referred to as the *radius* (thus, each cell has $2r + 1$ neighbors, including itself).

CAs are updated synchronously in discrete time steps, according to a local, identical rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells, including the cell itself:

$$s_i^{t+1} = f(s_{i-r}^t ..., s_i^t, ...s_{i+r}^t), \qquad f : B^{2r+1} \rightarrow B$$

where s_i^t denotes the value of site i at time t, $f(.)$ represents the local transition rule, r is the CA radius, and B is the binary alphabet. The term *configuration* refers to an assignment of 1s and 0s to all the cells at a given time step. It can be described by $\mathbf{s}^t = (s_0^t, s_1^t, \ldots, s_{N-1}^t)$, where N is the lattice size. Often one-dimensional CAs have periodic boundary conditions $s_{N+i}^t = s_i^t$. Configurations evolve in time according to a global update rule Ψ which is the result of applying f in parallel to all the cells $\mathbf{s}^{t+1} = \Psi(\mathbf{s}^t)$.

Here we will consider an extension of the concept of CA in which the rule is the same on each node but nodes can be connected in any way, that is, the topological structures are general graphs, provided the graph is connected and self and multiple links are disallowed.

The majority task is a prototypical distributed computational task for CAs, and can be contrasted with the well-studied tasks known as *Byzantine agreement* and *consensus* in the distributed computing litterature. For a finite CA of size N it is defined as follows. Let ρ^0 be the fraction of 1s in the initial configuration (IC) $\mathbf{s}^0$. The task is to determine whether ρ^0 is greater than or less than 1/2. If $\rho^0 > 1/2$ then the CA must relax to a fixed-point configuration of all 1s; otherwise it must relax to a fixed-point configuration of all 0s, after a number of time steps of the order of the lattice size N (N is odd to avoid the case $\rho^0 = 0.5$). This computation is trivial for a computer having a central control. However, it is nontrivial for a small radius one-dimensional CA since such a CA can only

transfer information at finite speed relying on local information exclusively, while density is a global property of the configuration of states [10].

It has been shown that the density task cannot be solved perfectly by a uniform, two-state CA with finite radius [8], although a slightly modified version of the task can be shown to admit perfect solution by such an automaton [2].

The *performance* P of a given rule on the majority task is defined as the fraction of correct classifications over 10^4 randomly chosen ICs. The ICs are sampled according to a binomial distribution. Clearly, this distribution is strongly peaked around $\rho^0 = 1/2$, thus making a difficult case for the CA to solve.

Watts [17] studied a general graph version of the density task. Since a CA rule table depends on the number of neighbors, given that a small-world graph may have vertices with different degrees, he considered the simpler problem of fixing the rule and evaluating the performance of small-world graphs on the task. The chosen rule is a variation of the *majority* rule: at each time step, each node will assume the state of the majority of its neighbors in the graph. If the number of neighbors having state 0 is equal to the number of those at 1, then the next state is assigned at random with equal probability. When used in a one-dimensional CA this rule has performance $P \simeq 0$ since it gives rise to stripes of 0s and 1s that cannot mix at the borders. Watts, however, has shown that the performance can be good on other network structures, where "long" links somewhat compensate for the lack of information transmission of the regular lattice case, in spite of the fact that the node degrees are still low. Indeed, Watts constructed with an *ad-hoc* algorithm many networks with performance values $P > 0.8$, while the Mitchell's best rule-evolved lattices with the same average number of neighbors had P around 0.77 [10] and were difficult to obtain.

In a remarkable paper [13], Sipper and Ruppin had already examined the influence of different connectivity patterns on the density task. They studied the co-evolution of network architectures and CA rules, resulting in non-uniform, high-performance networks, while we are dealing with uniform CAs here. Since those were pre-small world years, it is difficult to state what kind of graphs were obtained. However, it was correctly recognized that reducing the average cellular distance has a positive effect on the performance.

Following Watts [17], we will show our results as a function of the parameter ϕ, which is the fraction of edges in a graph that are shortcuts. The range of ϕ is $[0, 1]$, where a value of 0 (no shortcuts) corresponds to a perfect regular lattice, and 1 corresponds to the random graph limit (every link is a shortcut on the average). In between lies the small-world range, with the typical small-world behavior already present for low ϕ values (around 0.01-0.1). For higher ϕ values, the graphs tend to be more random-like.

3.2 Experiments and Results

Artificial Evolution of Small Worlds Evolutionary algorithms have been successfully used for more than ten years to evolve network topologies for artificial neural networks and several techniques are available [15]. As far as the network topology is concerned, the present problem is similar, and we use an

unsophisticated structured EA with the aim of evolving small-world networks for the density and synchronization tasks. Our EA is spatially structured, as this permits a steady diffusion of good solutions in the population due to a less intense selection pressure [6]. The population is arranged on a 20×20 square grid for a total of 400 individuals. Each individual represents a network topology and it is coded as an array of integers denoting vertices, each one of which has a list of the vertices it is connected to, as the graph is undirected. The automaton rule is the generalized majority rule described above for all cases. The termination condition is reached after computing exactly 100 generations.

The fitness of a network of automata in the population is calculated by randomly choosing 100 out of the 2^N possible initial configurations (ICs) with uniform density—i.e. any initial density has the same probability of being selected—and then iterating the automaton on each IC until it converges to a stable state but at mots for $M = 2N$ time steps, where $N = 149$ is the automaton size. The performance at the end of the evolutionary cycle is computed over 10'000 ICs with a density around 0.5, thus the task difficult to solve. Figure 2 shows the performance of the best individual (i.e. the individual with the highest performance) of 50 independent evolutionary runs.

Selection is done locally using a central individual and its north, east, south and west first neighbors in the grid. Binary tournament selection is used with this pool. The winner is then mutated (see below) and evaluated. It replaces the central individual if it has a better fitness.

Mutation is designed to operate on the network topology and works as follows. Each node of an individual is mutated with probability 0.5. If chosen, a vertex (called target vertex) will have an edge either added or removed to a randomly chosen vertex (called destination vertex) with probability 0.5. This will only happen if all the requirements are met (minimum and maximum degree are respected). If the source vertex has already reached its maximum degree and should be added one edge or its minimum degree and should be removed one edge, the mutation will not happen. If the same case happens with the target, another one is randomly chosen. This version of the algorithm does not use recombination operators.

As starting point, we used populations of, on the one hand, slightly perturbed regular one-dimentional radius-two lattices (ring-based individuals) and, on the other hand, random graphs (random-based individuals) with average degree $\langle k \rangle \simeq 4$. Having different initial population types reduces the chances of a biased evolution towards small-world networks.

We see on Fgures 1 that, both in the case of ring-based and random based evolved networks, fitness quickly reaches high levels, while performance, which is a harder measure of the generalization capabilities of the evolved networks on the density task, stays lower and then stabilizes at a level greater than 0.8 (See results in Table 3). The population entropy remains high during all runs, meaning that there is little diversity loss during evolution. Note that the entropy refers to the "genotype" and not to fitness. This is unusual and probably due to the spatial structure of the evolutionary algorithm, which only allows slow diffusion

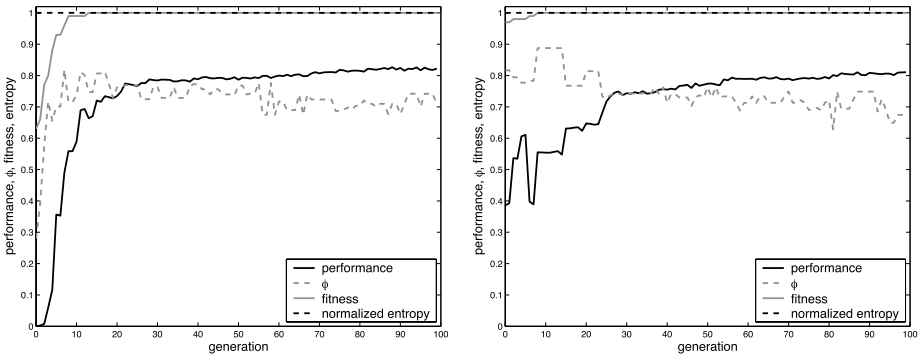


Fig. 1. A typical evolutionary run starting from a perturbed ring population (a) and from a random-based population (b). Graphs are of the evolutionary run of the best individual on that run.

of good individuals through the grid [5]. The ϕ curve is particularly interesting as it permits a direct comparison with Watts' hand-constructed graphs [17].

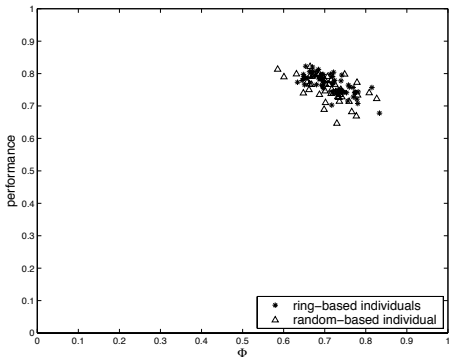


Fig. 2. The 50 best ring-based evolved networks and the 50 best random-based evolved networks of all independent runs

The results fully confirm his measurements, with networks having best performance clustering around ϕ values between 0.6 and 0.8. This is clearly seen in figure 2 where the 50 best ring-based and the 50 best random-based networks of independent evolutionary runs are reported as a function of their ϕ, which is to be compared with figure 7.2, p.190, in [17]. The mean degree $\langle k \rangle$ of the evolved networks is around 7, which compares well with the ring case and Watts's (see Table 3). Therefore, we see that even a simple EA is capable of consistently evolving good performance networks in the small-world range. This is not the case for the standard ring CAs for the majority task, where good rules are noto-

riously difficult to evolve. In fact, while we consistently obtain networks having performance around 0.8 in each evolutionary run, Mitchell *et al.* [10] found that only a small fraction of the runs lead to high-performance CAs. As well, our networks and Watts' reach higher performance: 0.82 against 0.77 for the lattice. Evidently, the original fitness landscape corresponding to the 2^{128} possible ring CAs with radius three is much more difficult to search than the landscape corresponding to all possible graphs with N vertices.

Ring-net	$\langle k \rangle$	C	L	Φ	P	Rand-net	$\langle k \rangle$	C	L	Φ	P
A	7.906	0.053	2.649	0.654	0.823	A	7.798	–	2.695	0.664	0.821
B	7.611	0.053	2.703	0.670	0.820	B	7.543	–	2.736	0.585	0.812
C	7.409	0.048	2.750	0.685	0.813	C	7.355	–	2.729	0.686	0.800
D	7.342	0.049	2.736	0.669	0.807	D	7.422	0.062	2.736	0.631	0.798
E	7.450	0.057	2.730	0.679	0.807	E	6.778	–	2.858	0.748	0.797

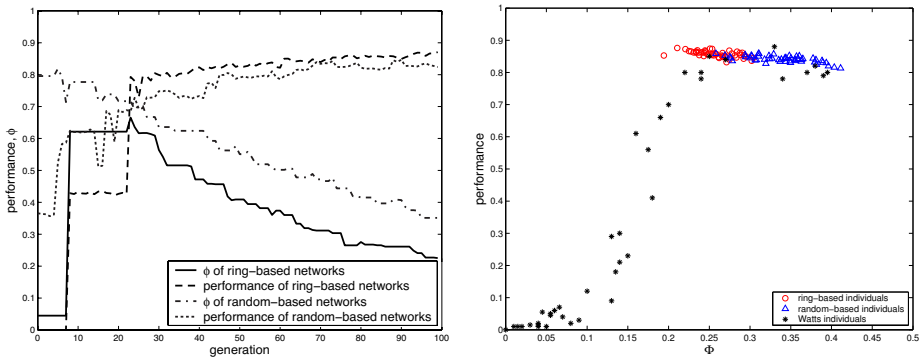
Fig. 3. The ten best evolved networks. $\langle k \rangle$ is the mean node degree. C is the clustering coefficient. L is the characteristic path length. ϕ is the percentage of shortcuts, and P is the network performance on the density task. Left part: ring-based evolved individuals. Right part: random-based evolved individuals (a – in random-based graphs means that the clustering coefficient is not computable since those graphs are allowed to have vertices with a degree smaller than 2).

Reducing the number of shortcuts In the case above, we feel that the value of ϕ around 0.6 is too close to that of random networks. In order to evolve networks in the small-world region, we include a term in the fitness function which, for a given network fitness, favors networks having a lower ϕ value. Obviously, the most general way to solve the problem would be to use multi-objective optimization. However, the simpler technique will prove sufficient for our exploration. The new fitness function is thus:

$$f^{'} = f + (1 - \phi) \times w,$$

where f is the usual CA fitness, w is an empirical weight factor with $w \in [0, 1]$, and $f^{'}$ is the effective fitness. After experimenting with a few different w values, we finally used $w = 0.6$ in the experiments described here, although the precise w value only makes a small difference. At each generation a different set of ICs is generated for each individual.

As depicted in Figure 4, we see that the introduction of a selection pressure favoring networks with small ϕ values is effective in evolving graph-CAs that show high performance. Starting from a population of perturbed rings or random graphs does not make a big difference although, as expected, starting from slightly perturbed rings, which have low ϕ, tends to favor slightly lower ϕ values of the evolved networks. The ϕ values are around 0.3 (see Figure 4b).



(a) (b)

Fig. 4. Density task results. (a): ϕ and performance curves over generations for an initial population of 50 perturbed rings and a population of 50 random graphs. (b): ϕ vs. performance values of the 50 best individuals evolved in the 50 independent evolutionary runs starting from rings and starting from random graphs, and compared to Watts' hand-constructed networks with $\langle k \rangle = 12$.

The average degrees are 11.76 and 9.72 for ring-based and random graph-based respectively. This compares favorably with Watts' hand-constructed networks, where one can see high-performance networks with ϕ around 0.3 but with average degree $\langle k \rangle$ equal to 12. It is clear thus that, to some extent, having more neighbors on the average compensates for the reduced number of shortcut links.

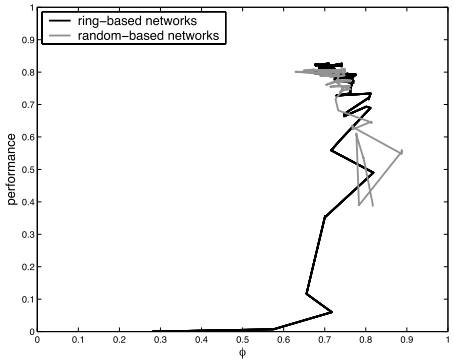


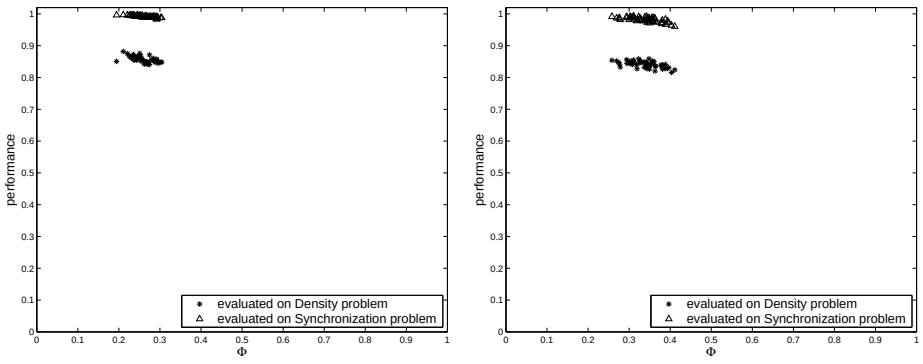
Fig. 5. Evolution of a typical run for both ring and random-based individuals. The curves show performance vs. ϕ, each angle in the curves is a new generation.

Interestingly, Figure 5 contrasts the evolution of ϕ over the generations for both individual kinds. In the case of the ring-based individual, we can see it becoming less organized, while the random-based one is becoming more so. Both then join and cluster around a point of semi-organization and higher perfor-

mance.

Task Flexibility of the Evolved Networks The one-dimensional synchronization task was introduced in [3]. In this task the CA, given an arbitrary initial configuration, must reach a final configuration, within $M \simeq 2N$ time steps, that oscillates between all 0s and all 1s on successive time steps. As with the density task, synchronization also comprises a non-trivial computation for a small-radius CA, and it is thus extremely difficult to come up with CA rules that, when applied synchronously to the whole lattice produce a stable attractor of oscillating all 0s and all 1s configurations.

When changing the rule, networks evolved specifically for the density task yield good performance when used for the synchronization task. As noted by Watts [17], the two tasks are nearly identical and thus this finding is not surprising. Furthermore, this remains true for the whole range of ϕ values for which automata have been evolved or generated by hand.



(a) (b)

Fig. 6. Performance vs ϕ of networks evolved for the density task on both density and synchronization. Ring-based (a) and random graph-based (b) networks.

For instance, Figure 6 shows how networks evolved for the density task using ϕ as a second objective (see previous section) are also well-suited for synchronisation. The opposite is also true: namely, that networks evolved for the synchronization task can be used for solving the density problem.

Robustness in Presence of Random Faults Noisy environments are the rule in the real world. Since these automata networks are toy examples of distributed computing systems, it is interesting and legitimate to ask questions about their fault-tolerance aspects. A network of automata may fail in various ways when random noise is allowed. For instance, the cells may fail temporarily or they may

die altogether; links may be cut, or both things may happen. In this section, we will compare the robustness of standard lattice-CAs and small-world CAs with respect to a specific kind of perturbation, which we call *probabilistic updating*. It is defined as follows: the CA rule may yield the incorrect output bit with probability p_f, and thus the probability of correct functioning will be $(1 - p_f)$. Futhermore, we assume that errors are uncorrelated. This implies that, for a network with N vertices, the probability $P(N, m)$ that m cells (vertices) are faulty at any given time step t is given by

$$P(N, m) = \binom{N}{m} {p_f}^m \, (1 - p_f)^{N-m}$$

i.e. it is binomially distributed. It should be noted that we do not try to correct or compensate for the errors, which is important in engineered system but very complicated and outside our scope. Instead, we focus on the "natural" fault-tolerance and self-recovering capabilities of the systems under study.

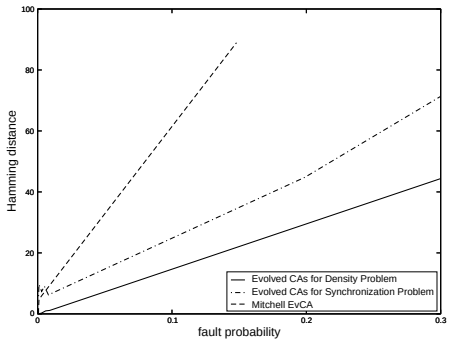


Fig. 7. Hamming distance (normalized over 100) over $2 \times N$ steps (298) vs. fault probability (x-axis) of the topology-evolved networks (black line) and of Mitchell's rule-evolved ring CAs (grey line) for the density problem. The curves are averages over 10^3 distinct initial configurations.

To observe the effects of probabilistic updating on the CA dynamics, two initially identical copies of the system are maintained. One proceeds undisturbed with $p_f = 0$, while the second is submitted to a nonzero probability of fault. We can then measure such things as Hamming distances between unperturbed configurations and those subject to faults, which give information on the spreading of damage (e.g., [14] where the case of synchronous, nonuniform CAs is examined). Figure 7 shows that, for the density task, the amount of disorder is linearly related to the fault probability. For each fault probability, each 50 individual has been evaluated over 10'000 Initial Configurations (ICs). This is an excellent result when compared with ring CAs where already at $p_f = 0.001$ the average

Hamming distance is about 20 [14], and tends to grow exponentially. At $p_f = 0.1$ it saturates at about 95, while it is still only about 20 for the small-world CA.

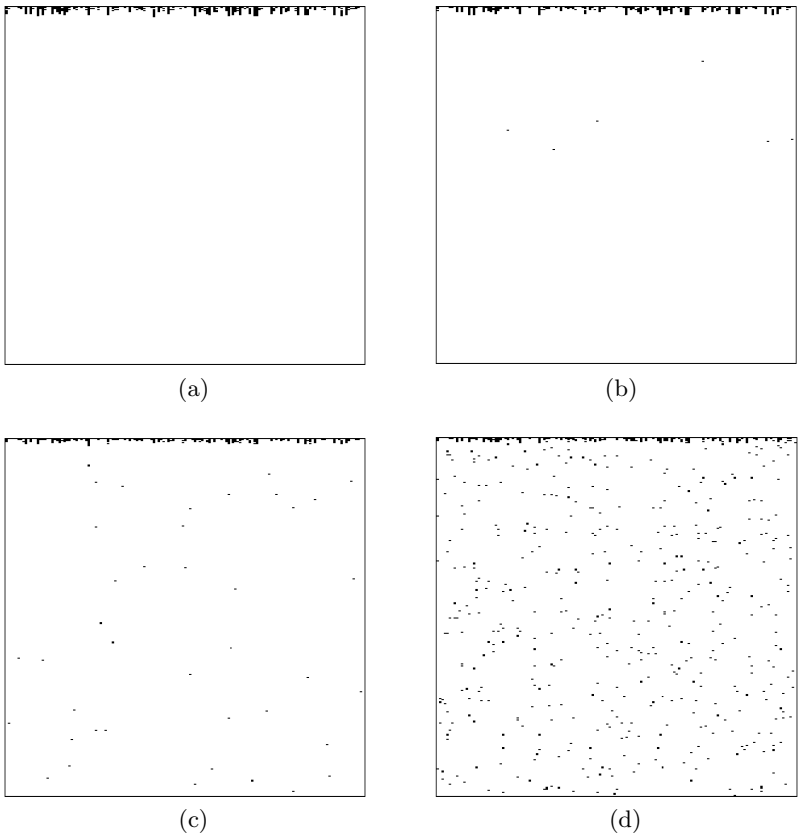


Fig. 8. Typical behavior of a small-world CA over $2 \times N$ (298) time steps for the density task under probabilistic updating. The first line represents the IC (1 is black and 0 is white) and then each line is a time step, whit the state of each cell. The density ρ^0 of the Initial Configuration is 0.490 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

This striking difference is perhaps more intuitively clear by looking at figures 8 and 9. The faulty CA depicted is figure 9 is the best one obtained by artificial evolution in [10, 11] and it is called EvCA here. It is clear that even small amounts of noise are able to perturb the lattice CA so much that either it classifies the configuration incorrectly (c), or it cannot accomplish the task any longer (d) as p_f increases further. For the same amount of noise the behavior of the small-world CA is much more robust and even for $p_f = 0.01$ the fixed point configuration is only slightly altered. Note also that the EvCA configuration has $\rho^0 = 0.416$ whereas the one used in the small-world CA has $\rho^0 = 0.490$, and it

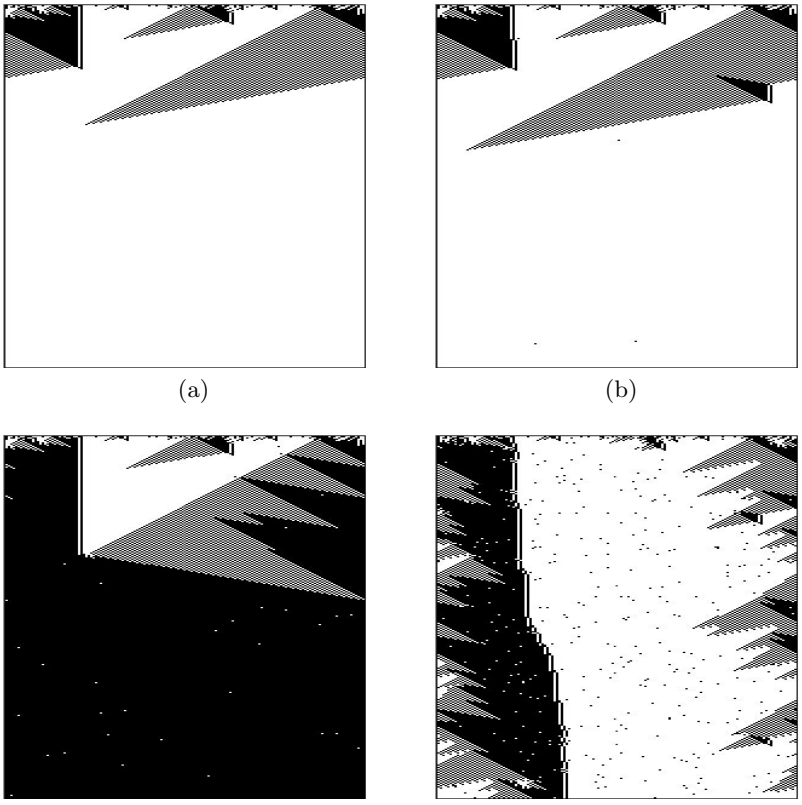


Fig. 9. Typical behavior of EvCA [11] under probabilistic updating on the density task. The density ρ^0 is 0.416 and the probabilities of fault p_f in (a), (b), (c), and (d) are, respectively, 0, 0.0001, 0.001, and 0.01.

is thus more difficult to classify. For completeness, we note that in a previous study [16] we investigated the behavior of evolved *asynchronous* lattice CAs for the density task under probabilistic noise. We found that, while asynchronous CAs are much more fault-tolerant than synchronous ones, their robustness is not as good as that of small-world CAs and their performance is significantly lower.

Looking again at figure 7 we see that the behavior of the synchronization task (dashed line) under noise is poorer. In fact, it is not possible to maintain strict synchronization in the presence of faults. The system manages to limit the damage for low fault probabilities but it goes completely out of phase over $p_f = 0.2$. For higher probabilities the distance stabilizes around 75 (i.e. half of the cells on the average are in the wrong state). In spite of this, the behavior is still much better that the one observed for ring CAs, where at $p_f = 0.01$ the Hamming distance is already about 55 [14], while it is only about 8 in the small-world CA.

4 Achievements

Starting from the work of Watts on small-world automata for the density task, we have used an evolutionary algorithm to evolve networks that have similar computational capabilities. Without including any preconceived design issue, the evolutionary algorithm has been consistently able to find high-performance automata networks in the same class of those constructed by Watts. The power of artificial evolution is seen in the fact that, even starting from a population of completely random graphs, the algorithm finds automata in the same class. This result is an indication that small-world network automata in this range have above average distributed computation capabilities, although we only studied one problem of this type. This is also in line with the results of recent research which point to a pervasive presence of small-world and scale-free graphs in many different domains. Finally we have studied the tolerance of such evolved networks to transient probabilistic faults and showed these structures are highly resistant to random faults thank to the presence of *shortcuts*. These results open perspective for future work. One immediate interesting subject will be to compare the performances and tolerance to random transient faults of *Scale-Free* networks built following the Barabási-Albert [1] model. The next step will probably be a thorough study around the co-evolution of the network structure (as described in this work) and of the rules of each node of the CA.

5 Feedback

What I expect from EvoPhD2006 go along three different lines. From a project perspective, I would like to have a glimpse at what other groups/Ph.D. students working in the field of evolutionary algorithms are producing and in which direction is the whole field going. From an experience perspective, I would like to have a first personal encounter with people who might be interested in, criticize, compare, contrast, challenge or make evolve in any way my current and future work and ideas. From a future perspective, I am hoping to meet possible collaborators and groups with which synergies of different kinds might be possible be in on this project or on related/complementary ones in the years to come.

References

1. R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.
2. M. S. Capcarrère, M. Sipper, and M. Tomassini. Two-state, r=1 cellular automaton that classifies density. *Physical Review Letters*, 77(24):4969–4971, December 1996.
3. R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.
4. M. Garzon. *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*. Springer-Verlag, Berlin, 1995.

5. M. Giacobini, E. Alba, A. Tettamanzi, and M. Tomassini. Modeling selection intensity for toroidal cellular evolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference GECCO'04*. Springer Verlag, Berlin, 2004. To appear.
6. M. Giacobini, M. Tomassini, A. Tettamanzi, and E. Alba. Selection intensity in cellular evolutionary algorithms for regular lattices. *IEEE Transactions on Evolutionary Computation*, 2005. to appear.
7. S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.
8. M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):5148–5150, June 1995.
9. F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.
10. M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361–391, 1994.
11. M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.
12. R. Serra and M. Villani. Perturbing the regular topology of cellular automata: implications for the dynamics. In S. Bandini, B. Chopard, and M. Tomassini, editors, *Cellular Automata, ACRI 2002*, volume 2493 of *Lecture Notes in Computer Science*, pages 168–177. Springer-Verlag, Heidelberg, 2002.
13. M. Sipper and E. Ruppin. Co-evolving architectures for cellular machines. *Physica D*, 99:428–441, 1997.
14. M. Sipper, M. Tomassini, and O. Beuret. Studying probabilistic faults in evolved non-uniform cellular automata. *International Journal of Modern Physics C*, 7(6):923–939, 1996.
15. A. Tettamanzi and M. Tomassini. *Soft Computing: Integrating Evolutionary, Neural and Fuzzy Systems*. Springer, New York, 2001.
16. M. Tomassini and M. Venzi. Evolving robust asynchronous CA for the density task. *Complex Systems*, 13(3):185–204, 2002.
17. D. J. Watts. *Small worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton NJ, 1999.
18. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

Extended Particle Swarm to Simulate Biology-Like Systems

Cecilia Di Chio

Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
cdichi@essex.ac.uk

Abstract. Is it possible to simulate socio-biological behaviours using particle swarm systems? And if so, what should it be the best approach to use? These are the questions which I would like to answer with my research. Particle swarm systems have been originally developed to model social behaviours. My research will therefore follow the initial socio-biological metaphor underlying particle systems. The idea is to use a genetic programming approach to automatically evolve the particle swarm equations to model animal social behaviours. This research is intended to be a first example of application of genetic programming and particle swarm to simulate animal behaviours.

1 Introduction

Particle Swarm Optimisation (PSO) is an optimisation algorithm for nonlinear functions which was discovered through the simulation of a simplified social (and cultural) model called the *particle swarm* [12].

Bearing in mind the original context in which particle swarm systems have been developed (i.e. modelling social behaviours), this research will follow the social and biological metaphor that constitutes the background for particle systems. It will therefore be focused on:

1. analysing existing versions of particle swarm systems from a socio-biological point of view,
2. studying possible integration with other methodologies in the context of the simulation of biological systems,
3. exploring new interactions among particles, among swarms, and among swarms and the environment.

The main goal of the research will be using genetic programming (GP) to automatically evolve the particle swarm equations to model simulations of animal social behaviours. I will start modelling some simple behaviours, and then progressively add complexity to them in order to match real data. This research is intended to be a first example of application of genetic programming and particle swarm algorithms in the simulation of animal behaviours. I hope, in the future, to be able to extend this approach to other socio-biological systems, to let genetic programming be a generic tool for the simulation of these systems.

In the following part of this section (sec. 1.1), I present a short survey of the most important PSO literature. In section 2 I describe the research questions I am trying

to answer (sec. 2.1), what I have done so far (sec. 2.2), and what I will do until the end of my study (sec. 2.3). Section 3 summarises some of the results I have previously presented. I present my achievements in section 4.

1.1 Literature review

General PSO papers Particle Swarm Optimisation (PSO) is an algorithm for finding optimal regions of complex search space through the interaction of individuals in a population of particles. The concept is first presented in 1995 by Kennedy et al. [10]. The authors introduce a new methodology for optimising continuous nonlinear functions based on the simulation of a simplified social model. The inspirational work for Kennedy and Eberhart is a 1987 paper by Reynolds [22] (which in turn takes inspiration from particle systems, first introduced in 1983 by Reeves [21]), in which the author describes a new approach to model the aggregate motion of flocks of birds (or schools of fish) in order to obtain a realistic computer animation. In 1998, Kennedy proposes a simplified version of the algorithm [11] to better understand the trajectories of the particles while they are looking for the solutions in the search space. In 1999, another paper to study trajectories of particles is presented by Ozcan et al. [17]. The first generalisation of the model is proposed in 1998 by Shi et al. [23] with the introduction of the *inertia weight*, a parameter used to control the impact of the previous history of velocities on the current velocity, balancing between wide-ranging and nearby exploration (i.e. *global* and *local search*). Another generalised model of PSO including methods to control the system's convergence tendencies is introduced by Clerc et al. in 2002 [4]). The authors introduce a new parameter to control the behaviour of the particles called *constriction coefficient*.

Comparisons and hybrids Few papers have been proposed in the literature with either comparison between PSO and other evolutionary computation paradigms or hybrid PSOs based on ideas taken from other evolutionary algorithms:

- A 1998 paper by Eberhart et al. [8], which presents a comparison between PSO and GA, highlighting the differences among the operators of each paradigms.
- A 1998 paper by Angeline [1], in which the author reviews the differences between PSO and evolutionary optimisation from a more philosophical point of view.
- A 2001 paper by Løvbjerg et al. [15], where the authors present two hybrid models of PSO based on ideas taken from genetics algorithms (GA). In particular, they combine the particle swarm update rules for velocity and position with the concepts of breeding and subpopulation.
- A 2005 paper by Holden et al. [9], in which is presented a solution to a classification problem by taking the best characteristic from Ant Colony Optimisation (ACO) (i.e its capacity to solve classification problems with categorical data), and from PSO (i.e. its ability to solve optimisation problems with continuous values).

PSO extensions Several variations and extensions of the PSO have been made over the years, with the aim of improving the performance of the algorithm:

- In 2002, Vesterstrøm et al. [25] present a version of the particle swarm algorithm extended with the concept of division of labour (DoL) typical of biological systems (e.g. social animals).

- In 2002, Brits et al. [3] introduce the concept of niche in the PSO framework to locate multiple optimal solutions in multimodal problems in parallel.
- In 2002, Silva et al. [24] take inspiration from the predator-prey model of biological interaction.
- In 2002, Blackwell [2] introduces the idea of charged PSO, which are well suited to solve dynamical search problems as inter-particle repulsion maintains population diversity and good tracking can be achieved.
- In 2005, Poli et al. ([19]) present a new way to automatically generate the equations of the forces driving the particles in the swarm. Variations of this idea have been implemented, and will be discussed later in this document (see § 3).

2 Research and study

2.1 Goals

The particle swarm paradigm has been "invented" to model social behaviours. In my research, I would like to focus on this socio-biological background, exploring it and extending the definition of the particle swarm model in order to simulate biological systems.

Is it possible to simulate biological systems (and model social behaviours, individuals interactions, animal social organisation) using particle swarms?

One of the aspects of particle swarms I have been very interested about since I started to work with them is the interaction among particles. I believe that, to simulate animal behaviours, it is necessary to differentiate both the particles and the interactions that occur among them. By differentiation, I mean introducing in the particles some kind of diversity in order to be able to distinguish a particle with certain specific characteristics and another particle with different characteristics. For instance, among social animals, roles and specialisation exist, and in some cases the social differences become so extreme (*eusociality*) that a reproductive specialisation happens, with some individuals devoted to reproduction and others sterile devoted to work and defence of the nest. Social differences among individuals should be reflected in differences in interactions, i.e. interactions among individuals in the same social class are different from interactions among individuals belonging to different social groups (animal social organisation). Anyway, individuals in different social groups are still part of the same swarm, i.e. a swarm can be made up of one or more social groups of individuals (in this context, a swarm can be seen as an animal species).

How many different types of interactions among individuals, social groups and swarms can emerge?

A very preliminary (and possibly subject to variations) classification of the possible typologies of interactions that can emerge is the following:

- among particles in the same social group
- among particles in different social groups within the same swarm
- among social classes
- among swarms
- between a social group and the environment

- between a swarm and the environment

The next steps would be first studying each different type (*level*) of interaction independently, and later gather them for an overall analysis of their effects.

What kind of interactions can occur?

Several types of interactions among animals within the same species (e.g.mating, feeding) and among species (e.g. predation, mutualism) exist. Interaction between a species and the environment in which the individuals live can be seen as a special instance of the interaction among species, considering the environment as a species itself (e.g. herbivore eating plants is a kind of predation, bees pollinating plants is a form of mutualism). Different types of interactions will introduce some sort of cooperation among individuals and co-evolution of species, and will also affect the population size of each swarm. The first two aspects have already been considered in the PSO theory, but again only from the point of view of optimisation. They will now be studied from the point of view of social behaviours. The latter (changes in swarm population size) has not been inspected too much even in the optimisation context. Anyway, in my opinion, this is a crucial aspect in biological systems (birth is also another particular type of interaction) that cannot be ignored.

What kind of approach should be used to model the simulation of socio-biological systems?

The idea is to use a genetic programming approach as a tool to automatise the process of simulating social behaviours in animals.

2.2 Current status

My research is part of the eXtended Particle Swarm (XPS) project[1]. The aim of this multidisciplinary research project is to systematically explore the extension of particle swarms by including strategies from a wide range of collective behaviours in biology, by extending the physics of the particles, by generating an extensive set of engineering problems and a flexible simulation engine, and by providing a solid theoretical and mathematical basis for the understanding and problem-specific design of new particle swarm algorithms. My research will sit between the biological and the theoretical streams, that are just two of the many disciplinary streams in which the project is partitioned.

During the first six months of my research, my main activity has been reviewing PSO literature in order to get a suitable background. After few weeks, I have also started to develop my programming skill in the context of the PSO working on a paper by Poli et al. ([19]), which represents an existing research topic in the XPS project. First, I have extended the idea of using genetic programming to evolve particle swarm forces by adding ingredients to the equations. I have then tested the performance of the new model with different neighbourhood topologies and on pool of testing functions (I will explain these works in detail in § 3). Thanks to this initial investigation, I have been able to publish two papers. The first [20] was presented at the 2005 "Genetic and Evolutionary Computation Conference (GECCO)", whilst the second [7] was presented at the 2005 "Workshop on Evolutionary Computation (GSICE)", where it won an award as Best Paper.

[1] http://xps-swarm.essex.ac.uk

In the following six months, I have improved my biological background with a literature review on the subject ([14, 18, 16, 13]) and keeping in touch with members of the biology stream of the XPS project. I have also worked on the initial design of the model, and I am presently implementing the standard version of the particle swarm algorithm in Java with the Mason simulation toolkit[2].

2.3 Future planning

For the two remaining years, my research will be divided into two phases, during which I will approach the same problem of simulating (social) animal behaviours (evolving group behaviours in animals) following two paths, that can be considered one the "natural" extension of the other.

Phase 1 In this first phase, I will use the GP has an automatic tool to evolve the equations of a model of animal behaviour. These models' performances will be compared to those of the particle swarm and a model of animal aggregation proposed by Couzin et al. ([5, 6]). In this phase, the key point will be the choice of the problem (and the respective fitness function). I will let the GP find the model for three different animal grouping scenarios, namely feeding, hunting and escaping predators. Each of these three problems has a different difficulty to be analysed, in particular:

feeding - homogeneous swarm looking for multiple static resources
hunting - heterogeneous swarm (bold/shy) looking for (single) dynamic resources
escaping (predators) - multiple (prey/predator) homogeneous swarms

I will then compare the results obtained by the GP evolved models on those scenarios with the results of the particle swarm algorithm and the biological inspired model on the same problems. The fitness for the three scenarios can be evaluated on various factors, e.g. amount of food eaten (feeding), catching the prey (hunting), staying alive (escaping).
Outline:

1. Implement the particle swarm algorithm, the biologically inspired model and a simple version of genetic programming in Java (with the Mason simulation toolkit).
2. Evolve simple animals behaviours (on each one of the three previously presented scenarios), modifying the equations of the models with a genetic programming approach.
3. Test the obtained extended models over real data.

Phase 2 On the second phase, I will again use a GP approach to evolve the strategies that characterise animal grouping behaviours. In this phase of evolving the strategies, it will be interesting to see if the GP is able to evolve a "social" behaviour without having any "handcrafted information" about the fact the agents have to stay close together (one of the three rules that describe a flocking behaviour). This way, a social behaviour should emerge only thanks to certain conditions in the environment (i.e. if the environment is such that is more advantageous to stay in group, then sociality will emerge, otherwise the animals will stay isolated). The problems (scenarios) will be

[2] http://cs.gmu.edu/~eclab/projects/mason

the same as before. Initially, the GP will evolve a strategy for each one of those, but they will eventually be combined to create an unique multi-objectives scenario (this will allow me to make experiments with multiple and heterogeneous swarms). The fitness will be measured either as the number of time-steps needed to obtain the goal (feed, mate, hunt, or survive) or as the survival of the population (in terms of number of individuals surviving after having accomplished a task or after a fixed amount of time).

Outline:

1. Initialise the population of agents with random strategies (i.e. different strategies for different agents) and position them at random in the landscape (i.e. different position in the space).
2. Let the agents (i) go for n time-steps and/or (ii) reach the target.
3. Evaluate the fitness as described in (i) **Phase 1** and/or (ii) **Phase 2**.
4. Select the "best" agents (i.e. the agents with the fittest behaviour) and crossover them to obtain the new population.

3 Results

3.1 Methodology

In this section, I describe the results obtained in previous works [20, 7]. These have already been presented in other conferences and are based on [19]. In that paper, the authors started exploring the possibility of evolving, through the use of GP, the force-generating equations which control the particles:

$$a_i = F(x_i(t-1), x_{s_i}, x_{p_i}, v_i(t-1)) \tag{1}$$

where x_i is the particle's current location, x_{s_i} is the best point visited by the swarm, x_{p_i} is the particle's personal best, and v_i is the particle's velocity. In [19], the GP evolved PSOs with good performances, in particular:

PSOG1 - $F = (x_{s_i} - x_i) - (v_i R)$
PSOG3 - $F = R_1(x_{s_i} - x_i) - 0.75R_2R_1x_ix_{s_i}^2 - 0.25R_3R_2R_1x_ix_{s_i}$

However, the exploration was limited by the use of the same ingredients present in the original PSO model. In [20], we have independently verified the previous findings and then extended the search by considering additional information about the global state of the swarm:

$$a_i = F(x_i(t-1), x_{s_i}, x_{p_i}, v_i(t-1), x_{c_i}, d) \tag{2}$$

where x_{c_i} is the centre of mass of the swarm, and d is the dispersion of the swarm around its centre of mass. These were believed to be very important so as to provide a way of adapting the nature of the forces used depending on the current situation of the swarm as a whole. We extended our work further in [7]. Here we considered other new ingredients for the control law of the forces acting on the particles:

$$a_i = F(x_i(t-1), x_{s_i}, x_{p_i}, v_i(t-1), v_{s_i}, x_{c_i}, d, t) \tag{3}$$

where v_{s_i} is the velocity for the best point visited by the members of the swarm, and t is a time factor. These were believed to be important to allow the adaptation of these forces to the current situation of the swarm: providing the particles with more information about the overall state of the swarm, should allow more sophisticated search behaviour.

3.2 Results

In [20], experiments have been done with the settings summarised in the table 1. In [7], the experiment settings changed according to the parameters in the equation of the force governing the PSO. The new objective and terminal set are summarised in table 2. All the other settings remained the same as before.

Table 1. GP's parameter settings in [20]

Objective	Evolve equation (2) over 10 random problems from the City-block Sphere or Rastrigin function classes with dimensions $N = 2$ and $N = 10$, and values for the global optima $G = 1.0$ and $G = 2.0$
Terminal set	x_i, x_{s_i}, x_{p_i}, v_i, x_{c_i}, d, -1.0, -0.5, 0.5, 1.0 and a random number generator R which returns numbers uniformly distributed within $[-1, 1]$
Function set	$+$, $-$, $\times$ and the protected division DIV
Fitness cases	10 particles with random initial position (in the interval $[-5.0, 5.0]$) and initial velocity null; 30 iterations on each problem; for each problem, 5 runs with different initial random positions for the 10 particles; velocity of particles updated using Clerc's update rule [4] with $\kappa = 0.7$ and the components of the velocity vector constrained within $[-2.0, 2.0]$
Fitness	$\sum_x \sum_i \lvert x_i - g_i \rvert$, i.e. the sum of the distances between each particle and the global optimum
Population size	1000 forces
Initialisation method	Random
Simulation time	100 generations
Crossover probability	90% standard sub-tree crossover (with uniform random selection of crossover points)
Mutation probability	10% point mutation with a 2% chance of mutation per node
Initial program length	6 levels, the root being at level 0
Selection scheme	Steady state binary tournaments for parent selection and binary negative tournaments to decide who to remove from the population
Termination criteria	Automatically at generation 100

In order to evaluate the PSOs produced by GP, we compared them with a number of both human-designed (e.g. standard **PSO** and random **PSOR1**) and previously evolved update rules (e.g. **PSOG1** and **PSOG3**; for a comprehensive set of results, and the details of the testing settings, please refer to [19]).

For [20], we selected the set of testing problems from the City-block Sphere and Rastrigin benchmark functions with dimension $N = 2$ and $N = 10$, and $G = 1.0$ and $G = 2.0$:

Table 2. GP's parameter settings in [7]

Objective	Evolve equation (3) over 10 random problems from the Sphere or Rastrigin function classes with dimensions $N = 2$, and values for the global optima $G = 1.0$
Terminal set	x_i, x_{s_i}, x_{p_i}, v_i, v_{s_i}, x_{c_i}, d, t, -1.0, -0.5, 0.5, 1.0 and a random number generator R which returns numbers uniformly distributed within $[-1, 1]$

City-block sphere - unimodal with global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$

$$f(x) = \sum_{i=1}^{N} |x_i - g_i|$$

Rastrigin's - multimodal with global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many local optima

$$f(x) = 10N + \sum_{i=1}^{N} \left((x_i - g_i)^2 - 10\cos(2\pi(x_i - g_i)) \right)$$

Among the many new PSOs automatically created by GP, we present here only the two best performing ones:

PSODIS2 - evolved on the two dimensional City-block Sphere functions with $G = 2.0$, with only the dispersion as new added characteristic in the terminal set.

$$F = (x_{s_i} - x_i) + (x_{p_i} - x_i) - dx_i$$

The evolved function is completely deterministic. Should the swarm collapse (in which case the dispersion term d would be zero) then this rule would become the standard (deterministic) PSO. Normally d is non-zero, so the third component (dx_i) tends to push the swarm towards the origin.

PSOCD1 - evolved on the two dimensional City-block Sphere functions with $G = 1.0$, with both centre of mass of the swarm and dispersion in the terminal set.

$$F = (x_{s_i} - x_i) - \frac{R^2}{d} v_i$$

The first term is the 100% social term, which tends to move the swarm closer to its best point. The random friction term ($\frac{R^2}{d} v_i$) is inversely proportional to swarm's dispersion d. So when the particles are close to each other the friction is higher than it is when the swarm is more spread. This could mean that at the beginning the search is more rapid while, when the particles get closer their movement becomes slower.

Table 3 shows the mean over the 30 runs and the standard deviation of the normalised distance between the best location found by each PSO and the global optimum.

For [7], we tested the evolved PSOs on a larger set of benchmark functions:

Table 3. Normalised mean (and standard deviation) of the distance between the best location found by each PSO and global optima of the City-block Sphere and Rastrigin functions in [20]. Best results in bold

	City-block Sphere				**Rastrigin**			
	$N = 2$		$N = 10$		$N = 2$		$N = 10$	
	$G = 1.0$	$G = 2.0$	$G = 1.0$	$G = 2.0$	$G = 1.0$	$G = 2.0$	$G = 1.0$	$G = 2.0$
PSO	0.184	0.22	0.826	0.946	0.726	0.859	1.336	1.471
	(0.248)	(0.273)	(0.579)	(0.548)	(0.322)	(0.31)	(0.47)	(0.382)
PSOR1	0.003	0.003	0.279	**0.315**	0.638	0.687	1.332	1.402
	(0.0005)	(0.0005)	(0.018)	(0.029)	(0.076)	(0.09)	(0.065)	(0.06)
PSOG1	**0.0005**	0.002	0.455	0.554	0.89	1.058	1.293	1.389
	(0.0005)	(0.002)	(0.02)	(0.047)	(0.101)	(0.1)	(0.063)	(0.072)
PSOG3	0.009	0.043	**0.183**	0.456	**0.307**	**0.556**	0.578	**0.943**
	(0.004)	(0.025)	(0.022)	(0.101)	(0.078)	(0.152)	(0.062)	(0.119)
PSODIS2	0.003	0.009	0.262	0.545	0.326	0.642	**0.562**	**0.943**
	(0.002)	(0.004)	(0.029)	(0.094)	(0.067)	(0.144)	(0.077)	(0.121)
PSOCD1	**0.0005**	**0.0005**	0.283	0.353	0.717	0.745	1.326	1.405
	(0.00)	(0.00)	(0.019)	(0.036)	(0.089)	(0.108)	(0.067)	(0.082)

Generalised Ackley - multimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many local optima.

$$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - g_i)^2}} - e^{\frac{1}{N}\sum_{i=1}^{N} cos(2\pi(x_i - g_i))}$$

Griewangk - multimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$, and many regularly distributed local optima.

$$f(x) = 1 + \sum_{i=1}^{N} \frac{(x_i - g_i)^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i - g_i}{\sqrt{i}}\right)$$

Generalised Rastrigin - see above

Rosenbrock - unimodal with one global optimum at $x = (g_1 + 1, \cdots, g_N + 1)$, with $f(x) = 0$.

$$f(x) = \sum_{i=1}^{N} (100((x_i - g_i) - (x_{i-1} - g_{i-1})^2)^2 + (1 - (x_{i-1} - g_{i-1}))^2)$$

Sphere - unimodal with one global optimum at $x = (g_1, \cdots, g_N)$, with $f(x) = 0$.

$$f(x) = \sum_{i=1}^{N} (x_i - g_i)^2$$

We used problems of dimension $N = 2$, $N = 10$ and $N = 30$, and $G = 1.0$ and $G = 2.0$. The best performing PSO was:

PSOTIME - evolved on the Rastrigin function, with only time as new added characteristic in the terminal set.

$$F = (t + 1)(x_{s_i} t - x_i)$$

This one as well is completely deterministic and 100% social. A qualitative explanation of the behaviour of this PSO can be given by analysing how the force changes as the time increases (i.e. the search progresses):

- at the beginning of the run ($t = 0$), all particles are attracted towards the area where the optima typically are (around the origin);
- as the run progresses, the swarm best becomes progressively the attractor for the particles. Eventually ($t \simeq 1$) and the bias towards the origin disappears.

Why has the GP evolved this particular force? Recalling the update rule modified with the constriction factor [4], we can rewrite the equation as

$$\begin{aligned} v_i(t) &= \kappa v_i(t-1) + \kappa F \\ &= v_i(t-1) - (1-\kappa) v_i(t-1) + \kappa F \end{aligned}$$

We can consider the second term $-(1-\kappa)v_i(t-1)$ as friction, while the coefficient multiplying F (the constriction factor, κ) can be interpreted as the *inverse* of the mass of the particles. In all our experiments we compose our force generating equations with this update rule (with $\kappa = 0.7$). So, the force is first scaled by κ and then added to the friction component $-(1-\kappa)v_i(t-1)$. Therefore, a time varying scaling factor $(1+t)$ has been evolved. This effectively means that at different times the particles have different masses. At time $t = 0$, the mass is $\kappa^{-1} \approx 1.4$. However, as time progresses, first the effects of the constriction coefficient are completely removed (at $t = 0.5$ the mass is $\frac{1}{\kappa(1+t)} \approx 1$) and then the mass is eventually reduced to approximately 0.7 at $t = 1$.

For each of the five benchmark functions, table 4 summarises the best results obtained in testing the newly evolved PSOs against the existing ones for dimensions $N = 2$, $N = 10$ and $N = 30$. Again the table shows the normalised mean and standard deviation over 30 runs of the distance between the swarm best and the global optimum.

4 Achievements and conclusions

The results obtained in the work I have done so far show that GP is able to evolve a variety of PSOs that work as well as, or considerably better than, standard human-designed ones. These work represent an important step within a new research trend: using search algorithms to discover new search algorithms. Analysis of the evolved programs has led to new insights in the design of PSOs tailored for specific classes of landscapes. Our main contribution is to show that genetic programming can automatically evolve better than human-designed PSOs in a few hours on a standard PC. Adding extra information has confirmed our hypothesis that, if a particle has a greater knowledge of the other individuals in the swarm, the whole swarm may perform better in finding the optimum.

The good results obtained in my previous work and the robustness of the method used, together with a personal interest in the socio-biological aspect of particle swarm systems, inspired the next step of my research: apply the genetic programming approach to simulate social behaviours in animals. This research is intended to be a first example

Table 4. Normalised mean (and standard deviation) of the distance between the best location found by each PSO and global optima for each test function in [7]. Best results in bold

		$N = 2$		$N = 10$		$N = 30$	
		$G = 1.0$	$G = 2.0$	$G = 1.0$	$G = 2.0$	$G = 1.0$	$G = 2.0$
Ackley	**PSOTIME**	**0.2420**	**0.3899**	**0.3843**	**0.4901**	0.4755	**0.5522**
		(0.0629)	(0.1017)	(0.0347)	(0.0545)	(0.0510)	(0.0459)
Griewangk	**PSOG3**	**0.0170**	0.2605	**0.0685**	**0.1915**	**0.1299**	**0.2709**
		(0.0202)	(0.2445)	(0.0184)	(0.0512)	(0.0115)	(0.0405)
	PSOTIME	0.0598	**0.2210**	0.1212	0.2583	0.1543	0.3089
		(0.0343)	(0.1687)	(0.0283)	(0.0514)	(0.0153)	(0.0420)
Rastrigin	**PSOG3**	**0.0997**	**0.1536**	0.1851	**0.3056**	0.1968	**0.3357**
		(0.0235)	(0.0488)	(0.0179)	(0.0414)	(0.0139)	(0.0235)
	PSOTIME	0.1130	0.2145	**0.1777**	0.3171	**0.1856**	0.3370
		(0.0512)	(0.0947)	(0.0291)	(0.0428)	(0.0146)	(0.0276)
Rosenbrock	**PSOG3**	0.2179	0.2233	**0.0797**	**0.1896**	**0.1320**	**0.2806**
		(0.0430)	(0.0856)	(0.0200)	(0.0453)	(0.0122)	(0.0269)
	PSOTIME	**0.1387**	**0.1931**	0.1236	0.2599	0.1467	0.3172
		(0.0582)	(0.0992)	(0.0288)	(0.0535)	(0.0158)	(0.0330)
Sphere	**PSOG3**	0.0036	0.0136	**0.0615**	**0.1451**	**0.1191**	**0.2508**
		(0.0023)	(0.0099)	(0.0102)	(0.0269)	(0.0099)	(0.0269)

of application of genetic programming and particle swarm algorithms in the simulation of animal behaviours. I hope, in the future, to be able to extend this approach to other socio-biological systems, to let genetic programming be a generic tool for the simulation of these systems.

References

1. P.J. Angeline, *Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences*, Evolutionary Programming Conference, 1998.
2. T.M. Blackwell and P.J. Bentley, *Dynamic Search with Charged Swarms*, Genetic and Evolutionary Computation Conference, 2002.
3. R. Brits, A.P. Engelbrecht and F. van den Bergh, *A Niching Particle Swarm Optimizer*, Asia-Pacific Conference on Simulated Evolution and Learning, 2002.
4. M. Clerc and J. Kennedy, *The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space*, IEEE Transaction on Evolutionary Computation, 2002.
5. I.D. Couzin, J. Krause, R. James, G.D. Ruxton and N.R. Franks, *Collective memory and spatial sorting in animal groups*, Journal of Theoretical Biology, 2002.
6. I.D. Couzin, J. Krause, N.R. Franks and S.A. Levin, *Effective leadership and decision-making in animal groups on the move*, Nature, 2005.
7. C. Di Chio, R. Poli and W.B. Langdon *Evolution of Force-Generating Equations for PSO using GP*, Workshop on Evolutionary Computation, 2005.
8. R.C. Eberhart and Y. Shi, *Comparison between Genetic Algorithms and Particle Swarm Optimization*, Evolutionary Programming Conference, 1998.
9. N. Holden and A.A. Freitas, *A Hybrid Particle Swarm/Ant Colony Algorithm for the Classification of Hierarchical Biological Data*, Swarm Intelligence Symposium, 2005.
10. J. Kennedy and R.C. Eberhart, *Particle Swarm Optimization*, International Conference on Neural Networks, 1995.
11. J. Kennedy, *The Behavior of Particles.*, Evolutionary Programming Conference, 1998.
12. J. Kennedy and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.
13. , J. Krause and G.D. Ruxton, *Living in Groups*, Oxford University Press, 2002.
14. , J.R. Krebs and N.B. Davies, *An Introduction to Behavioural Ecology*, Blackwell Scientific Publications, 1982.
15. M. Løvbjerg, T.K. Rasmussen and T. Krink, *Hybrid Particle Swarm Optimiser with Breeding and Subpopulations*, Genetic and Evolutionary Computation Conference, 2001.
16. D. MacFarland, *Animal behaviour*, Longman, 1999.
17. E. Ozcan and C.K. Mohan, *Particle Swarm Optimization: Surfing the Waves*, Congress on Evolutionary Computation, 1999.
18. J.K Parrish and W.M. Hamner, *Animal Groups in Three Dimensions*, Cambridge University Press, 1997.
19. R. Poli, W.B. Langdon and O. Holland, *Extending Particle Swarm Optimisation via Genetic Programming*, European Conference on Genetic Programming, 2005.
20. R. Poli, C. Di Chio and W.B. Langdon, *Exploring Extended Particle Swarms: a Genetic Programming Approach*, Genetic and Evolutionary Computation Conference, 2005.

21. W.T. Reeves, *Particle Systems - a Technique for modelling a Class of Fuzzy Objects*, ACM Transaction on Graphics, 1983.
22. C.W. Reynolds, *Flock, Herds, and Schools: A Distributed Behavioral Model*, Computer Graphics, 1987.
23. Y.Shi and R.C. Eberhart, *A Modified Particle Swarm Optimizer*, International Conference on Evolutionary Computation, 1998.
24. A.Silva, and A. Neves and Ernesto Costa, *Chasing the Swarm: a Predator-Prey Approach to Function Optimisation*, Centre for Informatics and Systems of the University of Coimbra - Artificial Intelligence Group, 2002.
25. J.S. Vesterstrøm, J. Riget and T. Krink, *Division of Labor in Particle Swarm Optimisation*, Congress on Evolutionary Computation, 2002.

Appendix: Feedback

Of course, the first feedback I would like to have from the Doctoral Consortium is a general comment on the topic I have chosen. How feasible do you think it is? How relevant/interesting do you think is my idea for the scientific (EC, ALife, simulation) community? Do you think it will still be interesting in a few years?

Something that I think it would be really useful is knowing who are the other researchers working on the same (or related) topic, to become part of sub-community (i.e. specialised). In fact, I believe that (any form of) collaboration between people is the best way to understand a subject, having new ideas, and making great discoveries.

Last but not least, since English is not my first language, feedback on my writing style would be highly appreciated.

Geometric Unification of Evolutionary Algorithms

Alberto Moraglio

University of Essex, UK
amoragn@essex.ac.uk

Abstract. Evolutionary algorithms are only superficially different and can be unified within an axiomatic geometric framework by abstraction of the solution representation. This framework describes the evolutionary search in a representation-independent way, purely in geometric terms, paving the road to a general theory of evolutionary algorithms. It also leads to a principled design methodology for the crossover operator for any solution representation.

1 Context of the research

Evolutionary Algorithms (EAs) are successful and widespread general problem solving methods that mimic in a simplified manner biological evolution. Whereas all EAs share the same basic algorithmic structure, they differ in the solution representation - the genotype - and in the search operators employed - mutation and crossover - that are representation-specific. Is this difference only superficial? Is there a deeper unity encompassing all mutation and crossover operators beyond the specific representation, hence all EAs? So far, no one has been able to attack this question successfully and has proposed a general mathematical framework that unifies search operators for all solution representations.

In the research community there is a strong feeling that the EC field needs unification and systematization in a rational framework to survive its own exceptional growth (De Jong [4]). Beside De Jong, there are important researchers who have been promoting EC unification: Radcliffe pioneered a unified theory of representations [11], although he never used the word "unification". Riccardo Poli unified the schema theorem for traditional genetic algorithms and genetic programming [3]. Chris Stephens suggests that all evolutionary algorithms can be unified using the language of dynamical systems and coarse graining [1]. Franz Rothlauf has initiated a theory of representations [12].

2 Research and study

2.1 Research questions and goals

My research questions are:

1. *Possibility: Is the unification of evolutionary algorithms within a general mathematical framework possible?*
2. *Utility: What are the advantages and consequences of the unification?*

The significance of the unification is not obvious a priori and resides in the important consequences and insights brought about by seeing evolutionary algorithms from a new and more general viewpoint. For this reason the focus of my research is both on possibility and utility of unification.

My goals are: developing a *formal framework* for the unification, show that this framework helps to *rethink* various familiar aspects of evolutionary algorithms simplifying and clarifying their roles, show that the *unification is possible* and many preexisting representations and operators fit the framework, show that the formal framework can be used to do *crossover principled design* for any representation, show that the framework forms a solid basis for a *representation-independent theory* that applies to all evolutionary algorithms.

2.2 Current status: overview of the achievements

In this section, a short overview of the research achievements is reported. A sample of the results that will appear in the thesis is reported in section 4. Most of this research has been already published [5] [6] [8] [7] [9] or is about to be published.

1. *Possibility of unification*: mathematical unification is possible. Developed an axiomatic geometric framework for unification. The definition of search operators is axiomatic and intentionally does not involve the notion of representation: unification by abstraction of the representation[1].
2. *Clarification and simplification*: the change in perspective coming with unification completely reverses the orthodoxy [2] clarifying and simplifying many fundamental aspects of evolutionary algorithms. Clarified the consequences of this new perspective on fundamental notions: common search space for mutation and crossover, problem-independent and representation-independent formal evolutionary algorithm, simple fitness landscape of crossover, geometric format of problem knowledge, role of the EA designer, duality of neighborhood search and representation-based evolutionary search.
3. *Interesting scope of unification*: the significance of unification lies on how many interesting cases it encompasses. Shown that many interesting pre-existing operators for the most-used representations fit the requirements of the unification.
4. *Crossover principled design*: by reversing the abstraction and applying the abstract definition of crossover to a distance firmly rooted in a specific representation one obtains a formal recipe to build new crossovers for any representation. When the distance chosen as basis for the new crossover is meaningful in terms of the problem addressed, the new operator is likely to perform well. This way of doing crossover principled design is the representation dual of the neighborhood search meta-heuristic path-relinking that suggests picking new solutions on a path (not necessarily shortest) in the search space connecting parent solutions. Unlike path-relinking that neglects altogether the underlying solution representation and does not show how to actually generate offspring (that is left as "implementation details"), geometric crossover tells exactly how to manipulate the syntax of the solution representation to build offspring solutions. Various examples of crossover design are given with good experimental results.

[1] This does not mean I regard solution representation as an "implementation detail" and unimportant. This means that the relationship between *representation and search operators*, which is ultimately what is relevant to the search, can be expressed in a geometric language that is representation-independent.

5. *Depth of the abstraction and general theory*: the significance of an abstraction lies on the kind of results that can be inferred using only the axioms the abstraction is based upon; if only trivialities encompassing all evolutionary algorithms can be inferred, the abstraction is not significant. The geometric abstraction is indeed significant. (i) A fundamental result is that all evolutionary algorithms with any solution representation and with search operators that fit the geometric framework do the same search, convex search. This is a simple, deep and important result arising from the geometric interpretation of mutation and crossover only. (ii) Convexity plays a major role in the way evolutionary algorithms search the space: the evolutionary search can be naturally recast from the point of view of the underlying metric convexity associated with the search space. This allows showing the correspondence of metric convex sets with inheritable genetic traits, and generalize in a representation-independent way the notion of schema. Then, doing coarse-graining of the equation of the evolutionary dynamics over the convex sets one obtains a representation-independent schema theorem. The importance of this result relies on the fact that it reconciles two fundamental notions that until now were separated: the notion of inheritance and the notion of fitness landscape. Specifying the representation-independent definition of inheritable trait (schema) for a distance rooted on a specific solution representation, one can reveal the syntactic appearance of schemata for any representation. When applied to DNA strands with edit distance, this can have important application in genetics to discover new genes. (iii) Knowing how all evolutionary algorithms search the space is preliminary to understand under what general condition on the fitness landscape they perform well. Fitness distribution, correlation of the fitness landscape and performance are studied together to show why positive correlation in the landscape makes geometric crossover and geometric mutation perform better than random search.

2.3 Future planning

From February to June 2006 focus on: writing-up the thesis. I have most of the material in the form of draft/submitted/published papers. Background activities: writing a journal paper, writing grant proposal and maybe submitting a paper to PPSN.

Future after PhD: ideally, I would like to stay in academia, becoming a lecturer and continuing this research.

2.4 Study

I am finishing my third year of PhD study. In theory, one could finish within 3 years; in practice, it is more common to do 3 years of research and then doing the writing-up and submitting the thesis the 4th year, the so-called completion year. Every 6 months there is a board meeting in which a panel of academics evaluate the progress of PhD students from the previous board meeting using a system with milestones and give suggestions/feedback on the research and research plan.

3 Methodology and thesis outline

3.1 Methodology: focus on unification

Unification is a delicate matter: besides proving that unification is possible in principle, one has to explain the consequences of the new angle on a network of related concepts. Plus, one has to go wide and show that many interesting cases are actually encompassed, but one also needs to go deep to show that the unification does capture some fundamental aspects common to all evolutionary algorithms and not only trivialities can be inferred for all evolutionary algorithms. Moreover one needs also to show that unification is not only theory for its own sake but that has practical advantages too. This all needed to be done within a time-window of a PhD study. To different extent I think I managed to address all the previous points. Naturally, since a complete and thorough unification is a monumental task, I focussed on particular topics that I felt had the priority and showed that a particular important territory is encompassed by the unification. By no means, I have exploited systematically each topic covered as much as it would have deserved. Indeed, each topic would have taken a PhD to be exhaustively addressed alone. Since the focus is the unification, for each topic I have shown instead how the geometric framework reveals its connection with all the others and this in turn had shed light on the specific topic itself.

3.2 Thesis outline

Title: *Geometric unification of evolutionary algorithms*
Thesis: Evolutionary algorithms are only superficially different and can be unified within an axiomatic geometric framework by abstraction of the solution representation. This framework describes the evolutionary search in a representation-independent way, purely in geometric terms, paving the road to a general theory of evolutionary algorithms. It also leads to a principled design methodology for the crossover operator for any solution representation.

```
1 Introduction
2 Geometric framework
    2.1 Geometric preliminaries
    2.2 Search operators definition
    2.3 Change in perspective
            2.3.1 Formal evolutionary algorithm
            2.3.2 Geometric fitness landscape
            2.3.3 Problem knowledge
            2.3.4 Representation/neighbourhood duality
3 Representation unification
    3.1 Geometric unification by abstraction
    3.2 Binary and multary strings
    3.3 Real vectors
    3.4 Permutations: simple, circular and with repetitions
    3.5 Syntactic trees
    3.6 Biological sequences
    3.7 Structural representations
4 Crossover principled design
    4.1 Geometric design principles
```

4 Results

In this section I report a sample of the results that will appear in the thesis. Section 4.1 introduces the representation-independent definition of geometric crossover and describes how crossover connects with fitness landscape. The merit of the geometric framework here is that it simplifies enormously this connection: unlike the established paradigm [2] the geometric interpretation of crossover in the landscape is simple and intuitive. Section 4.2 shows that a number of recombination operators for 5 important representations are actually specific instances of geometric crossover. Section 4.3 gives an example of crossover principled design for the TSP problem. Experimental results (not reported) show that this crossover performs extremely well. Finally, section 4.4 gives an example of a very general representation-independent theoretical result that holds for all evolutionary algorithms using instances of geometric crossover.

4.1 Geometric framework

Geometric preliminaries The terms *distance* and *metric* denote any real valued function that conforms to the axioms of identity, symmetry and triangular inequality. A simple connected graph is naturally associated to a metric space via its *path metric*: the distance between two nodes in the graph is the length of a shortest path between the nodes. Similarly, an edge-weighted graph with strictly positive weights is naturally associated to a metric space via a *weighted path metric*.

In a metric space (S, d) a *closed ball* is the set of the form $B(x; r) = \{y \in S | d(x, y) \leq r\}$ where $x \in S$ and r is a positive real number called the radius of the ball. A *line segment* (or closed interval) is the set of the form $[x; y] = \{z \in S | d(x, z) + d(z, y) = d(x, y)\}$ where $x, y \in S$ are called extremes of the segment. Metric ball and metric segment generalize the familiar notions of ball and segment in the Euclidean space to any metric space through distance redefinition. These generalized objects look quite different under different metrics. Notice that a metric segment does not coincide to a shortest path connecting its extremes (*geodesic*) as in an Euclidean space. In general, there may be more than one geodesic connecting two extremes; the metric segment is the union of all geodesics.

We assign a structure to the solution set by endowing it with a notion of distance d. $M = (S, d)$ is therefore a solution *space* and $L = (M, g)$ is the corresponding fitness landscape (where g is the fitness function). Notice that d is arbitrary and need not have any particular connection or affinity with the search problem at hand.

Geometric crossover definition The following definitions are *representation-independent* therefore crossover is well-defined for any representation. It is only *function of the metric d* associated with the search space being based on the notion of metric segment.

Definition 1. *(Image set) The* image set $Im[OP]$ *of a genetic operator OP is the set of all possible offspring produced by OP with non-zero probability.*

Definition 2. *(Geometric crossover) A binary operator is a geometric crossover under the metric d if all offspring are in the segment between its parents.*

Definition 3. *(Uniform geometric crossover) Uniform geometric crossover UX is a geometric crossover where all z laying between parents x and y have the same probability of being the offspring:*

$$f_{UX}(z|x, y) = \frac{\delta(z \in [x; y])}{|[x; y]|}$$
$$Im[UX(x, y)] = \{z \in S | f_{UX}(z|x, y) > 0\} = [x; y].$$

A number of general properties for geometric crossover and mutation have been derived.

Geometric crossover landscape Geometric operators are defined as functions of the distance associated to the search space. However, the search space does not come with the problem itself. The problem consists only of a fitness function to optimize, that defines what a solution is and how to evaluate it, but it does not give any structure on the solution set. The act of putting a structure over the solution set is part of the search algorithm design and it is a designer's choice. A fitness landscape is the fitness function plus a structure over the solution space. So, for each problem, there is one fitness function but as many fitness landscapes as the number of possible different structures over the solution set. In principle, the designer could choose the structure to assign to the solution set completely independently from the problem at hand. However, because the search operators are defined over such a structure, doing so would make them decoupled from the problem at hand, hence turning the search into something very close to random search.

In order to avoid this one can exploit problem knowledge in the search. This can be achieved by carefully designing the connectivity structure of the fitness landscape. For example, one can study the objective function of the problem and select a neighborhood structure that couples the distance between solutions and their fitness values. Once this is done problem knowledge can be exploited by search operators to perform better than random search, even if the search operators are problem-independent (as in the case of geometric crossover and mutation).

Under which conditions is a landscape well-searchable by geometric operators? As a rule of thumb, geometric mutation and geometric crossover work well on landscapes where the closer pairs of solutions, the more correlated their fitness values. Of course this is no surprise: the importance of landscape smoothness has been advocated in many different context and has been confirmed in uncountable empirical studies with many neighborhood search meta-heuristics [10].

4.2 Representation unification

We consider the application of geometric crossover to five important representations: real vectors, n-ary strings, permutations, variable-size sequences and syntactic trees. The aim here is to give a portfolio of concrete examples of application of the theoretical ideas outlined before. We will see how geometric crossover unifies pre-existing crossovers for different representations (real vectors, binary strings, permutations, syntactic trees), how it casts new interpretations of pre-existing crossovers (crossover for permutations are sorting algorithms), how it guides to the principled design and implementation of new crossovers for new representations (variable-size sequences), how it may connect artificial and biological evolution (variable-size sequences) and how it helps to understand what is the search space and distance associated to a pre-existing crossover operator (syntactic trees).

Real vectors Pre-existing crossover operators, both blending type crossovers and discrete type recombinations fit the definition of geometric crossover naturally (see fig. 1). The extended version of blending crossovers does not fit the definition of geometric crossover (for *any* distance).

Binary strings The Hamming scheme is the association scheme where the elements are vectors of length d over some alphabet of size q. The Hamming distance of two vectors is the number of coordinates where they differ. The Hamming graph H(d,q) is the graph that describes the distance-1 relation in the Hamming scheme. It is the direct product of d complete graphs of size q. For d = 2 one gets the qq grid, also known as the lattice graph of order q. The Hamming graphs H(d,2) are the familiar hypercubes associated with binary strings of size d. The search operators are implemented by syntactic manipulation of strings equivalent to the geometric transformations required by the search operators. A geodesic between two points is a shortest path between two points. A shortest path in the edit distance graph is a minimal sequence of edit moves that transforms the syntax of one parent to the syntax of the other. In the case of binary string the edit move is bit-flip and in the case of multary strings the edit move is a substitution. The uniform crossover for binary strings is equivalent to picking any minimal sequence of bit-flip (that changes at most once one bit) that transforms one parent string into the other and interrupt the transformation somewhere in the middle. The one-point crossover (selecting one crossover point and swapping strings tails) is equivalent to applying a macro edit-move equivalent to the application of a specific minimal sequence of edit moves connecting the two parent strings interrupted somewhere in the middle.

All traditional mask-based crossovers for binary and multary strings are geometric under Hamming distance. See also fig. 2.

Permutations Pre-existing operators for permutations are sorting algorithms in disguise because picking offspring on the shortest path based on edit distances for permutations translates into picking offspring on a minimal sorting trajectories between parent permutations (see fig. 3). For example, PMX is geometric under swap distance.

Syntactic trees The search space and distances associated with pre-existing genetic operators for syntactic trees are little understood. Here, differently form the previous

representations, we want to use the geometric definitions not to guide the design and implementation of new operators for syntactic trees but rather we want to find the distance, hence the search space, associated to pre-existing operators if any of such distance exists. There are various notions of distance defined for GP trees. Distances among GP trees are used to (1) maintain diversity in the population and (2) predict performance (fitness-distance correlation). If the distance employed does not match the operator used its use is meaningless. So far it has been difficult to show that a certain distance matches a certain operator. Here we propose a new distance, the structural hamming distance (SHD) (a variation of the well-known structural distance for trees), that perfectly matches with Poli's Homologous crossover. Fig.4 shows an example of how SHD and hyperschema connect.

Koza's crossover is not geometric: there is provably no distance for which it is a geometric crossover. Poli's homologous crossovers family (1-point crossover, uniform crossover, homologous crossover, point mutation) is geometric under SHD.

Variable-length sequences Let us consider a recombination for variable-length sequences that requires an inexact sequence alignment before applying the traditional crossover on the alignment.

```
Parent1=AGCACACA
Parent2=ACACACTA

best inexact alignment by dynamic programming:
AGCA|CAC-A
A-CA|CACTA

Child1=AGCACACTA
Child2=ACACACA
```

This crossover and its generalization can be proven to be geometric under Levinshtein edit distance (insertion, deletion, substitution).

Does it have biological significance? Edit distance is a very meaningful distance to compare DNA strands. The present model of crossover (based on perfect alignment) cannot explain molecular evolution. Molecular evolution is tried to be explained by mutation only or by unequal crossover (imperfect crossover). Two DNA strands before crossover align on their contents to minimize the free-energy admitting gaps, compressions and mismatch alignment (molecular annealing). Geometric crossover based on edit-distance could be a better model of biological crossover in that model more realistically the effect of molecular annealing and could explain molecular evolution.

4.3 Crossover principled design: TSP

We consider a solution as a tour of cities and, therefore, rather than being defined for permutations geometric crossover is defined over *circular permutations*.

In the case of circular permutations, the block-reversal move is the notion of edit distance that makes sense for TSP. In a single application to a tour, this does the minimal change to the adjacency relation among elements in the permutation. This move is the well-known 2-change move, and it is the basis for successful local search algorithms for TSP . Figure 5 shows the possible offspring (the segment) between two circular (parent) permutations under topological crossover.

Analogously to the linear case, the circular permutations in the segment under reversal distance are those laying in a minimal sorting trajectory from a parent circular permutation to the other. Sorting circular permutations by reversals is NP-hard. So, *the topological crossover under this notion of distance cannot be implemented efficiently.*

Sorting circular permutations by reversals is tightly connected with the problem of sorting linear permutations by reversals. So all the algorithms developed for the latter task can be used with minor modifications also for the former. Sorting linear permutations by reversals is NP-hard too. However a number of approximation algorithms (running in polynomial time) exist to solve this problem within a bounded error from the optimum. This allows implementing efficiently approximate crossovers whose image set is a super-set of that of the exact crossover. We have implemented this crossover and found that it outperforms edge recombination that is known to be very good for TSP.

4.4 Representation-independent theory: convex evolutionary search

Using the axioms of distance and the definition of geometric crossover we can prove a main result: an evolutionary algorithm using geometric crossover with any probability distribution, any kind of representation, any problem, any selection and replacement mechanism, does the same search: convex search. Proof based on abstract convexity (axiomatic geodesic convexity) and axiomatization of search process (abstract search process). See Fig. 6 for an example of convex search in the intuitive case of Euclidean space.

5 Achievements and future directions

Achievements: this work answers fully the research questions, now there is no doubt about possibility and utility of unification, and explores the main research directions the unification opens up discovering new territories that now are ready to be conquered. For its own nature, a unification project can be evaluated only at the end when the whole picture is laid down, since each block reinforces the significance of unification as a unity only when put aside all the others. My hope is that the overall picture that emerges looks like a harmonious balance between all aspects of the unification.

Future directions: there are three main roads to follow from here.

1. *The road to principled design*: the first one is to unify a design methodology merging path-relinking and geometric crossover design.
2. *The road to computational complexity*: the second road is developing a theory which aim is general computational complexity results for evolutionary algorithms.
3. *The road to biological evolution*: the third road is casting a computational perspective on biological evolution to understand why is able to do “intelligent design” so effectively and efficiently without intelligence.

6 Feedback

Unification is a delicate matter: to deliver a meaningful unification, on one hand, many different aspects needed to be explored in parallel without losing sight of the overall

objective. On the other hand, every topic needed to be exploited up to a point for which the consequences of unification on that particular direction were crystal clear. This all needed to be done within a time-window of PhD study. To different extent I think I managed to address all the important aspects of unification. However a question remains: is the overall emerging picture a harmonious balance between all aspects?

7 Acknowledgments

The author would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

1. A. Zamora C. R. Stephens. Ec theory: A unified viewpoint. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1394–1405, 2003.
2. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995.
3. W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, 2001.
4. A. Menon, editor. *Frontiers of Evolutionary Computation*. Springer, 2004.
5. A. Moraglio and R. Poli. Topological interpretation of crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1377–1388, 2004.
6. A. Moraglio and R. Poli. Geometric crossover for the permutation representation. *Technical Report CSM-429, University of Essex*, 2005.
7. A. Moraglio and R. Poli. Geometric landscape of homologous crossover for syntactic trees. In *Proceedings of CEC 2005*, pages 427–434, 2005.
8. A. Moraglio and R. Poli. Topological crossover for the permutation representation. In *GECCO 2005 Workshop on Theory of Representations*, 2005.
9. A. Moraglio, R. Poli, and R. Seehuus. Geometric crossover for biological sequences. In *Proceedings EuroGP (to appear)*, 2006.
10. P. M. Pardalos and M. G. C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, 2002.
11. N. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–205, 1991.
12. F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer, 2002.

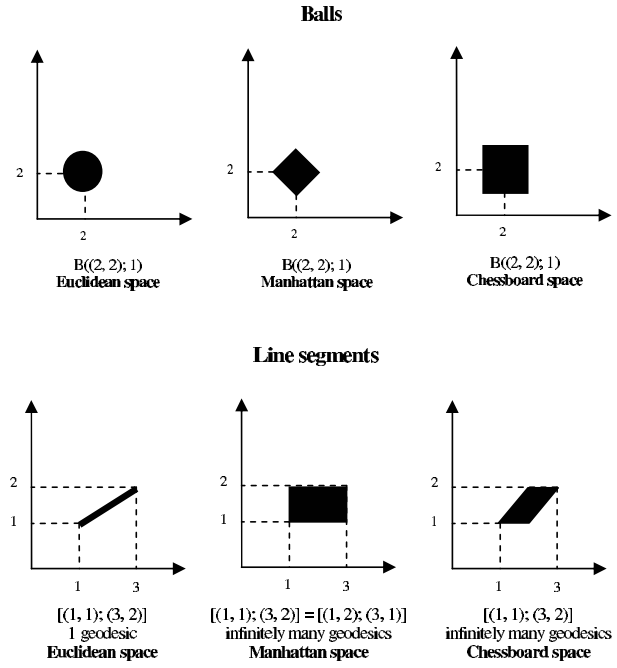


Fig. 1. Examples of balls and segments in Minkowsky spaces

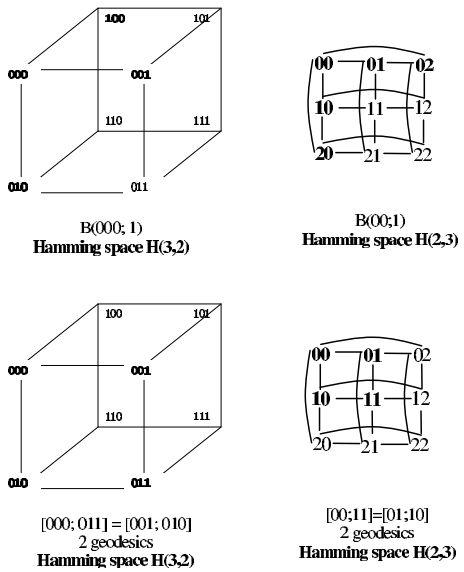


Fig. 2. Examples of balls and segments in Hamming spaces

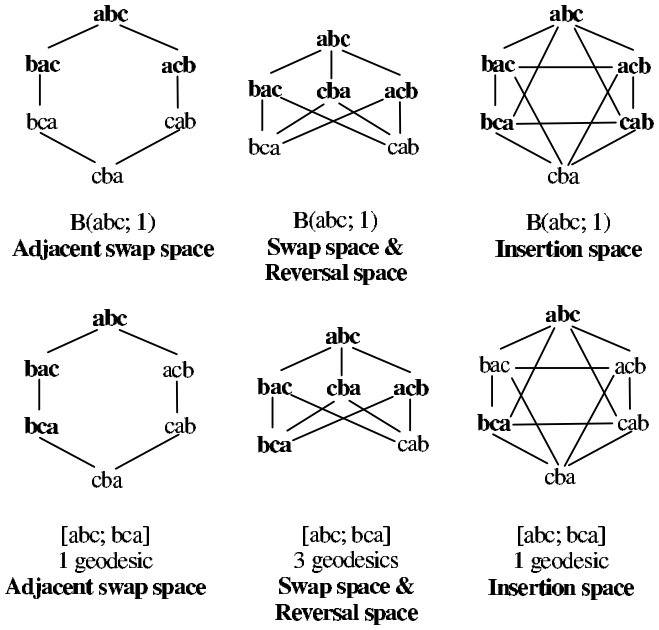


Fig. 3. Examples of balls and segments in Cayley spaces

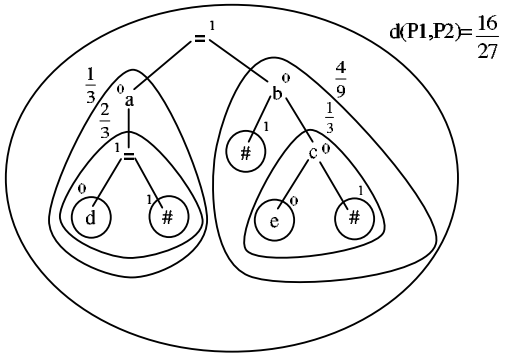
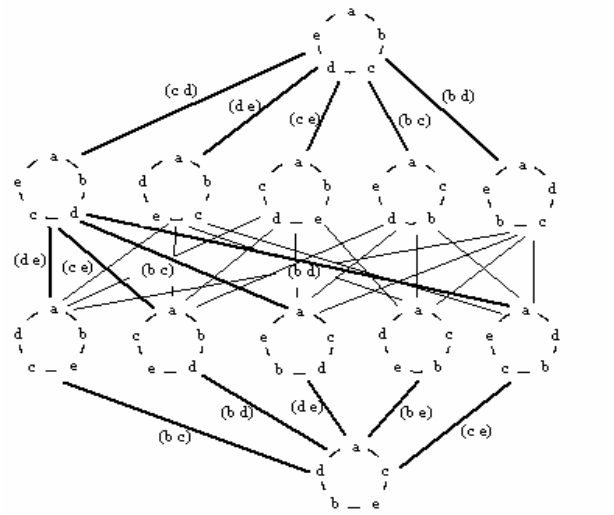


Fig. 4. Structural Hamming distance and hyper-schema


(c d)
(d e)
(c e)
(b c)
(b d)
(d e)
(c e)
(b c)
(b d)
(b d)
(d e)
(b e)
(c e)
(b c)

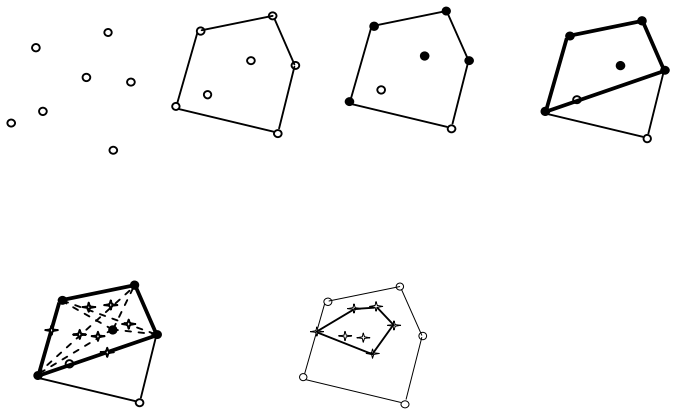


Fig. 6. Geometric crossover + selection = convex search.

"Good" Observers Enhance SGA Exploration

Christophe Philemotte

cphilemo@iridia.ulb.ac.be
http://iridia.ulb.ac.be/~cphilemo/
IRIDIA, ULB CP 194/6, 50 Avenue F. D. Roosevelt, 1050 Brussels, Belgium

Abstract. Most metaheuristics try to find a good balance between exploitation and exploration to achieve their goals. The exploration efficiency is highly dependent on the cardinality and ruggedness of the search space. A metaheuristic like the Simple Genetic Algorithm (SGA) can suffer a lot when traversing very large landscapes, especially deceptive ones. The approach proposed here improves the exploration of the SGA through the use of behavioural information of the SGA itself. Behavioural information on the SGA is obtained through a number of competitive processes which we refer to as "observers". The new metaheuristic we investigate, trains the observers for a specific time and then decides which of them is the most suitable to solve the whole problem. Concretely, a second evolutionary stage has been added to evolve observers for the SGA. These observers transform the cardinality and ruggedness of the search space through a simplification of the genotype. To test the proposed approach, we chose some difficult problems such as the Hierarchical IF-and-only-iF (HIFF). We obtained very good results, since we seriously improved the adaptive capacity of the SGA. Based on the current results, we are encouraged to continue in this way.

1 Introduction of the research Area

The essential part of research about evolutionary algorithms and optimisation in general is the discovery of mechanisms to improve the search. Many metaheuristics and hybridisations are invented and compared on their capacity to traverse the search space. Most metaheuristics try to find a good balance between exploitation and exploration to achieve their goals [1]. The exploration efficiency is highly dependent on the size and ruggedness of the search space. A metaheuristic like the Simple Genetic Algorithm (SGA) can suffer a lot when traversing very large landscapes, especially deceptive ones [2, 3]. Each metaheuristic proposes its own compromise on the balance between exploration and exploitation. For a wide search space, the SGA spends most of the time trying to discover a promising area in which to refine the search.

One approach when facing the problem of the space dimension is to discover some clever ways to reduce it. The notion of Intrinsic Emergence (IE), which wa originally inspired by the developments of Crutchfield and Mitchell, appears to be very helpful [4–9]. According to them, a "functional device" supplies some mechanical and non-human observations with added and profitable functionalities. Here an observer, by modifying the coding of the solutions, helps te optimisation process in order to simplify and reduce the search space. From a cognitive point of view, a human does not take into

consideration all detail to perform a task, such as finding the shortest driving path for a trip. He first concentrates on the global map and adopts a coarse level of observation to then focus on the details to refine the path. In the same idea, our approach improves the SGA through the use of high-level observers altering the whole process.

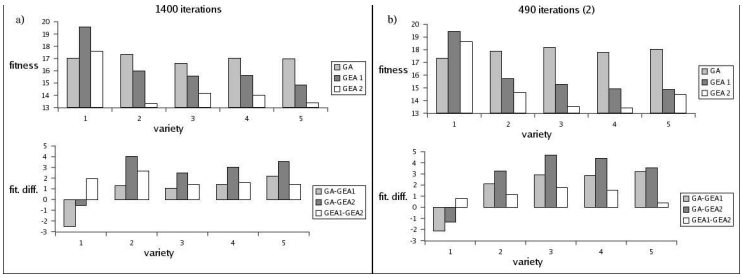


Fig. 1. The fitness score has to be minimised. (a) Results for experiment 3 (see [9] for parameters setup). Mean fitness (over 50 simulations) is plotting with respect to CA varieties. The first graph shows the results obtained after the GA, the GEA masking phase (GEA1) and the complete GEA (GEA2). The second graph gives the differences between these 3 functions: GA-GEA1, GA-GEA2 and GEA1-GEA2. (b) Results for experiment 4 (see [9] for parameters setup). Mean fitness (over 50 simulations) is plotting with respect to CA varieties. The first graph shows the results obtained after the GA, the GEA masking phase (GEA1) and the complete GEA (GEA2). The second graph gives the differences between these 3 functions: GA-GEA1, GA-GEA2 and GEA1-GEA2.

The papers [8, 9] give the main framework of our work in which we have illustrated the utility of the IE. In this work, we have examined in more details the concept of IE from an engineering point of view. We apply IE to the problem of finding a good CA implementation of a binary adder. The space cardinality can reach 10^{200}, making it hard for a classical GA to find a global optimum in decent time. To solve it, a classical GA was combined with another evolutionary process that looks for the best mask improving the GA. Based on the IE, an original way to observe the CA search space consists in masking some of the eight neighbours of all cells to be updated. The search space is considerably reduced because the mask alters the way the GA looks at the search space. In our framework, the mask and its effects are the functional device and the GA is the assisted system: the IE is implemented by this symbiosis mechanism which improves the classical GA exploration strategy. The improvements have been shown experimentally by comparing a single GA and the proposed IE implementation, the Genetic and Emergent Algorithm (GEA). The results obtained through several experiment sets have confirmed expected improvements: a gain in time or in fitness for all non-uniform cases (see Figure [1]). So, IE is a good way to reduce a search space and boost the adaptive capacity.

2 Your research and study

2.1 Goals

By design, metaheuristics are generic methods tackling a wide class of problems. Is it possible to optimise metaheuristics in a generic way? This is the long term goal of our research. Behind this goal we can determine more concrete questions like the role of observation, the utility of emergence or the relative need of information in a representation. Because we need a starting point, we focus first on Evolutionary Algorithms (EA) and especially the SGA for its simplicity [10, 2, 3]. The main question becomes: does SGA need the whole information contained in the chosen encoding to work efficiently? Indeed, the level of information depends on the nature of the problem, specially those of the hierarchical class. So, inspired by the IE concept, a macroscopic observation offers a way to inform the SGA and helps it to work better. Finally, we can summarise the goals through several questions. We order them by top down approach and their priority. We also separate them in two main categories:

1. Applications and design:
   - Does the addition of observers improve the performance of SGA?
   - Can we use macro observation to collect useful information?
   - Which level of details is needed to observe efficiently?
   - How can we find good observers?
   - Does this generic method help SGA to tackle harder problems and classical combinatorial problems?
2. Fundaments and epistemology:
   - How can we interpret their effect on the search space?
   - Can we theoretically model these observers?
   - Can emergence be useful?
   - Can we apply the same approach to all metaheuristics?

2.2 Current Status

The work started from Bersini's work about the IE from a engineering point of view. The first idea has been fully completed by adding a second evolutionary stage leading to a greater coherence with the IE concept. The current implementation allows to tackle any SGA-compatible problem. Therefore, we have to test different class of problem [11, 12]. After first publications [8, 9], it has been decided to test this new "meta-SGA" on the H-IFF problem [13, 12]. This second version gave excellent results on the H-IFF. The algorithm has been also tested on Royal Road functions [14, 15] and the same efficiency was obtained. During the run of many experiments, we focus on the meaning of observing the landscape and on the explanation of the mechanism for future publication. With respect to our previous questions, we can already answer to some of them:

1. Applications and design:
   - *Does the addition of observers improve the performance of SGA?*
     A way is proposed. It affects the accessible state in the landscape with respect to an observer. The algorithm is a second stage over the SGA, a *meta-SGA*. This proposed approach gives very good result on hard-GA problems such as the HIFF problem. We still have to test it much more, particularly on combinatorial problems like Travelling Salesman Problem (TSP).

- *Can we use macro observation to collect useful information?*
  The macro observation consists in an evaluation of the efficiency of the SGA. We have experimentally found a function giving the best results. We need to investigate much more this question.
- *Which level of details is needed to observe efficiently?*
  A version of the algorithm has been designed to adapt this level of details. This adaptation property of the observer must not be provided a priori.
- *How can we find good observers?*
  The second stage uses a SGA to evolve the observers. It is more efficient than random search.
- *How can we interpret their effect on the search space?*
  The meaning of the observer is a directed sampling. We need to formalise this meaning and give a theory background to the algorithm.
- *Does this generic method help SGA to tackle harder problems and classical combinatorial problems?*
  It gives good result on hard-GA problem such as HIFF problem or Royal Road functions. We need to test it much more.

2. Fundaments and epistemology:
   - *Can we model theoretically those observers?*
     Those observers can be defined like operators in the Vose's theory [3]. It is not done at all for the moment.
   - *Can emergence be useful?*
     This approach is inspired by the IE. This concept seems exploitable in engineering. That opens new perspectives to understand the role of emergence in biological system. Our research strongthen the Cruchtfield and Mitchell's idea.
   - *Can we apply the same approach to all metaheuristics?*
     Because this approach reduces the search space temporally, it might to applied to other metaheuristics. This is not tested at all for the moment.

2.3 Future Planning

For the future, it is important to test the approach on a wide range of problems. This way, we will be able to compare results obtained by the EA without IE and best known algorithms, and to better understand the effect of our approach on the SGA and other EAs, its dynamics and the parameters tuning. This milestone is very important to give more credit to our work and also new ideas to improve its design. The second important future step is the theoretical study of our approach with the Vose's theory [3].

3 Results

3.1 Methodology

General approach Our algorithm offers a way to assist the exploration process of a given metaheuristic. An observer defines how the metaheuristic can look at the search space: it is like a clever lens used by the metaheuristic to explore the landscape. This additional help includes two main steps (see Algorithm [1]): the efficiency evaluation `EvaluateMetaheuristicRunWithObserver(...)` and the building `BuildAnotherObserver(...)`.

Algorithm 1 Canonical Skeleton of the proposed approach

$O(0) \leftarrow$ `BuildAnotherObserver()`
$t \leftarrow 0$
while termination conditions not met **do**
  $SO \leftarrow$ `EvaluateMetaheuristicRunWithObserver`$(O(t))$ {SO temporarily stores the score of the Observer $O(t)$}
  $O(t+1) \leftarrow$ `BuildAnotherObserver`$(O(t), SO)$
  $t \leftarrow t+1$
end while
return best solution found by observed metaheuristic {i.e. a solution of the treated problem, not an observer}

The first step evaluates the quality of an observer: according to the solutions found, the observer is rewarded as a function of its quality that it helps the metaheuristics to obtain (`EvaluateMetaheuristicRunWithObserver(...)`) and the reward is stored in a temporary variable SO. This evaluation is a function of the quality of the solutions obtained when such an observer is applied. The way this quality is defined in the SGA case is discussed below.

The second step builds a new lens i.e. a new observer $O(t+1)$ with respect to the previous one $O(t)$. The next observer construction is directed by the score SO (`BuildAnotherObserver(...)`) and depends on the encoding and its effect in the problem search space.

Optimisation of observers Clearly a second search process is engaged here but, this time, in the space of the observers. Again, any search algorithm could be used for improving the observer solutions. In a preliminary attempt to evolve Cellular Automata, the observers were generated in a random and unguided way [7]. In a successive attempt, an evolutionary search was proposed and tested [8, 9]. Here we follow this second evolutionary search in the space of the observers. It is an iterated process to optimise the way the solutions are coded and therefore to improve the progression of the SGA [10, 2, 3] in this "observed" space (see Algorithm [3]). The termination criterion depends on the chosen search algorithm in the space of the observers. The current implementation of this search is again based on a SGA (see Algorithm [2]) transforming the whole algorithm as an intertwining of two evolutionary stages (through `Evaluate(...)`). The `Evaluate(...)` function corresponds to the `EvaluateMetaheuristicRunWithObserver(...)` and the observer evaluation function is then defined by [9]:

$$f_{obs}(O(t)) = \sqrt{\overline{f_{sol}} \times \max(f_{sol})}\,, \tag{1}$$

where $\overline{f_{sol}}$ is the average of the individual fitness over the population and $\max f_{sol}$ is the maximum of these fitness values. An observer should be as good as it allows a population of average good quality and an excellent best individual. And the `Select(...)`, `ApplyReproductionOperators(...)` and `Replace(...)` functions of the Algorithm [2] compose the `BuildAnotherObserver(...)` from the Algorithm [1].

Encoding of the observer The observer aims at improving the SGA by altering the landscape, hence the coding of the individuals. It aims at sampling the search

Algorithm 2 Evolutionary Implementation of our meta-(metaheuristic)

Generate initial observer population $O(0)$
Generate initial solution population $P(0)$
$t_1 \leftarrow 0$
while termination conditions not met **do**
  `Evaluate`$(O(t_1), P(0))$ {Defined in Algorithm [3]}
  $O'(t_1) \leftarrow$ `Select`$(O(t_1))$
  $O''(t_1) \leftarrow$ `ApplyReproductionOperators`$(O'(t_1))$
  $O(t_1 + 1) \leftarrow$ `Replace`$(O(t_1), O''(t_1))$
  $t_1 \leftarrow t_1 + 1$
end while
return best solution found in P with best found observer O

Algorithm 3 `Evaluate`(O: an Observer, P: a solution population) implementation for an EA

$Q(0) \leftarrow$ `ObservePopulation`(P, O)
$t_2 \leftarrow 0$
while termination conditions not met **do**
  `Evaluate`$(Q(t_2))$
  $Q'(t_2) \leftarrow$ `Select`$(Q(t_2))$
  $Q''(t_2) \leftarrow$
  `ApplyReproductionOperatorsWithObservation`$(Q'(t_2), O)$
  $Q(t_2 + 1) \leftarrow$ `Replace`$(Q(t_2), Q''(t_2))$
  $t_2 \leftarrow t_2 + 1$
end while
return efficiency of observation and best solution, i.e. $f_{obs}(O)$

space and restricting it to the most informative areas. When coding the individuals as a chromosome, the observer effect is to structurally group the genes. In a group, all genes are assigned with the same value (allele). The search space is projected onto a hyperplane. The observer defines a function that maps a gene location onto a group location. Each gene is assigned to a specific group. The genes are glued in groups of gene even if they are not contiguous in the original representation. This assignment and mapping functions partially reconstruct the encoding of the treated problem. The function works as described in Algorithm [4] and [5]. The `map(...)` function computed the boolean *AND* (noted by $\wedge$) of a gene locus i and a given observer O, both encoded as binary number. The `BuildGroup(...)` computes the genes group, i.e. the observer effect on problem encoding. Theses functions compose the `ObservePopulation(...)` from Algorithm [3]. Therefore, an observer is encoded in a bits string of length $l_O = \lceil \log_2(l_C) \rceil$ and defines the number of group as 2^{N_1}, where N_1 is the number of bit **1** in the observer chromosome. In the case of an observer of N_1 number of bits 1, the reduction in size of the search space is from 2^{l_C} to $2^{2^{N_1}}$. Since N_1 goes from 0 to l_O, depending on the observer, the size of the search space goes from 1 to l_C. The greater the number of 1 in the observer coding the less this observer reduces the precision of the coding and consequently the size of the search space.

Algorithm 4 $l = \texttt{map}(i, O)$

i {a given binary number for a gene locus}
O {a given binary number for a given observer}
$l \leftarrow i \wedge O$ {l is a binary number for the group locus}
return l

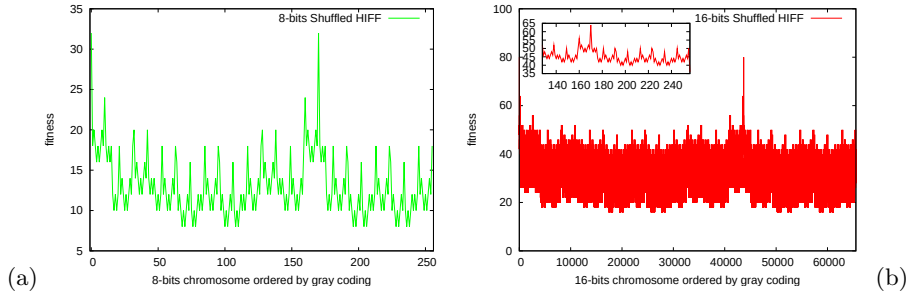
Algorithm 5 `BuildGroup`(C: a chromosome of length l_C, O: an observer)

for all i such that $0 \leq i \leq l_C$ **do**
  $l \leftarrow \texttt{map}(i, O)$
  $C(i) \in G(l)$
end for

3.2 Experiment

Our algorithm is tested on the Shuffled Hierarchical IF-and-only-iF (HIFF) problem which is a maximisation problem [13, 16]. HIFF is the canonical version of a specific class of problems modelling the interdependency between building blocks [11, 16, 12]. This kind of problems are very interesting for different reasons. They are tools to better understand the compositional and accretive mechanism of a SGA [16, 17]. They are also representative of hierarchies that we could identify in natural dynamic system. And HIFF is very difficult for the SGA, especially the shuffled version. It does not explore efficiently the search space and loses itself in the "fractal" landscape of HIFF (see Figure [2]): SGA does not succeed to traverse through multiple local optima efficiently

[16]. Different specific algorithms have been designed to tackle HIFF problems like SEAM [16, 17], Hierarchical-GA [12, 18] or Compact Genetic Codes [19].



(a) (b)

Fig. 2. The fitness landscape results from the recursive definition of modules and sub-modules. (a) is the landscape for the 8-bits instance and (b) for the 16-bits one. They give us a good idea of the landscape ruggedness and the number of local optima.

The tackled Shuffled HIFF (SHIFF) is $l_C = 128$ bits long and generated according to the Watson's implementation[1]. This problem size gives a maximum fitness of 1024 for both optima, i.e. binary strings of only **1** or **0**. Following what was explained before, the observers are encoded by a bit string of length $l_O = \lceil \log_2(128) \rceil = 7$. We compare our approach with two other algorithms: a single SGA and a SGA with randomly generated observers. The comparison is made through the following performance evaluation, i.e. what best solution can be found following a given number of fitness function evaluations. This is the most logical way to compare SGA without and with the addition of observers, both random and evolved.

3.3 Results

In this section, we experimentally answer to the following questions:

1. Does the addition of observers improve the performance of SGA?
2. Can we use macro observation to collect useful information?
3. Which level of details is needed to observe efficiently?
4. How can we find good observers?
5. Does this generic method help SGA to tackle harder problems and classical combinatorial problems?

We compare the basic SGA with RO-SGA (in this case the observers are randomly generated) and EO-SGA (in this case the observers are being evolved by a second SGA). The main criterion of interest when comparing algorithms is their scalability, i.e. the impact of the problem size on the computation time. For difficult problems, the cost in computation time is mainly determined by the evaluation of fitness values. This is why progresses in the GA community often aim at reducing this number. We

[1] The watson's source code is available on `http://www.cs.brandeis.edu/ richardw/hiff.html`.

	SGA	RO-SGA	EO-SGA (set 1)	EO-SGA (set 2)
p_{sol}	$\{20, 200, 2000\}$	$\{20, 50, 200\}$	$\{1, 5, 10, 20\}$	$\{20, 200, 2000\}$
g_{sol}	$\{5, 10\}$	$\{1, 10, 100\}$	$\{1, 2, 5\}$	$\{5, 10\}$
p_{obs}	-	$\{1\}$	$\{1, 2, 5\}$	$\{2, 5, 10\}$
g_{obs}	-	$\{1, 2, 5\}$	$\{1, 2, 5\}$	$\{2, 5, 10\}$
$\max(n_f)$	$p_{sol}g_{sol}$	$p_{sol}g_{sol}g_{obs}$	$p_{sol}g_{sol}p_{obs}g_{obs}$	
$\overline{n_f}$	$P_{cross}P_{mut}\max(n_f)$	$\sim \max(n_f)$	$P^2_{cross}P^2_{mut}\max(n_f)$	
$s*$	Fig. [3]	Fig. [4.b]	Fig. [5.b]	-
$o*$	-	Fig. [4.a]	Fig. [5.a]	-
$\overline{\mathbf{1}}$	-	Fig. [6.b]	Fig. [6.a]	Fig. [6.a]

Table 1. Parameters sets for each algorithm (SGA, meta-SGA and random meta-SGA): solution population p_{sol}, solution generations g_{sol}, observer population p_{obs} and observer generations g_{obs}. With respect to these parameters, we also calculate the maximum number of evaluations $\max(n)$ and its probabilistically weighted estimation $\overline{n}$. The crossover probability is $P_{cross} = 0.7$ and the mutation probability is $P_{mut} = 1/128$ (the best ones frome [16]). Both are used for each SGA. The bottom rows summarised the content of the figure.

therefore measure the best solution fitness $s*$, which is computed by f_{hiff} (see [16]), as a function of the number of evaluations of n_f. The fitness of the best observer $o*$, which is evaluated by f_{obs} (see Equation (1)), is also measured in order to investigate the second question above. For each parameters set (to be described below), the obtained results are summarised by drawing two statistics as a function of the number of evaluations:

- a measure of central tendency by the calculation of the median which is marked by a small dark horizontal bar,
- and a measure of statistical dispersion by the calculation of the extrema, the first and third quartiles which are represented by a candlestick.

These measures provide a fair comparison between the algorithms performances and give answers to most of the questions raised above except the third one. This question will be investigated later through the study of the average number of bits **1** in the best observer chromosome. The more **1**s we have, the more detailed is the observation. This number provides some indications on the capacity of the algorithm to autonomously tune the observation level, i.e. the information compression of the search space still sufficient enough to efficiently travel it.

Each SGA uses the elitist heuristic and fitness proportion selection. Both the populations of the solutions and the observers are initialised in an uniformly distributed way. The probabilities of crossover and mutation are respectively 0.7 and 1/128. For each experiment, we run the algorithms 30 times on 50 different instances of SHIFF. The SGA needs two parameters: the size of the solutions population p_{sol} and the maximum number of generations g_{sol}. The EO-SGA is composed of two intertwined evolutionary stages and needs two additional parameters: the size of the observers population p_{obs} and the maximum number of observer generations g_{obs}. However, in the case of the RO-SGA, only one more parameter is needed: the maximum number of random generation of observer g_{obs}. The used parameters sets are listed in Table [1].

As shown in [16], the SGA can not exceed a score around 400, even for higher evaluation numbers. By adding some heuristics like “deterministic crowding diversity

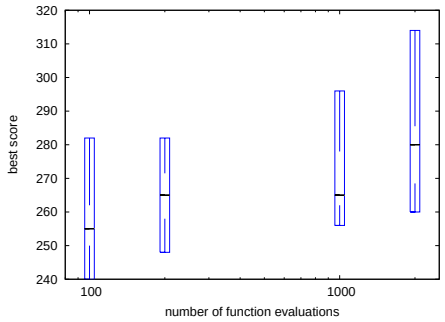
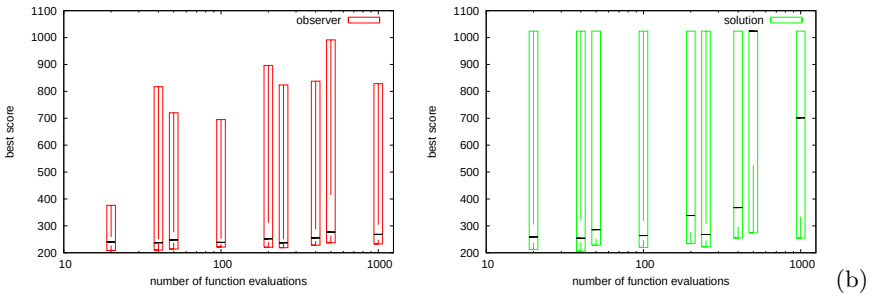


Fig. 3. Results for basic SGA. The statistical measurements of best obtained solution score $s*$ are plotted with respect to the number n_f of fitness evaluations. The SGA cannot exceed a score of around 300 to 400 even for higher evaluation numbers. By adding some heuristics like "diversity maintenance", Watson has experimentally shown that the SGA is unable to escape from local optima and hardly reaches a score of around 700 [16].

maintenance", Watson has experimentally shown that SGA is unable to escape from local optima and hardly reaches a score around 700. We obtain similar results for the basic SGA (see Figure [3]). The small results dispersion indicates how hard it is to travel through the SHIFF fractal landscape for a basic SGA.

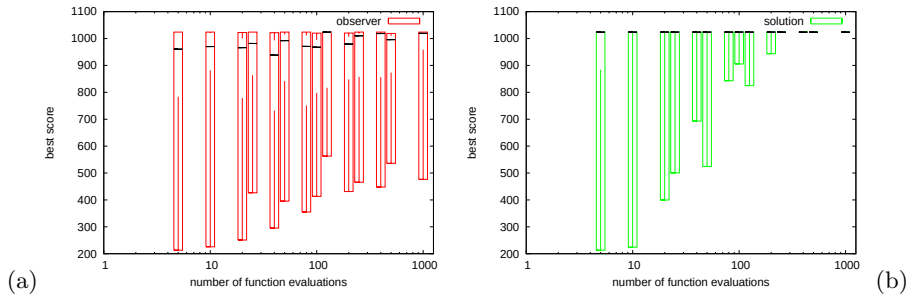


(a) (b)

Fig. 4. Results for RO-SGA. The statistical measurements of the best obtained observer score $o*$ (a) and of the best obtained solution score $s*$ (b) are plotted in function of the number n_f of fitness evaluations.

The RO-SGA provides a wide score range (see Figure [4.b]). The observers are randomly generated and they do not adapt their zoom properly. Due to the uniform distribution, each observer resolution is generated with the same frequency. The observer score $o*$ provides information on how good the observed population is with respect to f_{obs} definition (see Figure [4.a]). For higher evaluation numbers, the median

is bigger: SGA spends more time to explore the observer search space and increases the chance to find a good one.



(a) (b)

Fig. 5. Results of EO-SGA (set 1). The statistical measurements of the best obtained observer score $o*$ (a) and of the best obtained solution score $s*$ (b) are plotted in function of the number n_f of fitness evaluations. Beyond $n_f \sim 500$, the EO-SGA found the maximum solution score with a very small dispersion, which is reduced to zero at $n_f \sim 10^3$.

Beyond around 500 evaluations, the EO-SGA efficiently computes the maximum solution score (see Figure [5.b]). Around 1000 evaluations, no more dispersion is observed (see Figure [5.b]). By examining the Figures [5.a] and [5.b], we notice that the EO-SGA is capable of finding the maxima following 10 evaluations. The way the observers evolve depends on their quality evaluated by the evaluation function f_{obs} given above (see Equation (1)). After 1000 evaluations, the EO-SGA finds an adequate observer and the resulting observed population which contains the two optima. The convergence of the dispersion of $s*$ and $o*$ seems to support the original intuition that observer could really help the SGA to efficiently travel through the SHIFF rugged landscape. Compared to the random version, we also show the utility of evolving the observers as a function of how well they progress and how well they adapt to the search space of the problem. So, we answered to the questions above apart the third one:

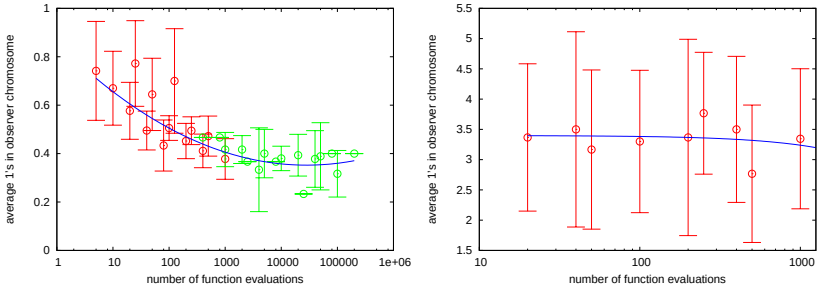
- Our EO-SGA drastically improves the basic SGA.
- The observer provides an adapted way to "observe" the landscape so as to better traverse it. With observer, only useful information is retained to explore the SHIFF search space.
- Our evaluation function f_{obs} rightly determines the quality of an observer. The evolution of the observers by means of this second SGA is an adequate and faster way to find them.
- The whole algorithm seems indeed to be well adapted to hard combinatorial problems.

Several specific algorithms have been implemented to tackle hierarchical problems such as de Jong's HGA [12] or Watson's SEAM [16, 17]. Is our algorithm a competitive approach? We compared our algorithm performances with the HGA and SEAM ones extracted from [18, 16]. Considering a 128-bits SHIFF, our EO-SGA is ten times faster than HGA and one thousand times more efficient than SEAM:

	SEAM	HGA	meta-SGA
evaluations number n_f	$\sim 10^6$	$\sim 5.10^4$	$\sim 3.10^3$
whole processing time	-	30s	0.2s

Over 1000 different instances, the EO-SGA finds the two optima in at most 3000 iterations and run in 0.2s on average[2]. Why is SHIFF such an easy problem for EO-SGA?

We propose an first explanation through the study of the mean of **1**s of the best found observer chromosome. If the observer is encoded by only **0**s, the 128 bits, which represent the solution chromosome, compose one single group. Only two solutions are possible and the search space falls from 2^{128} down to 2^1. With one **1**, the bits are distributed in two genes groups and only four candidate solutions are possible. Examining the Figure [6.a], the best observers contain around 0.5 bit on average. The more time the EO-SGA has, the better and coarser the observers are. Figure [6.b] shows that by randomly generating the observers, the number of **1** N_1 i.e. the precision of the observer is equal to the mean over all uniformly distributed 7-bits string, $3.5 = 7/2$. These results explain why and how the SHIFF problem turns out to be such an easy one for our approach. We also better understand the role of the precision of the observer, the third question raised above.



(a) (b)

Fig. 6. Average of **1**s in best observer chromosome. (a) For both parameter sets EO-SGA. (b) For the the parameter set of RO-SGA.

4 Achievements

Starting from the IE [4–9], we have designed a new approach called EO-SGA. Its main goal consists in improving the SGA exploration process. The principle is based on the evolution of observers which alter the way the SGA looks at the landscape [8, 9]. This alteration changes the encoding and reduces the search space to more informative subsets. We have shown the benefits gained by experimenting our algorithm on the SHIFF problem [13]. The results confirm the improve of the SGA and outperform specific

[2] The computer is a 1GHz AMD processor. The SGA implementation is not especially optimised. For instance, dynamic libraries are used.

algorithm such as the Watson's SEAM [16]. The added stage allow to find the best hierarchical level, i.e. the highest, to understand the strong genetic interdependency. So, provided theses observers, the SGA restricts its search at this level and quickly finds the best solutions due to the collapse of the search space size.

There are still many studies to achieve the thesis. That could be summarised in three main milestones: experiments, new designs, and theoretical study. New experiments should tackle more realistic problems such as Traveling Salesman Problem. Thes experiments will provide the necessary information to deeply understand our approach and improve its design. For instance, new designs should incorporate the coevolution of observers by allowing interaction among the population of solutions obtained by each observer. Finally, observers could be taken as genetic operator like crossover and mutation ones. We would try to provide a theoretical treatment of our EO-SGA. Some issues are still largely opened such as the generalisation of the approach and how well it extends to the machine learning framework in general and the whole family of metaheuristics.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Survey **35**(3) (2003) 268–308
2. Mitchell, M.: An introduction to genetic algorithms. MIT Press, Cambridge, MA, USA (1996)
3. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge, MA, USA (1998)
4. Steels, L.: Towards a theory of emergent functionality. In: Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats, Cambridge, MA, USA, MIT Press (1990) 451–461
5. Crutchfield, J.: Is anything ever new? considering emergence. In G. Cowan, D.P., Melzner, D., eds.: Integrative Themes. Volume XIX of Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley Publishing Company, Reading, Massachusetts (1994)
6. Crutchfield, J., Mitchell, M.: The evolution of emergent computation. In: Proceedings of the National Academy of Science. Volume 23. (1995) 103
7. Bersini, H.: Whatever emerges should be intrinsically useful. In: Artificial life 9, The MIT Press (2004) 226–231
8. Philemotte, C., Bersini, H.: Intrinsic emergence boosts adaptive capacity. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, New York, NY, USA, ACM Press (2005) 559–560
9. Philemotte, C., Bersini, H.: Coevolution of effective observers and observed multiagents system. In: Advances in Artificial Life, 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005, Proceedings. Volume 3630 of Lecture Notes in Computer Science., Springer (2005) 785–794
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional (1989)
11. Forrest, S., Mitchell, M.: What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. Machine Learning **13** (1993) 285–319
12. de Jong, E.D., Thierens, D., Watson, R.A.: Hierarchical genetic algorithms. In Yao, X., Burke, E., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J., Kabán, P.T.A., Schwefel, H.P., eds.: Parallel Problem Solving from Nature

- PPSN VIII. Volume 3242 of LNCS., Birmingham, UK, Springer-Verlag (2004) 232–241
13. Watson, R.A., Hornby, G., Pollack, J.B.: Modeling building-block interdependency. In: PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, London, UK, Springer-Verlag (1998) 97–108
14. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and ga performance. In Varela, F.J., Bourgine, P., eds.: Proceedings of the First European Conference on Artificial Life, Cambridge, MA, MIT Press (1992)
15. Forrest, S., Mitchell, M.: Relative building-block fitness and the building block hypothesis. In Whitley, L.D., ed.: FOGA, Morgan Kaufmann (1992) 109–126
16. Watson, R.A., Pollack, J.B.: Compositional evolution: interdisciplinary investigations in evolvability, modularity, and symbiosis. PhD thesis, Brandeis University (2002) Adviser-Jordan B. Pollack.
17. Watson, R.A.: A computational model of symbiotic composition in evolutionary transitions. Biosystems **69**(2–3) (2003) 187–209
18. de Jong, E.D., Watson, R.A., Thierens, D.: On the complexity of hierarchical problem solving. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, New York, NY, USA, ACM Press (2005) 1201–1208
19. Toussaint, M.: Compact genetic codes as a search strategy of evolutionary processes. In Wright, A.H., Vose, M.D., Jong, K.A.D., Schmitt, L.M., eds.: Foundations of Genetic Algorithms, 8th International Workshop, FOGA 2005, Aizu-Wakamatsu City, Japan, January 5-9, 2005, Revised Selected Papers. Volume 3469 of Lecture Notes in Computer Science., Springer (2005) 75–94

Evolving Expressive Performance through simulating artificial agents' interaction

Qijun Zhang

qijun.zhang@plymouth.ac.uk
Interdisciplinary Centre for Computer Music Research (ICCMR),
University of Plymouth, UK

Abstract. This PhD project focuses on the usage of Evolutionary Computation (EC) in expressive music performance research. We aim at building a computational model that co-evolves agent performers and agent listeners. Through these autonomous agents' interactions in the society, we are hoping to observe the emergence of shared repertoire of expressive music performance, and ideally this has some similarity with human performances. The work has been done is the first stage of this PhD project, in which we have implemented a system that uses Genetic Algorithm (GA) to evolve hierarchical time vs. amplitude matrices for music interpretation. The fitness for the GA is decided by how well the interpretation fits some rules associated with the piece's structural characteristics.

1 Introduction of the research area

The achievements and potential of two fields: (1) applying EC in studying musicological problems, and (2) computational modelling of expressive music performance, have inspired the idea of this PhD project. In this section, we will give an overview of the main concerns and some important works in these two fields, which are the context of this project's design.

1.1 Application of Evolutionary Computation in exploring musicological problems

There have been growing applications using Evolutionary Computation (EC) in music, which can be categorized into three approaches : the engineering, the creative and the musicological. [1] We here focus on the musicological approach, with which simulations are run with built models to demonstrate some theories about the origin and evolution of musical behaviours. It is the path this project follows.

Several facts inspire the musicological approach. The first perhaps, is the lacking of prehistoric proof or enough historic records, without which we cannot deduce convincing theories about the origin and evolution of music. Second, with computer simulation, it's possible to run simulation of generations in very short time, and also to observe certain factor's effect in the evolution by controlling related parameters. Therefore, evolutionary simulation provides the possibility to prove some hypothesis or theories about musicological problems, and helps us to understand them better.

Our system design has taken inspiration from two systems that have been using this approach. The first is Peter Todd and Gregory Werner's system that studies the evolution of musical tunes through coevolving artificial music critics and music composers in a community. [2] And the second is Eduardo Miranda's mimetic model, which managed to evolve a shared repertoire of tunes through imitations of autonomous agents from scratch. [3]

1.2 Computational modelling of expressive music performance

Music performances with proper expressions are defined as expressive music performances. "Proper expressions" is what makes music interesting and sound alive. These 'proper expressions' contribute to highlight the music's structure. Expressive music performance attracts investigations from various perspectives, such as physical, acoustic, physiological, psychological, social, artistic and so on. Among these studies, researchers in computer music area mainly focus on computationally modelling of expressive music performance, which is also the goal of this PhD project.

In the context of western tonal music, there is a commonly agreed notion that expression is conveyed in a music performance by performer's delicate deviations of notated score. Therefore, expressive music performance research is the study of why, where and how these deviations take place in expressive performances. Computational modelling of expressive music performance has the aim of building a bridge that connects the properties of a musical score and performance context with the physical parameters in the performance (such as note's timing, loudness, tempo, articulation and so on). [4] A good model might be either able to practically produce expressive performance, or predict what it is like in real performance. Of course we have to bear in mind that those predictions cannot be expected to always fit the real performance, because of the existence of various interpretations, which is true even when the same performer plays the same piece for more than once.

The following are the main strategies used in modelling expressive performance, and we will briefly introduce the most important published work for each of them.

- *Analysis-by-measurement.* Studies falling into this category are based on the analysis of deviations measured in recorded human performance, and aims at recognizing regularities in the deviation patterns and describing them by mathematical models. [5] The approaches used include statistical models and mathematical models, [6] as well as neural network [7] and so on.
- *Analysis-by-synthesis.* This method normally starts with hypotheses from observations of real performance and uses them to synthesize performance. The most important representative is the KTH rule system that has groups of comprehensive performance rules. [8]
- *Machine learning.* In a noteworthy application of artificial intelligence, Gerhard Widmer and his co-workers in Austria have been using machine learning and data mining technique to have the computer to discover regularities in large amounts of real performance data. This paradigm has the potential of discovering more generalized principles and interesting rules. [9]
- Lastly, there is Manfred Clynes' microstructure or pulse set theory which can be categorized as *model from intuition.*Clynes proposed that there exists particular timing and accentuation pattern (or to say microstructure) for specific composers. Consequently the best way to interpret a composer's music perhaps is to discover and follow this pattern. [10] Clynes has quantified these microstructures using the

concept of hierarchical "pulse set", which will be introduced in later section. We employ the idea of hierarchical pulse set to represent performance profile in our project.

As a summary, achievements in computational modelling of expressive performance affirm the promise of this field, and meanwhile promote further research on related topics. This PhD project is one of these attempts. At the same time, considering musical performance is such a cultural phenomena and has strong social context, we feel it would be feasible and interesting to use evolutionary simulation of agents' interactions within a community, for observing the emergence of shared expressive performance repertoires.

2 Research and study

2.1 Research goal

In keeping with most of the modelling work described above, one of our research goals is to build common performance principles for expressive music performance. In addition, with the belief that these common principles have been developed or evolved from a much more dispersed and irregular status, and also emerged from social interactions of performers and listeners, we are interested to prove this idea by simulation with computer. We have conceived the simulation model of this project, which is through coevolution of artificial performer and artificial listener in a community, by their evaluation or imitation of others. We are designing the system in hope of observing following results from the simulation:

- Emergence of general agreement on performance principles. In the design, this agreement is about the common way of using a collection of rules, the reason being explained in next section.
- Emergence of similar but not the same performance profiles for a piece, which show some similarity with real performance by human performers.
- Emergence of "perfect" performers and listeners. An agent performer can be defined as "perfect performer" when firstly its performance is highly evaluated by most of the audience, and secondly its evaluations for other performers' performance are always the same as most of the audience's, which happens the same with perfect listeners. If so, we can consider these performers represent the dominated expressive performance profile at that time.
- Built upon above results, the most interesting and difficult topic that needs further consideration is, how general is the evolved evaluation rules and performance profiles. It's perhaps true that it is possible to simulate the evolution of a performance profile once for only one piece, but we are interested to see if it also works well with another piece of the same style or same composer. If not, we would like to find out what are the strategies of transferring it to fit another similar piece. And what about for quite different pieces?

2.2 Current Status

Several issues require solution or hypothesis before we start the evolution simulation. Clearing up these problems is actually what has been completed so far.

Our first concern is, what are the performance principles we aim to get from the simulation? We consider it infeasible and inefficient to let the agent construct these principles from scratch. Instead, it is our hypothesis that, various performance profiles are derived from assigning various priorities or weights to those rules, along with implementing them differently in details. So we are hoping to evolve shared usage of existing principles which may result in similar performance profiles as what happens in reality. Recalling that expressive music performance mainly serves to highlight musical structure, our solution is to compose a principles' pool with several descriptive rules. [11] These rules explain some associations between certain structural characteristics and corresponding features of performance. And Second comes is the representation of performance profile for a piece. The one used in our system is hierarchical pulse set, which can be defined as a matrix of duration and amplitude values that decide the deviations of the physical attributes of musical notes. For three reasons we feel the concept of "hierarchical pulse set" proposed by Manfred Clynes fits what we need. Firstly, the choices of notes duration and amplitude significantly influence the expressive quality of a music performance, although not fully. Secondly, the hierarchical nature of pulse sets matches important features of most music genres; e.g., the notions of grouping and hierarchical structures. Finally, we regard hierarchical pulse sets as rather compact and informative forms for generative music interpretation.

In order to implement the above, we have designed an experimental system where we adopted Clynes's notion of pulse set as the representation of performance profile. Then, we tested the possibility of evolving good pulse set from randomly initiated ones using a GA. The fitness value of a pulse set is decided by its wellness or ability to highlight musical structures, when we use it to interpret a piece of music (in this case a quantised MIDI file with flat velocity). The fitness function uses rules derived from research into the cognition of musical structure and generative rules of expressive music performance. This system demonstrated that it is possible to evolve suitable pulse sets for musical interpretation according to the musical structure and the user's preferred interpretation principles. The "excellent" pulse sets evolved by the GA have shown diversity and also commonality. This could be observed both objectively and subjectively. This is necessary preparation before the next step of the project.

2.3 Design of the interaction model

As mentioned in the above sections, two types of agents, performer and listener are designed to interact within the society. Each agent has the ability to listen to and evaluate a performance according to its own combination of structural rules. As a performer, an agent obviously has the function to produce a performance using its performance profile (i.e. hierarchical pulse set). Figure 1 shows the interactions of the agents in the community. It includes 1) the evaluation and imitation among performers. For example, a performer P1 uses its performance profile S1 to produce a performance M1. Another performer P0 compares M1 with its own interpretation M0 of the piece. If M0 fits its rules better than M1, P0 remains its pulse set S0 unchanged, otherwise it modifies S0. This modification is done by comparing the amplitude/duration curves of M1 and M0, and updating M0 in a more detailed way (which we skip here). Then we extract a new hierarchical pulse set S0' from the modified M0 to replace S0; 2) a discussion among the audience (including listeners and the performers who are not currently performing)to conclude an overall evaluation of the current performer.

After every performer has got an evaluation, we rank these values in order to decide the best and worst performers, which are respectively rewarded with high priority in

next round to give overall evaluations to others, or to be removed because of their low rankings. Similarly, the best and worst listeners are decided according to the closeness of their judgement with the overall evaluations to all the performers, and will be rewarded or removed. New agents can be created by agents' crossover and mutation, and the type of agent is to be randomly decided. A key issue needs to be explained

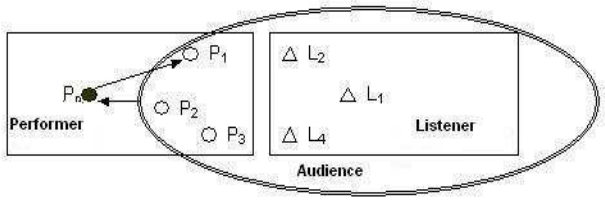


Fig. 1. Interaction model of agents

here is how to associate the performance principles (i.e.the way of combining structural rules) of an agent performer with its performance profile, as there certainly exists the bidirectional effect. For one thing, with certain musical structures, the preferred performance principles decide or generate the performance pattern for the piece. For another, when a more preferred performance profile appears, changes on the preference of performance principles should take place accordingly. Our solution for this is, firstly in the first iteration, we use GA to choose a good pulse set for every performer according to its structural rules. Then during the simulation process, whenever the performer has to modify its pulse set (because it finds better-preferred performance by others), we modify the combination of its structural rules accordingly.

2.4 Study

I have started my PhD since October 2004, so this is my second year. The time of my PhD is supposed to be three years plus one year writing up. As the programme I am in is MPhil/PhD, I am at the moment applying transfer from MPhil stage to PhD study.

3 Results

3.1 Methodology

The experiment I have done so far is with a system that uses Genetic Algorithm (GA) to evolve pulse set for music interpretation. The reasons to employ GA instead of constructing pulse set according to performance principles are as follows. Firstly, the critical problem of rule-based system of expressive performance is the difficulty of combining those complex rules. This is not what we are mostly interested in. Nevertheless, we undoubtedly need guidance from musically meaningful rules (e.g., musical structure, etc.) to evaluate whether a pulse set works well as a performance profile for

certain piece. For this purpose, our approach is to design descriptive performance principles without quantified regulations. And then we employ GA, whose fitness function is informed by these principles, to select and evolve suitable pulse sets, starting from randomly generated sets. In this sense, the usage of GA is ideal here because otherwise it would be hard to design manually a decent performance profile based on such descriptive principles. And furthermore, GA can evolve different and suitable pulse sets for the same piece, which will be demonstrated in the results of our experiments. This diversity is a noticeable phenomenon in real performances, and also a prerequisite for the next stage of our research.

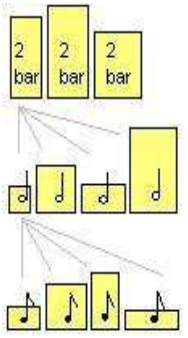
In order to explain this system, this section is organised as follow: firstly we introduce the notion of pulse set and how it is used in our system to interpret music. Then we present the rules that we have used to assign fitness values to different pulse sets. Next, we present the evolutionary procedure, followed by a demonstration and a discussion.

3.2 The Notion of Pulse Set

Representation and parameters of a pulse set
The inputs for our system are: (1) a quantised and flat MIDI file (without any expressive interpretation and the notes have exact durations as written on score), and (2) a hierarchical pulse set, which is a matrix of time vs. amplitude wraps. The process of interpreting the music with hierarchical pulse follows an algorithm devised by Clynes [12], as depicted in Figure 2a. In short, the values of hierarchical pulse set indicate the relative modification of the duration or loudness of the corresponding notes. Figure 2b gives a visual representation of a pulse set.

Representation	Meaning
8	The length of note at the lowest level
4 4 3	Number of elements in level3,2,1 (From the lowest level to the highest)
0.339 0.762 0.953 0.319	Level3 Amplitude Values
73 93 66 124	Level3 Duration Values
0.453 0.798 0.498 1.333	Level2 Amplitude Values
62 103 114 118	Level2 Duration Values
1.398 1.476 1.864	Level1 Amplitude Values
73 121 120	Level1 Duration Values

a.Genome representation



b.Visual representation

Fig. 2. Representation of a pulse set

Implementation of changes
Taking pulse set in Figure 2a as an example, the smallest unit of the music to be operated on is eighth note. Four eighth notes form one element in the intermediate level, and four such groups consist of one unit in the highest level. Therefore the entire length defined by this pulse set is three 2-bar groups (suppose there are 4 beats in one bar), comprising $4\times4\times3 = 48$ segments, each has the length of an eighth note.

Then, we can obtain individual amplitude and duration for each of the 48 segments, by multiplying parameters from different hierarchical levels.

After having time and amplitude patterns calculated from the pulse set, we globally change the file's play back tempos according to the temporal list. The loudness of a note is given by the amplitude information of its first smallest note component, which is done by assigning the note-on velocity for the note.When the amplitude or duration has the value of 1 or 100, the note's velocity or duration will be set as it is in the flat MIDI file.

3.3 Fitness Function based on Musical structure

In order to use structural rules for calculating fitness value, firstly we need an analysis of a piece's structure. In the present version of our system, we use David Temperley's software Melism, which performs several structural analysis such as metrical analysis, group analysis and harmony. Then, we design a few performance rules mainly based on Eric Clarke's generative rules for expressive performance.[11] We associate expressions in performance with the pieces structure features of grouping, accentuation and cadence. Thus the fitness value of a pulse set consists of 3 parts, *FitGrouper, FitAccent* and *FitCadence*.

a. FitGrouper
FitGrouper is obtained by a pulse set's fitness in relation to two rules, mainly concerning the notes' duration at group boundaries. These two rules are:
Rule 1: The time deviation of the last note of a group has either larger or smaller timing deviation than both the notes before and after it.
Rule 2: The last note of a group is always lengthened in order to delay the following note and signify the starting of a new group.
The value of FitGrouper is dependent on a pulse sets violation of above two rules. A parameter $numVio$ (initialised equal to 0) increases whenever the pulse set breaks either Rule 1 or Rule 2. If the number of groups in the piece is N_{group}, we define $FitGrouper = \frac{numVio}{N_{group}}$. The maximum value of *FitGrouper* is equal to 1.

b. FitAccent
FitAccent is an evaluation of how well the notes' loudness contour in an "interpreted" piece (i.e., after the flat MIDI file is modulated by a given pulse set) fits the metrical analysis. The rule used here is as follows:
Rule 3: Preference should be given to the contour of notes' loudness that has the most similar shape to the music's accentuation analysis.
Given two successive notes N0 and N1, *FitAccent* is produced by calculating: (i) The accentuation information (b0, b1) from the structure analysis, and (ii) The Velocity information (v0, v1) from the interpreted MIDI file. Because the accent value bi varies from 0 to 4, firstly, we normalize the velocity difference (v1-v0) to integers in the range of [-4, 4]. Then we assign a reward value between 0 and 1 to parameter x based on the difference between (v1-v0) and (b1-b0). The closer they are to each other, the larger the value assigned to x. If the number of notes in the piece is N_{note}, we define FitAccent $= \frac{\sum_1^{N_{note}-1} x_i}{N_{note}-1}$ $(0 \leq x_i \leq 1)$. As with *FitGrouper*, the maximum value of *FitAccent* is equal to 1.

c.FitCadence
FitCadence takes into account the strong chord progressions in a piece, which can also indicate group boundaries. While both *FitGrouper* and *FitAccent* work at the note level, *FitCadence* judges the performance features of a higher group level. The rule for calculating *FitCadence* is as follows:
Rule 4: Both segments corresponding to two chords in a cadence (e.g., VI, IV I, or DominantTonic, SubdominantTonic, respective) should be lengthened. Different weights are set for different categories of cadences because they have varying importance on a pieces structure.
As with the *FitGrouper*, the value of *FitCadence* is also decided by a pulse sets violation of Rule 4. And the pulse set will receive more penalties when it breaks the rule with stronger cadences. If the number of cadences in a piece is $N_{cadence}$, and we assign weight w_i to the i^{th} cadence, then $FitCadence = \frac{\sum_1^{N_{cadence}-1} w_i}{N_{cadence}-1}$ $(0 \leq w_i \leq 1)$. Alike the previous two fitness measures, the maximum value of *FitCadence* is equal to 1.
In the present version of our system, we define the total fitness of a pulse set to be the sum of *FitGrouper, FitAccent* and *FitCadence*. That is, Fitness = FitGrouper + FitAccent + FitCadence, with maximum value equal to 3.

3.4 Evolution procedure

It this section we introduce the procedure to evolve suitable pulse sets from scratch.

Genome representation of a pulse set
A pulse set is represented by a long string of real numbers in the same order as shown in Figure 2a. In this string, we separate lines with ";" and insert "," between elements in the same line. This makes it convenient to access and operate on parameters of different hierarchical levels. An additional number, either 0 or 1, is added at the end of an individual pulse set. This is used to indicate one of the possible two ways of applying a crossover operation, which will be clarified later.
As an example, the pulse set in Figure 2a is represented as follows (for the sake of clarity, we omitted Level 2 and Level 1); the additional number at the end of the string is equal to 0:
8; 4,4,3; 0.339,0.762,0.953,0.319; 73,93,66,124; ... ; 0

Initialisation of the first generation
The individual pulse sets of the first generation are randomly generated. For the moment, we have established that all pulse sets have 3 levels. All other pulse set values are randomly generated, including:
(1) The length of quickest note
(2) The number of elements in each hierarchical level
(3) Amplitude and duration values for each element in every level
(4) The additional number at the end of the string (for selecting the crossover operation)

Evolutionary algorithm
Each pulse set in a generation is used to interpret the music, as described in section 2, and a fitness value can be calculated according to the definition of fitness function introduced in section 3. Thus we can obtain an array of fitness values Fit0=f_0,f_1,...,f_n, where f_i is the fitness value of the i^{th} individual pulse set. Below we explain how

the offspring pulse sets for next generation are created using this fitness array. This procedure is as follows:

(1) Calculate the fitness values of the current generation $P0$

(2) Select parent candidates to compose of population $P0^1$

(3) Operating mutation on $P0^1$ and get population $P0^2$

(4) Operating crossover on pairs of pulse sets in $P0^2$ to get population $P0^3$

(5) Rank the fitness values of Generation P0 and $P0^3$ and the best half become Generation P1

We define the three genetic operations: selection, mutation and crossover scheme as following.

- Selection scheme

Based on Tobias Blickles comparative study of various widely used selection operators in GA (such as, tournament, linear and exponential rankings, and proportional)[13], we opted for using exponential ranking. This is because we wish to keep a certain degree of diversity in the evolutionary process and exponential ranking has proved to work well for this purpose. The pseudocode of our exponential ranking selection is as follows:

Exponential-ranking $(c, J_1, \ldots\ldots, J_N)$
  J←sorted population J according to fitness (first is the worst)
  S0←0
  For i←1 to N do
  $s_i \leftarrow S_{i-1} + p_i$
    Do
      For i←1 to N do
        r←random[0,s_N]
        $J_i \leftarrow J_k$ such that $s_{i-1} \leq$ r $< s_k$
    Do
  Return

- Mutation Scheme

Considering the hierarchical property of a pulse set, we have employed four different ways for operating mutation on a single pulse set. They are respectively:

Ma: Randomly modify every duration or amplitude values in the pulse set. The range of changes for amplitude is [-0.1, 0.1], and for duration is [-5, 5].

Mb: Append new duration and amplitude wraps or delete existing wraps from the end of the string. The number of added or removed elements is defined randomly, with the condition that the resulting pulse set is a valid pulse set. Also, if new elements are added, they are generated randomly. The length of the quickest note in the pulse set may be changed in this mutation, as we generate a new value for it randomly.

Mc: Swap the order of elements in the same level of the pulse set randomly, but keep the duration and amplitude wraps.

Md: Swap the order of hierarchical levels in the pulse set randomly.

An integer between 1 and 4 is generated randomly in each generation. This number defines which mutation method is used. For example, if we obtain 2, then only the first two mutation methods (Ma and Mb) are used in the respective generation. Then, whether to perform Ma or Mb to an individual pulse set is also decided randomly.

- Crossover Scheme

To maintain the evolved parameters in hierarchical levels, we only allow for segmentation and crossover at the positions of complete hierarchical levels. In this way, crossover

is actually where two parents pulse sets exchange some of their component levels. For clearer explanation, we use $X_1X_2X_3x$ and $Y_1Y_2Y_3y$ to represent the two pulse sets P_X and P_Y. X_n or Y_n refers to the n^{th} level of pulse set P_X or P_Y. x or y is the single bit at the end of a pulse set's representation, with the value of either 0 or 1.
Here we use the subtraction of $x - y$ (which can be 0,1,or -1) to decide the way to perform crossover with two pulse sets. This includes the choice of one-point crossover or two-point crossover, and also which levels of the parent pulse sets is the crossover operating on. The possible ways of crossover are shown in Table 1, dependent on the values of x and y.

Table 1. Crossover Schemes

xy	0	1
0	$X_1Y_2X_3x$ $Y_1X_2Y_3y$	$X_1X_2Y_3y$ $Y_1Y_2X_3x$
1	$X_1Y_2Y_3y$ $Y_1X_2X_3x$	$X_1Y_2X_3x$ $Y_1X_2Y_3y$

3.5 Demonstrations

As a didactic example, let us consider an excerpt from J.S. Bach's Prelude in C major, BWV 846 from WTK Book I. Figure 3 shows the excerpt with the structural analysis that is used to calculate the fitness value.

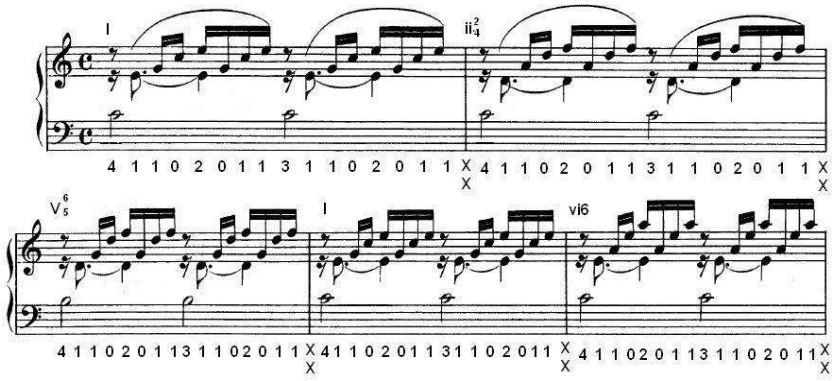
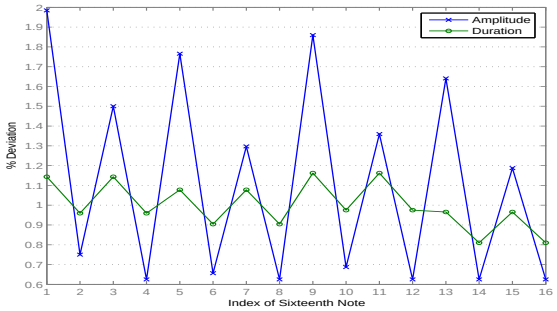


Fig. 3. Excerpt of Structural Analysis of BWV 846

The group boundaries are represented by $\overset{\times}{\times}$and the numbers under the starves give the accentuation information.

From a run lasting for 200 generations, the best pulse set obtained is depicted in Figure 4, with the total fitness value of 2.950 (the maximum this can value is 3). The sum of notes' length specified by this pulse set is 2×2×4×$\frac{1}{32}$ =1 bar. With

16
2 2 4
1.17 0.434
118 99
1.426 1.042
125 125
1.236 1.066 1.119 0.984
122 115 124 103



a.Representation b. Deviation Pattern

Fig. 4. Best pulse set obtained and the deviation pattern

this configuration we can say that the system has captured the positions of group boundaries. And we can also find similar accentuation pattern in the amplitude curve as the accentuation analysis in Figure 3. Since each note's duration is already a parameter determined by multi factors, and some of them may work against each other, at the moment it's hard to find out from this picture whether the pulse set has well fulfilled the cadence rules we had set.

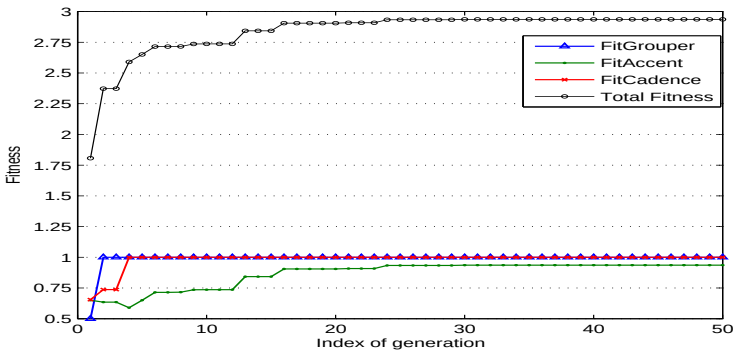


Fig. 5. Fitness

Figure 5 shows the best fitness values for each generation in another run of the system. FitGrouper and FitCadence, which are fitness components as penalties of violation, have been playing the dominated rules in the very beginning of evolution. After they reached the maximum value, the configuration of the best pulse set in following generations almost kept stable before some even better configuration emerged. During this stable period, there can be a number of pulse sets which share the same fitness value, but have different configurations and different duration or amplitude parameters.

As a conclusion, our system demonstrated that it is possible to use a Genetic Algorithm to evolve suitable pulse sets for musical implementation using fitness rules derived from the structure of the piece to be performed. In this way, we can make the computer to choose a performance profile for a piece of music according to the musical structure and the user's preferable interpretation principles. Comparing with directly constructing a performance profile based on the structural rules, the advantage of using Genetic Algorithm is, it is possible to find more than one suitable pulse set for a piece through an efficient way, especially when the structural rules are to be combined in a more complex manner. The possibility of various and suitable pulse sets is necessary preparation for the simulation of interactions in the future work.

4 Future Work

We are currently improving the fitness rules in order to take into account some performance principles associated or determined by melody. Also, having the concern that those performance principles always work on different musical levels and some principles have significant collision against each other, we are considering to modify the straightforward summation scheme for concluding the total fitness and to assign different importance coefficients for different performance principle. Furthermore, as a possible way for keeping diversity of pulse sets, we are also studying ways of varying the number of hierarchical levels, since the pulse sets with more hierarchical levels are supposed to be more compatible with the high complexity of music structure. And after all, we would like to utilize these diversity and similarity among different pulse sets to design the scheme for the goal of this project, which is to evolve performance profiles in form of pulse set by agents' interactions in agent society.

References

1. Miranda, E.R.: At the crossroads of evolutionary computation and music: Self-programming synthesizers, swar, orchestras and the origins of melody. Evolutionary Computation **12**(2) (2004) 137–158
2. Peter M.Todd, G.M.: Music Networks: Paralel Distributed Perception And Performance. (MIT Press)
3. Miranda, E.R.: On the music of emergent behaviour: What can evolutionary computation bring to the musician? Leonardo **36** (2003) 55–58
4. Widmer, G.: Computational models of expressive music performance: The state of the art. Journal Of New Music Research **31** (2002) 37–50
5. Poli, G.D.: Methodologies for expressiveness modelling of and for music performance. Journal Of New Music Research **33** (2004) 189–202
6. Todd, N.: The dynamics of dynamics: A model of musical expression. Journal Of The Acoustical Society Of America **91** (1992) 3540–3550
7. Bresin, R.: Artificial neural networks based models for automatic performance of musical scores. Journal Of New Music Research **27** (1998) 239–270
8. Sundberg, J.: Integrated Human Brain Science: Theory, Method Applicatinon(Music). (Elsevier Science)
9. Widmer, G., T.A.: Playing mozart by analogy: Learning multi-level timing and dynamics strategies. Journal Of New Music Research (2003)

10. Clynes, M.: Microstructural musical linguistics: Composer's pulses are liked best by the best musicians. Cognition, International Journal Of Cognitive Science **55** (1995) 269–310
11. Clarke, E.F.: Generative Processes In Music, The Psychology Of Performance, Improvisation, And Composition. Oxford Science Publications (1988)
12. Clynes, M.: Generative principles of musical thought integration of microstructure with structure. Journal For The Integrated Study Of Artificial Intelligence, Cognitive Science And Applied Epistemology **3** (1986) 185–223
13. Tobias Blickle, L.T.: A comparison of selection schemes used in genetic algorithms. Technical report, Computer Engineering And Communication Networks Lab (Tik), Swiss Federal Institute Of Technology (Eth) Zurich (1995)