Nicolas Monmarché El-Ghazali Talbi
Pierre Collet Marc Schoenauer
Evelyne Lutton (Eds.)

LNCS 4926

Artificial Evolution

8th International Conference, Evolution Artificielle, EA 2007
Tours, France, October 2007
Revised Selected Papers

Springer

Nicolas Monmarché El-Ghazali Talbi
Pierre Collet Marc Schoenauer
Evelyne Lutton (Eds.)

Artificial Evolution

8th International Conference
Evolution Artificielle, EA 2007
Tours, France, October 29-31, 2007
Revised Selected Papers

Springer

Volume Editors

Nicolas Monmarché
Ecole Polytechnique de l'Université de Tours, Laboratoire d'Informatique
64 avenue Jean Portalis, 37200 Tours, France
E-mail: nicolas.monmarche@univ-tours.fr

El-Ghazali Talbi
Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Bât. M3, 59655 Villeneuve d'Ascq, France
E-mail: el-ghazali.talbi@lifl.fr

Pierre Collet
Université Louis Pasteur, Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (LSIIT) Strasbourg
Parc d'innovation, Boulevard Sébastien Brant, BP 10413, 67412 Illkirch, France
E-mail: Pierre.Collet@lsiit.u-strasbg.fr

Marc Schoenauer
INRIA Saclay - Île-de-France, LRI
Université de Paris-Sud
Bât. 490, 91405 Orsay, France
E-mail: marc.schoenauer@inria.fr

Evelyne Lutton
INRIA Saclay - Île-de-France
Parc Orsay Université
4 rue Jacques Monod, 91893 Orsay, France
E-mail: evelyne.lutton@inria.fr

Library of Congress Control Number: 2008925337

CR Subject Classification (1998): F.1, F.2.2, I.2.6, I.5.1, G.1.6, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-540-79304-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-79304-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12257629 06/3180 5 4 3 2 1 0

Preface

This book is based on the best papers presented at the 8th Conference on Artificial Evolution, EA[1] 2007, held in Tours (France). Previous EA meetings took place in Lille (2005), Marseille (2003), Le Creusot (2001), Dunkerque (1999), Nimes (1997), Brest (1995), and Toulouse (1994).

Authors were invited to present original work relevant to artificial evolution, including, but not limited to: evolutionary computation, evolutionary optimization, co-evolution, artificial life, population dynamics, theory, algorithmics and modeling, implementations, application of evolutionary paradigms to the real world (industry, biosciences, ...), other biologically inspired paradigms (swarm, artificial ants, artificial immune systems, ...), memetic algorithms, multi-objective optimization, constraint handling, parallel algorithms, dynamic optimization, machine learning and hybridization with other soft computing techniques.

Papers submitted to the conference were reviewed by at least three members of the International Program Committee, and 30 out of the 62 submissions were selected to be presented at the Conference. As for the previous editions (see, in the same collection, volumes 1063, 1363, 1829, 2310, 2936, and 3871), 27 of those papers were revised according to the reviewers' comments, and are now included in this volume, resulting in a 43.5% acceptance rate for this volume.

We would like to thank the Program Committee for the conscientious work during the paper selection stage of this conference. We are also very grateful to the Organizing Committee for their efficient work and dedication to provide a pleasant environment for conference attendees. We take this opportunity to thank the different partners whose financial and material support contributed to the success of the conference: PolytechTours, University of Tours, DGA, Ministère de l'Éducation Nationale, de l'Enseignement Supérieur et de la Recherche, City of Tours, INRIA, AFIA, Région Centre, ROADEF, and EA association. Last but not least, we thank all the authors who submitted papers, and the authors of accepted papers for revising and sending their final versions on time.

December 2007

Nicolas Monmarché
El-Ghazali Talbi
Pierre Collet
Marc Schoenauer
Evelyne Lutton

[1] As with previous editions of the conference, the EA acronym is based on the original French name "Évolution Artificielle".

Évolution Artificielle 2007 – EA 2007

October 29–31, 2007

Université François Rabelais, Tours, France
8th International Conference on Artificial Evolution

Steering Committee

Organizing Committee

Program Committee

Invited Talks

Adaptive Business Intelligence, Zbigniew Michalewicz
Puzzle Based Learning (Invited Tutorial), Zbigniew Michalewicz

Sponsoring Institutions

École Polytechnique de l'Université de Tours
Université François Rabelais de Tours
Conseil Régional du Centre
Ville de Tours
Institut National de Recherche en Informatique et en Automatique (INRIA)
Délégation Générale pour l'Armement (DGA)
Ministère de l'Éducation Nationale de l'Enseignement Supérieur et de la Recherche
Association Française d'Intelligence Artificielle (AFIA)
Société française de Recherche Opérationnelle et Aide à la Décision (ROADEF)

Table of Contents

Treating Noisy Data Sets with Relaxed Genetic Programming

Luis Da Costa[1,2], Jacques-André Landry[1], and Yan Levasseur[1]

[1] LIVIA, École de Technologie Supérieure, Montréal, Canada
[2] INRIA Futurs, LRI, Univ. Paris-Sud, Paris, France

Abstract. In earlier papers we presented a technique ("**RelaxGP**") for improving the performance of the solutions generated by **G**enetic **P**rogramming (GP) applied to regression and approximation of symbolic functions. RelaxGP changes the definition of a *perfect* solution: in standard symbolic regression, a perfect solution provides exact values for each point in the training set. RelaxGP allows a perfect solution to belong to a certain interval around the desired values.

We applied RelaxGP to regression problems where the input data is noisy. This is indeed the case in several "real-world" problems, where the noise comes, for example, from the imperfection of sensors. We compare the performance of solutions generated by GP and by RelaxGP in the regression of 5 noisy sets. We show that RelaxGP with relaxation values of 10% to 100% of the gaussian noise found in the data can outperform standard GP, both in terms of generalization error reached and in resources required to reach a given test error.

1 Motivation and Background

Darwinian evolution and Mendelian genetics are the inspiration for the field of **G**enetic **P**rogramming (GP); GP solves complex problems by evolving populations of computer programs, and is particularly relevant in approaching non-analytical, multiobjectives and (practically) infinite solution space dimension problems (for details on GP, we invite interested readers to consult [6] or [1]). GP has proved its utility for supporting human analysis of problems in a variety of domains: computational molecular biology [7], cellular automata, sorting networks, analogical electrical circuits [5], among them.

We showed, in [2] and [3], that calculating fitness values with "relaxed" points on a GP problem helps avoiding the overfitting of the best solutions, as well as reducing the use of resources on the course of a GP run. Specifically, we designed a variation of GP where a *perfect* solution is not defined as the solution whose training points are reached with error zero, but as a solution that is *close enough* to them. In practical terms, this means that the fitness calculation is modified: a solution has fitness 0 ("perfect") if its approximation falls within an interval around the real targeted values. This concept is depicted in Fig. 1; the original ("perfect") point is (x_1, y_1). Now, with an absolute relaxation value of δ, any point with y value in $(y_1 - \delta, y_1 + \delta)$ has a perfect (zero) fitness.

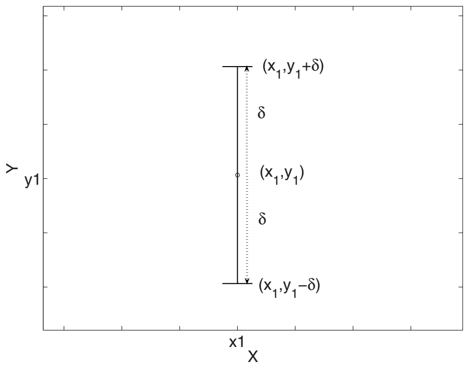
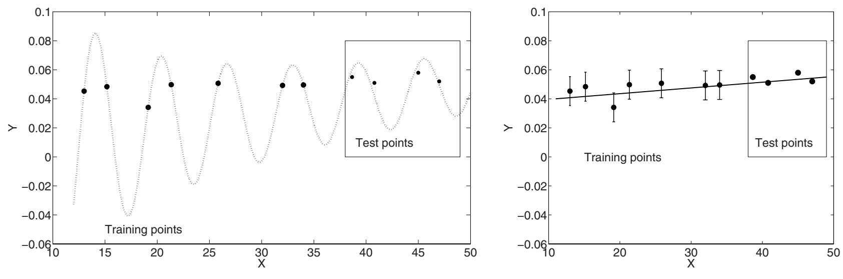


Fig. 1. Relaxation (δ) at one point. For fitness matters, the distance between (x_1, y_1) and (x_1, y_2) is 0 if $y_2 \in [y_1 - \delta, y_1 + \delta]$, $|y_2 - y_1|$ otherwise.

This definition enables a "relaxation" of the data, as the selection of individuals over the genetic process is more permissive. As an example, in Fig. 2, GP was used to evolve an individual well-trained to a certain training set (Fig. 2(a)); that best individual resulted in a sinusoidal function. On the other hand, a population evolved by RelaxGP yielded a straight line as best individual (the relaxation is noticeable by the vertical segments at each training point); a straight line being a simpler mathematical model than a sinusoidal function, the best individual generated by RelaxGP is then simpler than the one generated by GP. Their test errors are similar (right part of Figures 2(a) and (b)).



(a) GP (b) RelaxGP

Fig. 2. Solutions by standard and relaxed GP for a problem. The sinusoidal function (left) was produced by standard GP, and corresponds to $sin(x)/x + x/1000$. The straight line was produced by relaxed GP.

Our basic working hypothesis is that relaxing the data set gives more freedom to the solutions generated by GP, avoiding in this way overfitting to the training set, and guiding the population towards a generalizing solution. At the same time, giving some latitude to the individuals in a population produces more compact solutions than with regular fitness calculation.

In this paper we start from the idea that the proposed RelaxGP technique should be very useful in treating data that is, from the beginning, noisy and uncertain. We feel it is counterproductive to try to reach a perfect fit on points that are not perfect, by themselves. This is the case for most practical applications, where uncertainty in measurements is always present, and is considered part of the problem by the researchers.

To measure the adequacy of the technique for solving this kind of problems, we compared the performance of regular GP with those of RelaxGP (with various relaxations) on the task of predicting the output of a noisy set of points. These points were generated by introducing Gaussian noise to the output of the "quartic polynomial function" ($Q(x) = x^4 + x^3 + x^2 + x$).

In the remaining of the paper, we present our experimental protocol and experiments; our main objective is to find a relation between the quantity of gaussian noise in data and the best relaxation value for RelaxGP. This allows for an appropriate relaxation value to be chosen if gaussian noise can be found and measured in a problem's input data. We also show how the RelaxGP technique can outperform standard GP in terms of both the (computational) resources required for the experiments and the actual test performance when the optimal relaxation value is used.

2 Experimental Setting

2.1 Sets of Noisy Points

Gaussian Noise. A point y, perturbed by an amount of Gaussian noise a, yields a value y'; y' belongs to the interval $[y - a, y + a]$ with probability .95. a is called an *absolute* noise (a can be seen as approximately twice the standard deviation of the data).

A *relative* noise r is defined with respect to the total range δY of values $Q(x)$ can take when $x \in [-10, 10]$. The absolute value corresponding to r is $a = (r * \delta Y)/100$. For $x \in [-10, 10], \delta Y = 1.111 * 10^4$

For this paper, we used 5 relative noise values: 0.5, 1, 2, 5 and 10.

Training Points. We generated several sets of "noisy" points used for training. Each set has 80 points in the region $x \in [-10, 10]$ (one set for each noise value), generated from the mathematical definition of the quartic polynomial ($Q(x) = x^4 + x^3 + x^2 + x$) and introducing the given amount of Gaussian noise. In Fig. 3(a) the 80 points generated for training with noise 10% are shown.

Test Points. 160 noisy points were generated for testing, for each noise value, in the region on interval $x \in [-20, 20]$. In Fig. 3(b) the points generated for testing with noise 10% are shown.

2.2 RelaxGP: Relaxation Values

We applied an amount of relaxation relative to the noise present in the points of the set. In that way, different experiments, with different noises, can be compared

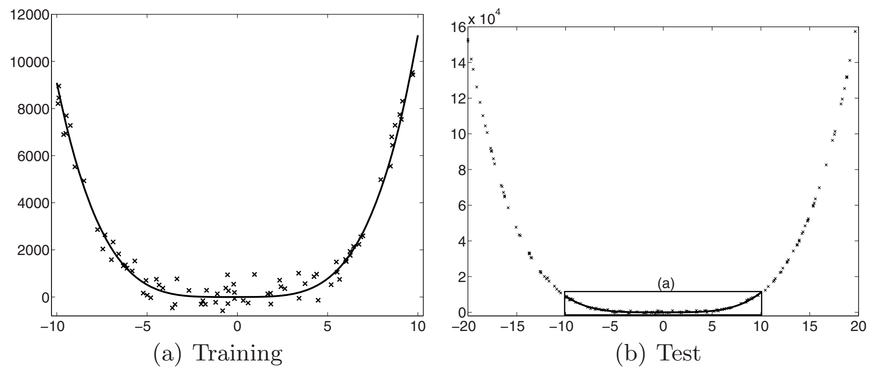


Fig. 3. Values from the quartic function. 80 values were generated for training in the interval $x \in [-10, 10]$ for each noise value. 160 values were generated for testing in the interval $x \in [-20, 20]$ for each noise value. The values generated with noise 10% are shown as crosses. The full line is the plot of $Q(x)$. Sub-plot (a) was zoomed from sub-plot (b).

to each other. For a given (relative) noise the amounts of relaxation applied to the fitness functions were: 0 (regular GP), 10%, 100% (so relaxation = noise), 200% and 500%.

2.3 GP (and RelaxGP) Runs' Parameters

500 runs for each experiment were conducted, with the following parameters for each run (for details on the meaning of each parameter, please refer to [6]):

– Population of 500 individuals.
– Roulette selection reproduction.
– Crossover probability: 95%
– Mutation rate = 10%.
– Elitism defined by selection after reproduction: children are generated from the population of parents, and from this new population (parents and children together) only the best 500 are kept.
– A run stops when either (a) the best solution of the population solves the success predicate or (b) generation 50 is reached.

The experiments were conducted with **GPLAB** [8], a GP toolbox for **MATLAB**, modified in order to allow optimization by intervals.

2.4 Measures

In this problem we're studying resource utilization and generalization power.

Resource utilization. We presented in [3] a complete method for measuring the performance of the genetic programming paradigm in terms of the amount of computer processing necessary to solve a particular problem. This is an extension of the method presented by Koza in [6, chap. 8]. A quick refresher is presented here.

The need for a careful inspection of the results of GP runs arises from the fact that there is randomness in the normal functioning of the GP algorithm: in the creation of the initial population, selection of individuals for reproduction, selection of crossover points, number of genetic operators to be executed and (possibly) the election of the points where fitness is calculated. Because of these probabilistic steps, there is no guarantee that a given run will yield an individual that satisfies the success predicate of the problem after being run for a number of generations [6, page 191].

One way to minimize this effect is to do multiple independent runs; the amount of computational resources required by GP[1] is then determined by (a) the number of independent runs needed to yield a success with a certain probability p and (b) the resources used for each run. The total number of resources is calculated as the sum of all resources used on each of the runs.

The method presented in [3] and [6, chap. 8] aims at estimating, in a robust statistical way, the amount of processing needed to reach a certain success predicate with a certain probability. For that we perform an important number of replications (500, in [3]) and we calculate (1) the practical probability of reaching the predicate at a certain generation, and (2) the amount of resources needed to reach a certain generation. Combining these two pieces of information yields the best couple (k, g), indicating the need to replicate the experiment k times up to generation g to reach the objective.

The result of such an analysis produces a curve and a table of values (see Fig. 4 and the accompanying Table). We can observe on Fig. 4 that the number of nodes to be evaluated is minimized when running 5 replications up to generation 40. The objective would then be reached at the evaluation of $1.63 * 10^6$ nodes.

Generalization Power. A central problem in the field of *statistical pattern recognition* is how to design a classifier that could, based on a set of examples (the *training set*), suggest actions when presented with *novel* patterns[4]. This is the issue of *generalization*.

Since good generalization is our objective, our data will be split in two parts: one is used as the traditional training set for designing the approximating function. The other (the "test" set) is used to estimate the generalization error. As mentioned in Subsec. 2.1, the training set for each experiment has 80 points on the interval $[-10, 10]$, the test set has 160 points on the interval $[-20, 20]$.

3 Results

For each noise we performed 6 groups of experiments, each with a relaxation relative to the noise. Each group corresponds to a relaxation of 10%, 50%, 100%,

[1] Or by the conventional genetic algorithm.

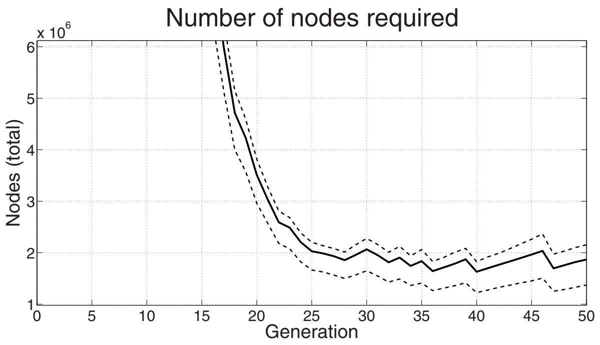


Fig. 4. Total resources (nodes) to evaluate to reach a given objective. The solid line is the average of the value, the dotted lines correspond to average ± standard deviation. The values are presented in the following table:

Generation	Runs	Number of required nodes (10^6)	Generation	Runs	Number of required nodes (10^6)
20	31	3.51	40	5	1.63
36	6	1.64	46	5	1.70
39	6	1.87	47	4	1.76

200% and 500%. For example, for noise 0.5, the points were relaxed a relative amount of 0.05, 0.25, 0.5, 1, and 2.5

For each experiment we measured, at each generation g:

1. **The test error for the best individual at** g: the test error for an approximation f to Q is calculated on the (160) test points as:

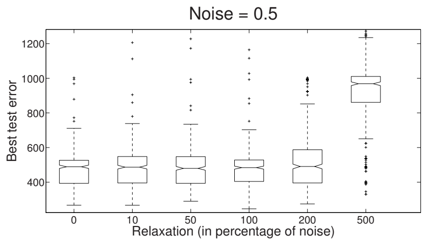
$$t_a = \sum_{i=1}^{160} |f(x_t^{(i)}) - y_t^{(i)}| \tag{1}$$

where $(x_t^{(i)}, y_t^{(i)})$ is the i-th test point.
2. **The number of nodes evaluated to reach** g

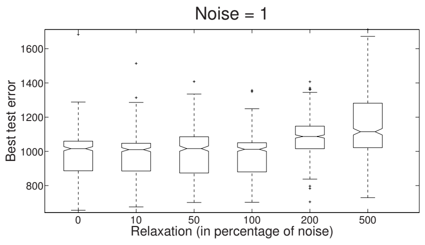
The first results are presented as boxplots showing the values obtained for the test error of the best individual after 50 generations. Results for noise 0.5 are presented in Fig. 5(a), for noise 1 in Fig. 5(b), for noise 2 in Fig. 6(a), for noise 5 in Fig. 6(b) and for noise 10 in Fig. 7.

We observe that, for all noise values, the best results are obtained when the value of the relaxation is less than the noise itself (relaxation $\leq$ 100 on all boxplots). It is interesting to notice that standard GP (relaxation 0 on boxplots) does not (statistically) outperform RelaxGP with relaxation $\leq$ 100.

To obtain more precise information we show the results of the study concerning resource utilization coupled with performance values (see 2.4, page 4 for details).

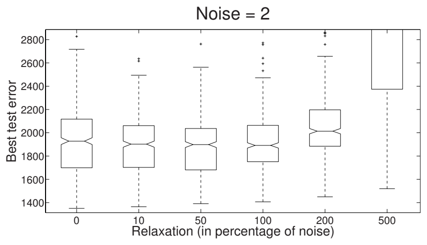


(a) Noise 0.5%

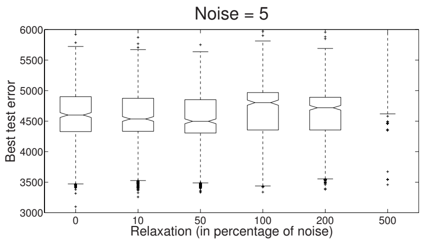


(b) Noise 1%

Fig. 5. Noise 0.5% and 1%



(a) Noise 2%



(b) Noise 5%

Fig. 6. Noise 2% and 5%

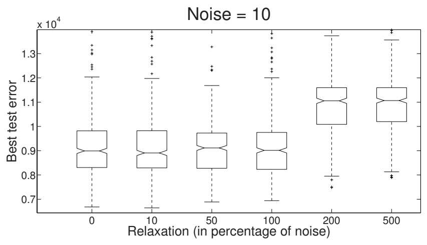


Fig. 7. Noise 10%

Comparing target errors with resources. We are interested in measuring the number of resources required to reach a certain test error t, for a certain relaxation r. For this we define a success criteria $\mathcal{S}$:

$$\mathcal{S} : \text{“}\textit{to reach test error } t\text{”}$$

We choose to look for the values that reach $\mathcal{S}$ with probability .99. The analysis of Subsec. 2.4 is then undertaken, and then the minimum value of nodes, n, reaching $\mathcal{S}$, is kept. The pair (t, n) is then interpreted as:

> *With probability* .99 *(based on 500 runs), RelaxGP with relaxation* r *needs* n *nodes to reach error* t*.*

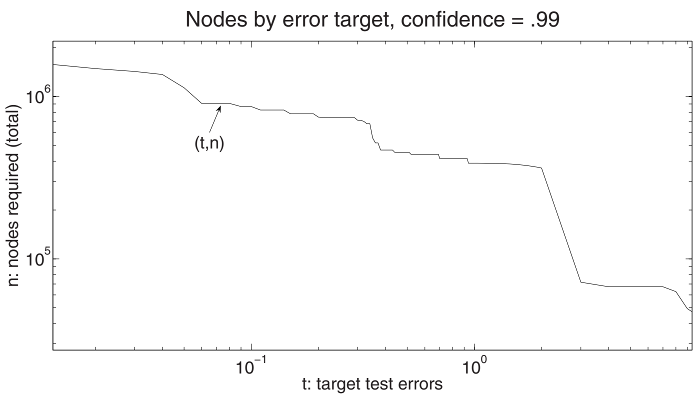


Fig. 8. Typical extended curve for a given relaxation r

Repeating this procedure for a set of test errors t yields a curve for relaxation r (Fig. 8). A point (t, n) is read as "*replications of experiment with relaxation r having reached a test error of* **at most** *t have evaluated n nodes*"; two points resulting from such calculations have been linked by a straight line.

For comparing the effects of two relaxation values r_1 and r_2 we plot the two curves together (Fig. 9). Certain comparisons can then be made; for example

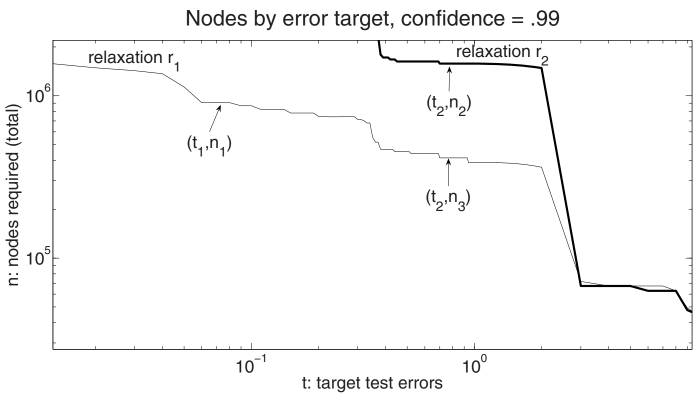


Fig. 9. Extended curve for relaxations r_1 and r_2

the curve for relaxation r_1 shows a set of points whose value can not be reached by the curve of relaxation r_2. Test error t_1 can not be reached with relaxation r_2, while with relaxation r_1 error t_1 is reached using, in average, n_1 nodes.

The relative position of the curves is also important; for example, we observe that for test error t_2 relaxation r_1 uses n_3 nodes, while relaxation r_2 uses n_2 nodes, and $n_2 > n_3$. So, if we are interested in using *less resources* for reaching error t_2, we should relax the training data by r_1 (instead of by r_2).

This idea is now being applied to the comparison of all the relaxations together. In order to compare different experiments, with different values, an absolute error t_a (defined in (1), page 6) is expressed as a percentage of (a) the number of test points and (b) the total range δY of values $Q(x)$ (the quartic polynomial) when $x \in [-20, 20]$ (interval from where the test points were generated). So, t_r, the relative test error corresponding to t_a, is:

$$t_r = \frac{t_a}{160 * \delta Y} * 100 \tag{2}$$

The analysis of the results yields several interesting points. First, for all sets of noisy values there are relaxations that outperform standard GP (***i.e.***, non-zero relaxations resulted in a best relative test error than relaxation 0). The best values for each noise are depicted in Table 1 (see Fig. 10 and Fig. 11 for details). An interesting fact from this Table is that relaxations 10% and 100% are always the best performers.

We also noticed that non-zero relaxations are generally better (in terms of resources required to reach a certain error) than standard GP in all experiments (Table 2), the only exception being a range around relative error 3.25 in noise 0.5% (Fig. 10(a)). In other words, relaxing the fitness function results in a better utilization of resources for reaching a given test error. The relative gain is the smallest in the case of the smaller noise (0.5, a gain of 12.57%), and comparable in the other cases (in the range [22 − 28]%).

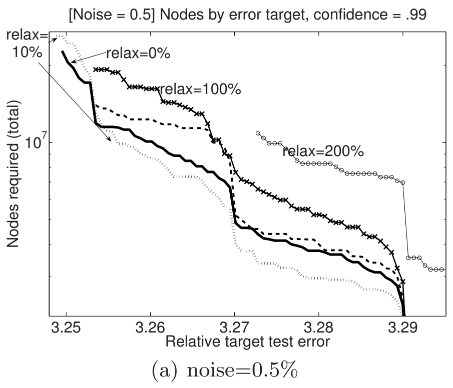
These and other results of our analysis are now presented in terms of resources required to reach a set of target **relative** errors: noises 0.5 and 1 are presented

Table 1. Best relative error reached

Noise	Standard GP	RelaxGP	
	Test Error	Best relaxation	Test Error
0.5	3.258	10%	3.257
1	6.731	10%, 100%	6.724
2	11.14	100%	11.11
5	33.70	10%	33.62
10	52.12	100%	51.76

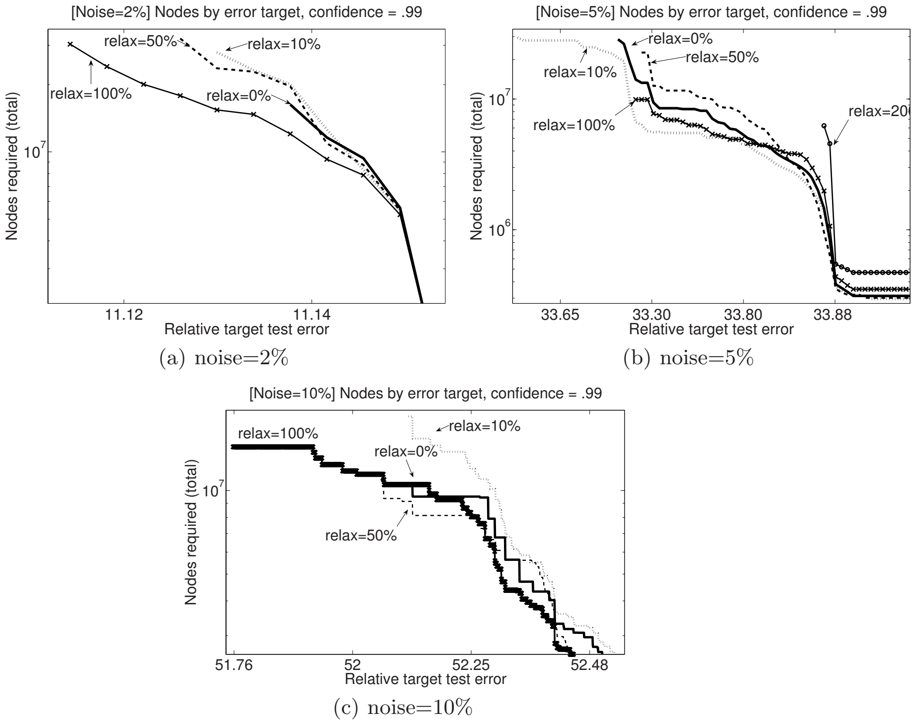
Table 2. Utilization of resources

Noise	Relative error	RelaxGP		Standard GP.	Gain
	for measure	Relaxation	Resources	Resources	(%)
0.5	3.26	10%	$8.83 * 10^6$	$1.01 * 10^7$	12.57%
1	6.73	10%	$1.57 * 10^7$	$2.02 * 10^7$	22.17%
2	11.14	100%	$1.20 * 10^7$	$1.63 * 10^7$	26.38%
5	33.70	10%	$2.06 * 10^7$	$2.86 * 10^7$	27.97%
10	52.12	50%	$8.13 * 10^6$	$1.05 * 10^7$	22.57%



(a) noise=0.5% (b) noise=1%

Fig. 10. Noises 0.5% and 1%



(a) noise=2% (b) noise=5%

(c) noise=10%

Fig. 11. Noises 2%, 5%, and 10%

in Fig. 10. Noise 2, 5 and 10 are in Fig. 11. In all plots, the thick black line corresponds to the standard GP. For being considered into the plot, a target error of value t must have been reached by at least 30 of the (500) replications.

An observation made from the plots reaffirms the results presented earlier in the boxplots (Figs. 5, 6 and 7): only relaxations equal or less than the noise are competitive with standard GP: curves of relaxations 200% and 500% are worse than relaxation 0 both in best error reached and in terms of resources used.

4 Discussion

In this paper we proposed the use of a technique we developed earlier (RelaxGP, in [2] and [3]) as an alternative to treat noisy data sets in the context of regression and approximation of symbolic functions. RelaxGP stands on a new definition of a *perfect* solution: in standard symbolic regression, a perfect solution provides exact values for each point in the training set. RelaxGP allows a perfect solution to belong to a certain interval around the desired values.

Our main hypothesis was that RelaxGP should outperform classical GP in the solving of regression problems where the input data is originally noisy. Noisy data is actually found in several "real-world" problems, where the noise comes, for example, from the imperfection of sensors. We compare the performance of solutions generated by GP and by RelaxGP in the regression of 5 noisy sets. The performance was assessed through the measure of the solutions' *generalization error* (cumulative error on a set of values not used for training) and the amount of resources (in number of nodes) used for attaining a certain performance.

For all our experiments, RelaxGP, using an appropriate relaxation value, outperformed standard GP, both in terms of generalization error reached and in number of resources required to reach a certain given test error. Moreover, the amount of relaxation "optimal" for each noise n is experimentally discovered to be between 10 and 100% of the noise of the data (always assuming a Gaussian noise). In other words, if we can have an idea of "how large" the noise of the measures making up the input to a problem is, the best way to attack that problem is by using RelaxGP with relaxation lower than this noise.

These results are positive and motivate the need for another set of experiments, where we could understand more precisely the exact nature of the influence of relaxation on the dynamics of the evolutionary process, in general, and on the "cleaning" of noisy data, in particular.

Acknowledgements

The authors would like to acknowledge the financial support of the the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming An Introduction. Morgan Kauffman Publishers, San Francisco (1998)
2. Da Costa, L.E., Landry, J.-A.: Relaxed genetic programming. In: Keijzer, M. (ed.) Proceedings of GECCO 2006, Seattle, USA, pp. 937–938 (2006)

3. Da Costa, L.E., Landry, J.-A., Levasseur, Y.: Improving genetic programming by relaxing the fitness function. Improving genetic programming by relaxing the fitness function (2007) (Submitted to Genetic Programming and Evolvable Machines)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley and sons, Inc., New York (2001)
5. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.A.: Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 132–140. The MIT Press, Stanford University. Cambridge, MA (1996)
6. Koza, J.R.: Genetic Programming-On the programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge, London (1992)
7. Koza, J.R.: Evolution of a computer program for classifying protein segments as transmembrane domains using genetic programming. In: Altman, R., Brutlag, D., Karp, P., Lathrop, R., Searls, D. (eds.) Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pp. 244–252. AAAI Press, Menlo Park, CA (1994)
8. Silva, S.: Gplab-a genetic programming toolbox for matlab (2004), http://gplab.sourceforge.net

Cost-Benefit Investigation of a Genetic-Programming Hyperheuristic

Robert E. Keller and Riccardo Poli

Department of Computing and Electronic Systems, University of Essex, UK

Abstract. In previous work, we have introduced an effective, grammar-based, linear Genetic-Programming *hyperheuristic*, i.e., a search heuristic on the space of heuristics. Here we further investigate this approach in the context of search performance and resource utilisation. For the chosen realistic travelling salesperson problems it shows that the hyperheuristic routinely produces metaheuristics that find tours whose lengths are highly competitive with the best results from literature, while population size, genotype size, and run time can be kept very moderate.

1 Introduction

A heuristic is a method that, given a problem, often finds a good solution within acceptable time, while it cannot be shown that a found solution cannot be bad, or that the heuristic will always operate reasonably quickly. A metaheuristic is a heuristic that approaches a problem by employing heuristics. The term *hyperheuristic* [21], see [22] for its origin, refers to a heuristic that explores the space of metaheuristics that approach a given problem.

Over the past few years, hyperheuristics (HH) have increasingly attracted research interest. For example, [7] suggests a method of building low-level heuristics for personnel scheduling, [6] proposes tabu search on the space of heuristics, [9] describes a timetabling application of a hyperheuristic, and [8] suggests simulated annealing as learning strategy for a hyperheuristic. [19] employs Genetic Programming (GP) [2, 15, 16] for evolving Evolutionary Algorithms that are applied to problems of discrete optimisation. For the bin-packing problem, [4] introduces a hyperheuristic that is driven by GP. This system successfully reproduces a human-designed bin-packing method.

While the approaches presented in these papers use fixed, problem-specific languages implying sequential execution of actions, our linear GP hyperheuristic, introduced in [13] and further investigated in [14], uses grammars to obtain independence from a given problem domain and to contribute to guiding the search for a solution to a given problem.

In our previous work we saw that the introduction of a looping construct in one of the investigated grammars proved crucial to the effectiveness of the hyperheuristic: it routinely produced metaheuristics that actually delivered best-known solutions to larger TSP benchmark instances despite the simplicity of the underlying grammar. Also the low-level heuristics, given to the hyperheuristic as building material, were basic, showing that a user is only required to provide simple heuristics.

The advantage of the approach is that domain knowledge becomes a free resource for the GP hyperheuristic that does not have to rediscover the provided component heuristics. Moreover, by crafting a grammar appropriately, one can direct evolutionary search towards promising types of metaheuristics.

The demonstrated principle and its real-world effectiveness clearly confirmed the original hope behind hyperheuristics that they can lead to optimisation methods that are more flexible in their application to different practical domains. In this context, the domain-independence of the principle is of particular relevance since a fixed HH that efficiently operates for all domains cannot be designed [24]. This obstacle can be circumvented with our GP hyperheuristic because a decision maker can specialise it for a given problem domain by changing the supporting grammar.

As seen in our previous work, the demands on the user of the hyperheuristic are very modest in terms of sophistication of heuristics to be supplied to the HH. In the present paper, we shall investigate the question whether the HH is also easy on its computing resources, in particular in terms of the sizes of populations and genotypes, and whether obtaining a significant increase in search performance does only require a modest additional investment of resources. To the end of experimenting, we use those grammars from our previous work that have shown most beneficial in guiding the hyperheuristic search.

The paper is organised as follows. In Section 2, we introduce the hyperheuristic in detail. In Section 3, we describe the types of problem used in experiments with the hyperheuristic. In Section 4, we describe the grammars that we then use for experiments described in Section 5. In Section 6, we give a summary and conclusions, while in Section 7 we describe interesting avenues for future work.

2 A Linear-GP Hyperheuristic

Our GP hyperheuristic accepts the definition of the structure of desired metaheuristics for D, an arbitrary, fixed domain of problems. Then, in principle, after changing this description appropriately, one can apply the HH to a different domain.

To give the definition, one may represent some of D's low-level heuristics or well-known metaheuristics as components of sentences of a language that one describes by a grammar G. In this manner, $\sigma \in L(G)$ defines a metaheuristic for D. Then, any form of grammar-based GP (e.g., [20] [23] [18] [12] [25] [10]), evolving programs from $L(G)$, is a hyperheuristic for D. Here, we describe our HH implementation that is a flavour of linear GP [3].

A metaheuristic is represented as a genotype $g \in L(G)$ with a domain-specific grammar G. T shall designate the set of terminals of G. $L(G) \subset \mathbf{T}^*$, the set of all strings over T. We call a terminal $t \in T$ a *primitive* (and T a *primitive set*) to avoid confusion regarding “terminal” as used in the field of GP. Primitives

Algorithm 1. GP-based HYPERHEURISTIC

```
 1: given: grammar $G$, population size $\mathsf{p}$, length $\mathsf{l}$
 2: **repeat**
 3:    produce next random primitive-sequence $\sigma : |\sigma| = \mathsf{l}$
 4:    EDITING($\sigma$,$G$) $\rightarrow g$ genotype
 5: **until** $\mathsf{p}$ genotypes created
 6: **while** time available **do**
 7:    *Selection*: 2-tournament
 8:    *Reproduction*: Copy winner $g$ into loser's place $\rightarrow g'$
 9:    *Exploration*: with a given probability
          Mutate copy $g' \rightarrow \delta$
          EDITING($\delta$,$G$)$\rightarrow g''$ genotype
10: **end while**
```

may represent manually created metaheuristics, low-level heuristics, or parts of them.

The execution of a metaheuristic, g, with $g = \mathsf{i}_0\, i_1 ... i_n$, $i_j \in T$, means the execution of the i_j. This execution constructs a complete structure, s, that is a candidate solution to the given problem. More specifically, s is obtained from an initial, complete structure, $\mathsf{i}_0()$:

$$s = i_n(...(i_1(\ \mathbf{i_0}()\ ))...).$$

All i_j with $j \neq 0$ accept a complete structure as input. All i_j deliver a complete structure as output. In particular, i_0, in some straightforward fashion, delivers an initial, complete structure.

g's fitness shall depend on the quality of s because the execution of g's primitives builds s in the described manner.

At the beginning of a run of the GP hyperheuristic (s. Algorithm 1), given population size p, *initialisation* produces p random primitive-sequences from T^*. All such sequences are of the same length, l. *Mutation* of a genotype $g \in L(G)$ randomly selects a locus, j, of g, and replaces the primitive at that locus, i_j, with a random primitive, $t \in T, t \neq i_j$.

Naturally, both initialisation and mutation may result in a primitive-sequence, $\sigma = \mathsf{i}_0..i_j..i_k \in T^*$, that is not a valid genotype, i.e., $\sigma \notin L(G) \subset T^*$. In this case, the sequence is passed to an operator called *EDITING* that starts reading σ from left to right.

If EDITING reads a primitive, p, that represents a syntax error in its current locus, EDITING replaces it with the *no-operation primitive*, **n**. These steps are repeated until the last primitive has been processed. Then, either the current σ is in $L(G)$, and EDITING ends, or still $\sigma \notin L(G)$. In the latter case, EDITING keeps repeating the above steps on σ, but this time processing it from right to left. The result is either a $\sigma \in L(G)$ or a σ that consists of n-instances only. In this latter, unlikely case, EDITING then assigns the lowest available fitness value to σ. This way σ will most likely disappear from the population during *tournament selection*.

Note that, although we initialise the population using sequences of a fixed length, l, the application of EDITING effectively leads to a population containing genotypes of variable lengths not longer than l. This variation in genotype size is beneficial, as, in principle, it allows the evolution of parsimonious heuristics.

We actually observed this effect and described it in [14]. It may contribute to saving run time since a shorter genotype may execute faster. In any case, l, the maximally available genotype size, controls the actual genotype sizes, and we shall investigate its influence on search performance later.

3 Problem Domain

To study aspects of resource use in the context of the performance of the GP hyperheuristic, we select the NP-hard set of travelling salesperson problems (TSP) [17].

In its simplest form, a TSP involves finding a minimum-cost *Hamiltonian cycle*, also known as "*tour*", in a given, complete, weighted graph. Let the n nodes of such a graph be numbered from 0 to $n-1$. Then, one describes a tour involving edges (v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_0) as a permutation $p = (v_0, ..., v_{n-1})$ over $\{0, ..., n-1\}$.

We call permutation $(0, 1, ..., n-1)$ the *natural* cycle of the graph. The weight of an edge (i, j) represents the cost of travelling between i and j. Here, we shall interpret this cost as the distance between i and j. Thus, the shorter a tour is, the higher is its quality.

4 Grammars

We describe TSP-specific languages that will support experimenting. To that end, we require a few simple routines, including basic heuristics, that are represented as primitives of terminal sets of the describing grammars.

The primitive `NATURAL` designates the method that creates the natural cycle for a problem.

The low-level heuristic `2-CHANGE` identifies a minimal change of a tour H into a different tour: given two of H's edges $(a, b), (c, d) : a \neq d, b \neq c$, `2-CHANGE` replaces them with $(a, c), (b, d)$. When the hyperheuristic is about to call a `2-CHANGE` primitive, it randomly selects two appropriate edges, $(a, b), (c, d)$, as arguments for `2-CHANGE`.

Another primitive, `IF_2-CHANGE`, executes `2-CHANGE` only if this will shorten the tour under construction. As every greedy operator, `IF_2-CHANGE` is a boon and a curse, but its introduction is safe here since there is a randomising counterweight in the form of `2-CHANGE`.

Another low-level heuristic is known as a *3-change*: delete three mutually disjoint edges from a given tour, and reconnect the obtained three paths so that a different tour results. Given this method, we define the heuristic `IF_3-CHANGE`:

```
randomly select edges as arguments for 3-change;
if 3-change betters the cycle for the arguments, execute 3-change.
```

```
metaheuristic       ::= NATURAL
                      | NATURAL search

search              ::= heuristic
                      | heuristic search

heuristic           ::= 2-CHANGE
                      | IF_2-CHANGE
                      | IF_3-CHANGE
                      | IF_NO_IMPROVEMENT
```

Fig. 1. Grammar ThreeChange

```
Preamble

heuristic           ::= 2-CHANGE
                      | IF_2-CHANGE
                      | IF_3-CHANGE
```

Fig. 2. Grammar NoNoImprove

Since `IF_3-CHANGE` introduces a further greedy bias if it is used in combination with `IF_2-CHANGE`, it may or may not be helpful to provide a counter-bias, for instance by occasionally allowing for possibly worsening a tour, such as in the heuristic

`IF_NO_IMPROVEMENT`:
`if none of the latest` $1,000$ `individuals produced has found a better best-so-far tour, execute a 2-change`.

Using the defined primitives, we give grammar *ThreeChange* (s. Figure 1). In further grammars, we shall represent the top two grammar rules, i.e., `metaheuristic` and `search`, by the symbol **Preamble**.

A small language is desirable, as it means a small solution space for the hyperheuristic. To understand, by means of a coming experiment, whether `IF_NO_IMPROVEMENT` does or does not improve the effectiveness of the GP hyperheuristic, we remove it from grammar *ThreeChange*. We call the resulting grammar and its language *NoNoImprove* (s. Figure 2).

So far, only sequential and conditional execution of user-provided heuristics are available to evolved metaheuristics. A loop element is required to complete the set of essential control structures. To that end, we introduce the primitive `REPEAT_UNTIL_IMPROVEMENT` p:
`execute primitive` p `until it has lead to a shorter tour or until it has been executed` ι `times for user-given` ι.

An example for the use of `REPEAT_UNTIL_IMPROVEMENT` in a grammar, *DoTill-Improve*, is shown in Figure 3.

```
Preamble

heuristic ::= 2-CHANGE
            | loop IF_2-CHANGE
            | loop IF_3-CHANGE

loop      ::= REPEAT_UNTIL_IMPROVEMENT
            | /* empty */
```

Fig. 3. Grammar DoTillImprove

5 Experiments

5.1 Setup

We number the loci of genotypes, beginning with zero. For the present setup of the hyperheuristic, its random choice of an element from a set shall be uniform. For all experiments, mutation probability (cf. Algorithm 1, step 9) shall be 0.5. The GP-HH shall measure time in terms of the number of offspring individuals produced after creation of the initial individuals (cf. Algorithm 1, step 6).

5.2 First Problem

We consider problem `eil51` from [1]. Its dimension is $n = 51$ nodes, its best known solution has a length of 428.87 [11] with natural length of approximately 1,313.47. For a symmetrical TSP instance, the number of tours that are different in human terms equals $(n-1)!/2$.

The evolved metaheuristics operate on permutations of n nodes, so that the size of their search space is $n!$. $n = 51$ gives about 1.6×10^{66} search points and 1.52×10^{64} different tours. Table 1 reports a subset of results from [14] obtained with grammars *ThreeChange* and *NoNoImprove*, the latter lacking the primitive `IF_NO_IMPROVEMENT`.

Here, we comment on an effect that was not in the focus of our previous work: surprisingly, eliminating the non-destructive primitive `IF_NO_IMPROVEMENT` from grammar *ThreeChange* yields better search performance (s. bottom row of table).

A possible explanation of this phenomenon is the much smaller size of the resulting language which is the solution space of the hyperheuristic. This smaller size may at least partially compensate for the loss of `IF_NO_IMPROVEMENT`. For each of both grammars, given its primitive set T and genotype length l, the size of the induced solution space equals

$$|L(G)| = \sum_{i=0}^{l-1} (|T|-1)^i, \tag{1}$$

since the grammar generates language

Table 1. Performance for eil51 over *ThreeChange* and *NoNoImprove*, 30 runs. Basic parameters: pop.size 100, genotype size 500, offspring 1×10^5, mut. prob. 0.5. "Best" mentions the best value over all runs.

eil51	**Mean best**	SD	Best
Natural length	1,313.47	n.a.	n.a.
ThreeChange	874.96	26.55	810.73
NoNoImprove	798.32	15.98	763.30

$$L(G) = \bigcup_{i=0}^{l-1} \{\mathsf{NATURAL}\} \otimes (T \setminus \{\mathsf{NATURAL}\})^{\otimes i} ,$$

where $\otimes$ is the Cartesian product and the superscript $^{\otimes i}$ represents the Cartesian product iterated i times. The largest term of the finite geometric series (1) is $(|T|-1)^{l-1}$. So, for grammar $G = \mathit{ThreeChange}$, where $|T| = 5$, and for genotype length $l = 500$, this term equals $4^{499} \approx 2.7 \times 10^{300}$. For $G = \mathit{NoNoImprove}$, we obtain merely $3^{499} \approx 1.2 \times 10^{238}$.

Therefore, in terms of enhancing effectiveness and efficiency of the GP hyperheuristic, it may well be recommendable in general to approach a problem first with a small primitive set for the underlying grammar. This is because every language increases in size, often exponentially, when one adds an element to its primitive set. Should the problem at hand resist solution, one can still incrementally add beneficial primitives.

5.3 Second Problem

While the metaheuristics' search space for eil51 already has a realistic size, next, we consider eil76 [1], a 76-node problem with a size of about 1.9×10^{111} search points.

We use the same basic parameters as given in Table 1, and, from here, grammar *DoTillImprove* with parameter ι for the loop primitive. The best known result from literature [11] is $\alpha = 544.37$, obtained by a highly specialised, manually designed hybrid Genetic Algorithm.

An individual of the GP hyperheuristic that locates a tour whose length is at least as good as α shall be called a *top metaheuristic.*

Table 2 shows results regarding our GP hyperheuristic. The mean best over all runs is well within one percent of α. Our evolved top metaheuristics yield tour lengths that are actually shorter than $\alpha = 544.37$. Unfortunately, [11] does not specify whether α is a rounded value. Thus, we report that our GP hyperheuristic has found an overall best tour length of $\alpha_{HH} = 544.36908$.

In any case, since the hyperheuristic at least finds α, the used parameters are a good starting point for further experiments.

Population size. We ask how the performance of the GP hyperheuristic depends on the population size. Therefore, we vary the population size over several orders of magnitude for different experiments.

Table 2. Performance of metaheuristics evolved over language DTI, on problem eil76. 100 runs of GP hyperheuristic. Basic parameters: pop.size 100, genotype size 500, offspring 1×10^6; mut. prob. 0.5. Evolved top metaheuristics at least match effectiveness of hand-crafted Hybrid GA. **P.%**: Mean best or natural length in terms of % of best known result α. All real values rounded off to nearest hundredth.

eil76	<u>Mean best</u>	S.D.	Best	ι	<u>P</u>.%
Nat. length	1,974.71	n.a.	n.a.	n.a.	262.75
DTI	548.99	1.67	**544.37**	2×10^3	0.85
Hybrid GA	n.a.	n.a.	***544.37*** *best known*		n.a.

Table 3. Performance of metaheuristics evolved over language DTI, on problem eil76. 100 runs of GP hyperheuristic for each given population size. Other basic parameters: genotype size 500, offspring 1×10^6, ι 2,000; mut. prob. 0.5. Bottom row gives best known rounded result as found by GP hyperheuristic and Hybrid GA. Over all runs, column *First* gives the mean of the serial number of the first metaheuristic of a run that finds a shortest tour of the run, rounded off to the nearest 1,000, given in the unit of 1,000 individuals.

eil76	<u>Mean best</u>	S.D.	Best	**Pop.size**	<u>P</u>.%	First[k]
Nat. length	1,974.71	n.a.	n.a.	n.a.	262.75	n.a.
	677.53	13.76	614.55	10	24.46	**476**
	548.99	**1.67**	**544.37**	*100*	0.85	627
DTI	**547.48**	1.69	**544.37**	500	**0.57**	845
	552.86	2.55	546.74	1,000	1.56	905
	722.89	29.03	651.82	10,000	32.79	955
GPHH/HGA	n.a.	n.a.	***544.37*** *best known*		n.a.	n.a.

Table 3 shows the results. Starting at p=10,000, smaller population sizes yield better results, up to a point: the drop to p=10 clearly worsens the effectiveness of the GP hyperheuristic. This can be explained by the enhancement of tournament-selection pressure that comes with a smaller population size, which prematurely stalls progress when the pressure becomes too high too early during a run.

We are interested in the *efficiency* of a run of the GP hyperheuristic in terms of the number of individuals it produces before it locates the first of its best individuals.

Table 3 gives these values in its "First" column. While p=10 yields the highest efficiency, the resulting effectiveness (column "Best") is poor. However, p=100 gives best effectiveness (column "Best"), best reliability ("S.D."), second best overall effectiveness ("Mean best"), and second best efficiency ("First"). Thus, an investment in a larger population size is of secondary interest only, as it, at best, yields a marginal improvement in the overall effectiveness, without yielding a better metaheuristic.

Genotype size. Next, we ask for the connection of efficiency and genotype size in the context of effectiveness. To that end, we fix p=100 as it has given best effectiveness, and we shall vary the genotype size, l. For the chosen p-value, Table 3 suggests setting the number of offspring to be produced to 627,000.

Table 4. Runs of GP hyperheuristic for given genotype sizes. Other parameters: population size 100, offspring 627,000; ι 2,000; mut. prob. 0.5. Over all runs done for a given genotype size l, column *Firstbest* gives the mean of the serial number of the top metaheuristic discovered first, if any, else "—"; unit: 1,000 individuals. Column *Runs* gives the number of runs performed for given l. Other details as given in caption of Table 3.

eil76	Mean best	S.D.	Best	**Geno.size**	P.%	Firstbest[k]	Runs
Nat. length	1,974.71	n.a.	n.a.	n.a.	262.75	n.a.	n.a.
	1,314.000	17.55	1,263.400	50	141.38	—	100
	713.601	14.89	677.980	250	31.09	—	100
	642.870	12.4	613.129	300	18.11	—	100
	568.134	4.57	556.451	400	4.37	—	100
DTI	556.043	3.46	545.667	450	2.14	—	55
	548.990	1.67	**544.369**	500	0.85	621	100
	546.531	1.1	**544.369**	600	0.4	494	16
	544.765	0.79	**544.369**	700	0.07	463	10
	544.369	**0.0**	**544.369**	900	0.**0**	**255**	15
GPHH/HGA	n.a.	n.a.	***544.370*** *best known*		n.a.	n.a.	n.a.

Table 4 collects the results. Already for modest values of l, such as 400, the GP hyperheuristic produces competitive metaheuristics. Higher values, still in the same order of magnitude, increase all aspects of performance, such as efficiency ("First best") and effectiveness. l=900 even guarantees the best known result for every observed run. Thus, clearly, if one considers making an additional investment of memory, it should be spent on the genotype size.

Iteration number. Finally, we are interested in the question whether, for p=100, l=500, 1×10^6 offspring, and mutation probability 0.5 (cf. Table 3), at a reasonable expense of more run time per metaheuristic, the hyperheuristic can clearly improve its overall effectiveness. For $\iota = 2,000$, on average, the hyperheuristic produces 97 metaheuristics per second. To approach our question, we run the HH for $\iota = 15,000$.

We find that the mean best's P.% value drops to 0.26, less than a third of its previous value, 0.85. On average, the GP hyperheuristic still produces 56 metaheuristics per second. Thus, the run-time increase by factor $1/56/(1/97) \approx 1.7$ has more than tripled the overall effectiveness of the hyperheuristic. Also, while we consider only 30 runs, a very low standard deviation (0.97) indicates reliable search behaviour.

6 Summary and Conclusions

We have investigated our domain-independent, linear GP hyperheuristic (HH) [14, 13] with respect to its demand for computing resources in the context of its effectiveness and efficiency. The HH produces metaheuristics from a user-given language, employing provided heuristics. We experimented on this approach, using the domain of travelling-salesperson problems. To this end we provided the hyperheuristic with elementary heuristics for this domain and with a progression of simple grammars.

On the used, realistic benchmark problems, it shows that the GP hyperheuristic shows excellent competitiveness, yielding best known tour lengths usually only produced by specialised, sophisticated, man-made solvers initialised with selected tours as good starting points.

We observed that one can increase efficiency and effectiveness of the hyperheuristic by making only modest additional investments of population size, genotype size, and production time of an evolved metaheuristic. Favourable scalability may well be a common property of GP-based hyperheuristics over different problem domains, since [5] reports related results for a different challenge.

Also, regarding our GP hyperheuristic, we noted that decreasing the size of the primitive set of the underlying grammar may help solve a problem. We argued that this is at least due to the resulting, exponentially smaller solution space facing the hyperheuristic. It may thus be beneficial to start out with a small primitive set, before incrementally adding primitives if solution quality stays unacceptable.

In our experiments with the TSP domain, it was important to approach a problem with a medium-sized population when using tournament selection, as neither small nor high selection pressure appeared beneficial. It is also useful, if run time is acceptable and memory available, to rather increase the genotype size instead of the population size. These two approaches may also work well for problems other than TSP, but further research is needed to confirm this.

We conclude that, in addition to asking for only little domain knowledge of its user, the GP hyperheuristic, while being competitive, is also undemanding in terms of computing resources.

7 Further Work

In the future we intend to test the presented GP hyperheuristic on further real-world problems from several domains, again comparing the produced metaheuristics to domain-specific algorithms.

Also, it would be interesting to explore what happens if one breaks up low-level heuristics into their components and represents them as primitives. In this way, in principle, the hyperheuristic would be able to produce even more novel and powerful metaheuristics.

Furthermore, on the level of search guidance, we intend to have the GP hyperheuristic collect and use information on the topology of the search space of an underlying problem.

Acknowledgements

We would like to acknowledge financial support from EPSRC (grants EP/C523377/1 and EP/C523385/1) and to thank the anonymous reviewers for helpful comments.

References

[1] http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95/tsp/

[2] Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: On the Automatic Evolution of Computer Programs and its Applications. In: Genetic Programming – An Introduction, Morgan Kaufmann, San Francisco, CA, USA (1998)

[3] Brameier, M., Banzhaf, W.: Linear Genetic Programming. In: vol. 1, Genetic and Evolutionary Computation, Springer, Heidelberg (2006)

[4] Burke, E., Hyde, M., Kendall, G.: Evolving bin packing heuristics with genetic programming. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 860–869. Springer, Heidelberg (2006)

[5] Burke, E., Hyde, M., Kendall, G., Woodward, J.: Scalability of evolved on line bin packing heuristics. In: Proceedings of Congress on Evolutionary Computation, CEC 2007 (2007)

[6] Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6), 451–470 (2003)

[7] Chakhlevitch, K., Cowling, P.: Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 23–33. Springer, Heidelberg (2005)

[8] Dowsland, K., Soubeiga, E., Burke, E.: A simulated annealing hyper-heuristic for determining shipper sizes. European Journal of Operational Research 179(3), 759–774 (2007)

[9] Gaw, A., Rattadilok, P., Kwan, R.: Distributed choice function hyper-heuristics for timetabling and scheduling. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 495–497. Springer, Heidelberg (2005)

[10] Janikow, C.Z.: Constrained genetic programming. In: Hussain, T.S. (ed.) Advanced Grammar Techniques Within Genetic Programming and Evolutionary Computation, Orlando, Florida, USA, 13 July 1999, pp. 80–82 (1999)

[11] Jayalakshmi, G., Sathiamoorthy, S., Rajaram, R.: An hybrid genetic algorithm — a new approach to solve traveling salesman problem. International Journal of Computational Engineering Science 2(2), 339–355 (2001)

[12] Keller, R.E., Banzhaf, W.: The evolution of genetic code on a hard problem. In: Spector, L., Langdon, W.B., Wu, A., Voigt, H.-M., Gen, M. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisco, CA, July 7–11, 2001. pp. 50–66, Morgan Kaufmann, San Francisco, CA (2001)

[13] Keller, R.E., Poli, R.: Linear genetic programming of metaheuristics. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, July 7-11, 2007, ACM Press, London (2007)

[14] Keller, R.E., Poli, R.: Linear genetic programming of parsimonious metaheuristics. In: Proceedings of Congress on Evolutionary Computation (CEC 2007), Swissotel The Stamford, Singapore September 25-28 (2007)

[15] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)

[16] Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)

[17] Lawler, E., Lenstra, J., Kan, A.R., Shmoys, D. (eds.): The Travelling Salesman Problem. Wiley, Chichester (1985)

[18] Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
[19] Oltean, M.: Evolving evolutionary algorithms using linear genetic programming. Evolutionary Computation 13(3), 387–410 (Fall 2005)
[20] O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Genetic programming, vol. 4. Kluwer Academic Publishers, Dordrecht (2003)
[21] Ross, P.: Hyperheuristics. In: Burke, E., Kendall, G. (eds.) Search Methodologies, pp. 529–556. Springer, New York, Berlin (2005)
[22] Soubeiga, E.: Development and application of hyper-heuristics to personnel scheduling. PhD thesis, Computer Science, University of Nottingham (2003)
[23] Whigham, P.A.: Search bias, language bias, and genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, July 28–31, 1996, pp. 230–237, MIT Press, Cambridge (1996)
[24] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
[25] Wong, M.L., Leung, K.S.: Applying logic grammars to induce sub-functions in genetic programming. In: 1995 IEEE Conference on Evolutionary Computation, Perth, Australia, November 29 - 1 December, 1995, vol. 2, pp. 737–740. IEEE Press, Los Alamitos (1995)

Automatic Design of Vision-Based Obstacle Avoidance Controllers Using Genetic Programming

Renaud Barate and Antoine Manzanera

ENSTA - UEI, 32 bd Victor,
75739 Paris Cedex 15, France
renaud.barate@ensta.fr, antoine.manzanera@ensta.fr

Abstract. The work presented in this paper is part of the development of a robotic system able to learn context dependent visual clues to navigate in its environment. We focus on the obstacle avoidance problem as it is a necessary function for a mobile robot. As a first step, we use an offline procedure to automatically design algorithms adapted to the visual context. This procedure is based on genetic programming and the candidate algorithms are evaluated in a simulation environment. The evolutionary process selects meaningful visual primitives in the given context and an adapted strategy to use them. The results show the emergence of several different behaviors outperforming hand-designed controllers.

1 Introduction

Obstacle avoidance is an essential function for any mobile robot. Range sensors are the most commonly used devices for detecting obstacles but the accuracy of sonars depends on the angle of reflection and the material of the detected object and laser range sensors are expensive and can be harmful. More, both are active sensors which is undesirable for military applications for instance. On the contrary, video cameras are now cheap, low consumption and high resolution sensors. Nevertheless detecting obstacles with a single camera is a difficult problem. Different solutions exist, using either optical flow or more simple contextual information, but none is generic enough to deal with changing environments.

Ideally, the robot should be able to adapt itself dynamically to the current context and to select its behavior (here, the obstacle avoidance algorithm) in real time depending on its environment. The robot would learn those most efficient behaviors only by interacting with the environment. As a first step towards this goal, we propose here an offline method based on genetic programming for the automatic design of obstacle avoidance controllers within a given environment. The next steps will be to develop a set of algorithms adapted to different environments and to design a high level controller able to select in real time the best algorithm for the current environment. Our system uses a simulation environment to test different algorithms. We will show that the evolved solutions use relevant visual primitives and that they perform better than hand-designed controllers.

2 Inspiration and Principles

2.1 Vision Based Obstacle Avoidance

One commonly used method to perform obstacle avoidance using a single camera is based on the calculation of optical flow. Optical flow represents the perceived movement of the objects in the camera field of view. It is commonly represented by a vector field, each vector representing the displacement of the corresponding pixel. According to the parallax principle, the perceived displacements will be greater for nearer objects when the camera has a translational movement. A simple and robust approach for obstacle avoidance is to move the robot forward for a few seconds, to calculate the mean of the optical flow on the right and left sides of the image, then to turn the robot in order to balance the flow on each side. This technique is inspired by the behavior of bees and has been successfully applied to center a mobile robot in a corridor [1,2]. More recently it has been used for the control of an autonomous helicopter [3,4]. However, systems based on optical flow don't cope well with thin or lowly textured obstacles.

On the other hand, simple primitives can be used to extract different informations about the image. Those include threshold, Gaussian and Laplacian filters as well as orientation and frequency filters. In many cases, combinations of some of those primitives can deliver enough information to discriminate potential obstacles. For instance, Michels implemented a system to estimate depth from texture information in outdoor scenes [5]. Other obstacle avoidance systems use this kind of information to discriminate the floor from the rest of the scene and calculate obstacle distances in several directions [6,7,8]. Nevertheless those methods suppose that the floor may be clearly discriminated and they neglect potentially useful contextual information from the rest of the scene. Ideally, a good obstacle avoidance system should be able to use any visual clue.

2.2 Vision in Evolutionary Robotics

Evolutionary techniques have already been used for robotic navigation and the design of obstacle avoidance controllers but in general vision is either overly simplified or not used at all. For instance, Harvey used a 64x64 pixels camera but simplified the input to a few average values on the image [9]. Marocco used only a 5x5 pixels retina as visual input [10]. A description of several other evolutionary robotics systems can be found in [11,12] but most of them rely only on range sensors. On the other hand, genetic programming has been proved to achieve human-competitive results in image processing systems, e.g. for the detection of interest points [13,14]. Ebner also developed a navigation system for a robot using range sensors [15] but he didn't combine the two approaches for a vision-based navigation system. Parisian evolution has also been shown to produce very good results for obstacle detection and 3D reconstruction but those systems need two calibrated cameras [16,17].

To our knowledge, only Martin tried evolutionary techniques with monocular images for obstacle avoidance [18]. The structure of his algorithm is based on the floor segmentation technique and the evaluation is done with a database of hand labeled real world image. The advantage of such an approach is that the evolved algorithms are more likely to work well with real images than those evolved with computer rendered images. Nevertheless, it introduces an important bias since the algorithms are only selected on their ability to label images in the database correctly. In our work, the vision algorithms are not limited to a particular technique and the selection is based on a functional evaluation of the algorithms, that is their ability to avoid obstacles in the simulation environment.

3 Material and Methods

3.1 The Vision Algorithms

Generally speaking, a vision algorithm can be divided in three main parts: First, the algorithm will process the input image with a number of filters to highlight some features. Then these features are extracted, i.e. represented by a small set of scalar values. Finally these values are used for a domain dependent task, here to generate motor commands to avoid obstacles. We designed the structure of our algorithms according to this general scheme. First, the filter chain consists of spatial and temporal filters, optical flow calculation and projection that will produce an image highlighting the desired features. Then we compute the mean of the pixel values on several windows of this transformed image (feature extraction step). Finally those means are used to compute a single scalar value by a linear combination. Several scalar values can be produced with different filter chains, these values are then combined using scalar operators. A final step will generate a motor command using the resulting scalar value(s). We extended this purely vision based algorithmic structure by adding two scalar input variables: the goal location distance and heading relative to the robot position. These variables can be used for command generation along with the other scalar values.

An algorithm is represented as a tree, the leaves being input data, the root being output command, and the internal nodes being primitives (transformation steps). The program can use different types of data internally, i.e. scalar values, images, optical flow vector fields or motor commands. For each primitive, the input and output data types are fixed. Some primitives can internally store information from previous states, thus allowing temporal computations like the calculation of the optical flow. Fig. 1 shows an example program for a simple obstacle avoidance behavior based on optical flow. Here is the list of all the primitives that can be used in the programs and the data types they manipulate:

- **Spatial filters** (*input: image, output: image*): Gaussian, Laplacian, threshold, Gabor, difference of Gaussians, Sobel and subsampling filter.
- **Temporal filters** (*input: image, output: image*): pixel-to-pixel min, max, sum and difference of the last two frames, and recursive mean operator.
- **Optical flow** (*input: image, output: vector field*): Horn and Schunck global regularization method, Lucas and Kanade local least squares calculation and simple block matching method [19].

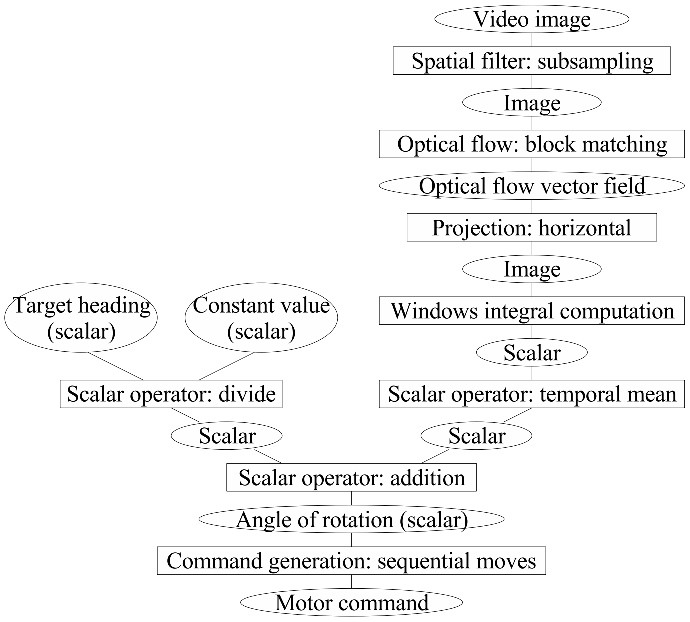

Video image
Spatial filter: subsampling
Image
Optical flow: block matching
Optical flow vector field
Projection: horizontal
Image
Windows integral computation
Scalar
Scalar operator: temporal mean
Target heading (scalar)
Constant value (scalar)
Scalar operator: divide
Scalar
Scalar
Scalar operator: addition
Angle of rotation (scalar)
Command generation: sequential moves
Motor command

Fig. 1. Algorithmic tree of a program example for obstacle avoidance. Rectangles represent primitives and ellipses represent data.

– **Projection** (*input: vector field, output: image*): Projection on the horizontal or vertical axis, Euclidean or Manhattan norm computation, and time to contact calculation using the flow divergence.
– **Windows integral computation** (*input: image, output: scalar*): The method used for this transformation is:
  1. A global coefficient α_0 is defined for the primitive.
  2. Several windows are defined on the left half of the image with different positions and sizes. With each window is paired a second window defined by symmetry along the vertical axis. A coefficient α_i and an operator ($+$ or $-$) are defined for each pair.
  3. The resulting scalar value R is a simple linear combination calculated with the following formula:

$$R = \alpha_0 + \textstyle\sum_{i=1}^{n} \alpha_i \mu_i$$
$$\mu_i = \mu_{\mathrm{L}i} + \mu_{\mathrm{R}i} \text{ or } \mu_i = \mu_{\mathrm{L}i} - \mu_{\mathrm{R}i}$$

  where n is the number of windows and $\mu_{\mathrm{L}i}$ and $\mu_{\mathrm{R}i}$ are the means of the pixel values over respectively the left and right window of pair i.

The number of windows pairs, their positions, sizes, operator and coefficient along with the global coefficient are characteristic parts of the primitive and will be customized by the evolutionary process.

– **Scalar operators** (*input: scalar(s), output: scalar*): Addition, subtraction, multiplication and division operators, temporal mean calculation and simple if-then-else test.
– **Command generation** (*input: scalar(s), output: command*): The motor command is represented by two scalar values: the requested linear and angular speeds. We created two operators to generate these values:
  - Direct generation: The required linear and angular speeds are two input scalar values.
  - Sequential moves: The movement is decomposed in straight moves and in-place rotations. This facilitates the usage of optical flow based strategies, since optical flow exploitation is more straightforward when the movement has no rotational part. This operator uses one input scalar value, which will be the angle of rotation at the end of a straight move.

Most of those primitives use parameters along with the input data to do their calculations (for example, the standard deviation value for the Gaussian filter). Those parameters are specific to each algorithm; they are randomly generated when the corresponding primitive is created by the genetic programming system described in Sect. 3.3.

3.2 Evaluation of Algorithms

For the evaluation of the different obstacle avoidance algorithms, we use a simulation environment in which the robot moves freely during each experiment. The simulation is based on the open-source robot simulator Gazebo. The simulation environment is a closed room of 36 m^2 area (6 m x 6 m) with block obstacles or furniture items depending on the experiments.

The simulation uses ODE physics engine for the movement of the robot and collisions detection and OpenGL for the rendering of the camera images. The physics engine update rate is 50 Hz while the camera update rate is 10 Hz. In all the experiments presented in this paper, the simulated camera produces 8-bits gray-value images of size 320x160 representing a field of view of 120° x 60°. This large field of view reduces the dead angles and hence facilitates obstacle detection and avoidance. All the obstacles are immovable to prevent the robot from just pushing them instead of avoiding them.

In each experiment, the goal of the robot is to go from a given starting point to a goal location without hitting obstacles. For that, we place the robot at the fixed starting point and let it move in the environment during 60 s driven by the obstacle avoidance algorithm. Two scores are attributed to the algorithm depending on its performance: a goal-reaching score G_1 rewards algorithms reaching or approaching the goal location, whereas score C_1 rewards the individuals that didn't hit obstacles on their way. Those scores are calculated with the following formulas:

$$G_1 = \begin{cases} t_{\mathrm{G}} \text{ if the goal is reached} \\ t_{\max} + d_{\min}/V \text{ else} \end{cases}$$
$$C_1 = t_{\mathrm{C}}$$

where t_G is the time needed to reach the goal in seconds, $t_{\max}$ is the maximum time in seconds (here 60 s), $d_{\min}$ is the minimum distance to the goal achieved in meters, V is a constant of 0.1 m/s and t_C is the time spent near an obstacle (i.e. less than 18 cm, which forces the robot to keep some distance away from obstacles).

We proceed then to a second run with a different starting point and a different goal location. Scores G_2 and C_2 are calculated the same way and the total scores G_{T} and C_{T} are obtained by adding the scores from the two runs:

$$G_{\mathrm{T}} = G_1 + G_2$$
$$C_{\mathrm{T}} = C_1 + C_2$$

The goal is hence to minimize those two scores G_{T} and C_{T}. Performing two different runs favors algorithms with a real obstacle avoidance strategy while not increasing evaluation time too much. The starting points are fixed because we want to evaluate all algorithms on the same problem.

3.3 The Evolution Process

We use genetic programming to evolve obstacle avoidance algorithms with the least possible *a priori* on the structure of the algorithm. Genetic programming, as introduced by J. Koza [20], is the evolution of computer programs by means of artificial evolution. Like other evolutionary algorithms, it is based on a selection-breeding process inspired by biological evolution which creates better algorithms by combining the primitives of the best algorithms of previous generations.

As usual with evolutionary algorithms, the population is initially filled with randomly generated individuals. The difficulty that arises with algorithms that use different data types is to ensure that the generated algorithms are valid. Montana introduced strongly-typed genetic programming to overcome this problem [21] and Whigham extended it by using a grammar to generate the algorithms [22]. We decided to use this grammar-based genetic programming as it also allows us to bias the search toward more promising primitives and to control the growth of the algorithmic tree.

In the same way that a grammar can be used to generate syntactically correct random sentences, a genetic programming grammar is used to generate valid algorithms. The grammar defines the primitives and data (the bricks of the algorithm) and the rules that describe how to combine them. The generation process consists in successively transforming each non-terminal node of the tree with one of the rules. This grammar is used for the initial generation of the algorithms and for the transformation operators. The crossover consists in swapping two subtrees issuing from identical non-terminal nodes in two different individuals. The mutation consists in replacing a subtree by a newly generated one. For clarity and space reasons, we cannot present the whole derivation process here but detailed explanations can be found in the paper from Whigham [22]. Table 1 presents the exhaustive grammar that we used in all our experiments.

The numbers in brackets are the probability of selection for each rule. A major advantage of this system is that we can bias the search toward the usage of more

Table 1. Grammar used in the genetic programming system for the generation of the algorithms

[1.0] START → COMMAND	[0.14] SPATIAL_FILTER → threshold
[0.5] COMMAND → sequentialMove(REAL)	[0.14] SPATIAL_FILTER → gabor
[0.5] COMMAND → directMove(REAL,REAL)	[0.14] SPATIAL_FILTER → differenceOfGaussians
[0.1] REAL → targetDistance	[0.14] SPATIAL_FILTER → sobel
[0.1] REAL → targetHeading	[0.15] SPATIAL_FILTER → subsampling
[0.1] REAL → scalarConstant	[0.2] TEMPORAL_FILTER → temporalMinimum
[0.05] REAL → add(REAL,REAL)	[0.2] TEMPORAL_FILTER → temporalMaximum
[0.05] REAL → subtract(REAL,REAL)	[0.2] TEMPORAL_FILTER → temporalSum
[0.05] REAL → multiply(REAL,REAL)	[0.2] TEMPORAL_FILTER → temporalDifference
[0.05] REAL → divide(REAL,REAL)	[0.2] TEMPORAL_FILTER → recursiveMean
[0.05] REAL → temporalRegularization(REAL)	[0.33] OPTICAL_FLOW → hornSchunck(IMAGE)
[0.05] REAL → ifThenElse(REAL,REAL,REAL,REAL)	[0.33] OPTICAL_FLOW → lucasKanade(IMAGE)
[0.4] REAL → windowsIntegralComputation(IMAGE)	[0.34] OPTICAL_FLOW → blockMatching(IMAGE)
[0.3] IMAGE → videoImage	[0.2] PROJECTION → horizontalProjection
[0.4] IMAGE → SPATIAL_FILTER(IMAGE)	[0.2] PROJECTION → verticalProjection
[0.15] IMAGE → PROJECTION(OPTICAL_FLOW)	[0.2] PROJECTION → euclideanNorm
[0.15] IMAGE → TEMPORAL_FILTER(IMAGE)	[0.2] PROJECTION → manhattanNorm
[0.15] SPATIAL_FILTER → gaussian	[0.2] PROJECTION → timeToContact
[0.14] SPATIAL_FILTER → laplacian	

promising primitives by setting a high probability for the rules that generate them. We can also control the size of the tree by setting small probabilities for the rules that are likely to cause an exponential growth (rules like REAL → ifThenElse(REAL,REAL,REAL,REAL) for example).

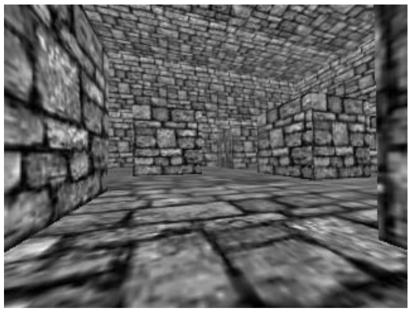
As described in the previous section, we wish to minimize two criteria G_{T} and C_{T}. There are different ways to use evolutionary algorithms to perform optimization on several and sometimes conflicting criteria. For the experiments described in this paper, we chose the widely used multi-objective evolutionary algorithm called NSGA-II introduced by K. Deb. We will not describe this algorithm here, as detailed and thorough explanations can be found in [23].

For the parameters of the evolution, we use a crossover rate of 0.8 and a probability of mutation of 0.01 for each non-terminal node. The population size is 100 individuals and the experiments last for 100 generations. We use a classical binary tournament selection in all our experiments. Those parameters were determined empirically with a few tests using different values. Due to the length of the experiments, we didn't proceed to a thorough statistical analysis of the influence of those parameters.

4 Experiments and Results

4.1 Experiment in a Simple Environment with Block Obstacles

The experiment described in this subsection uses only simple block obstacles. The goal is to cross the environment diagonally without hitting those block obstacles. We use a single brick texture for all the walls, floor, ceiling and blocks. We also manually created a simple obstacle avoidance controller for this environment based on optical flow. It manages to avoid some obstacles but it is quite slow and gets stuck in some situations. For this problem, the evolution created more efficient solutions, as shown on Figure 2.



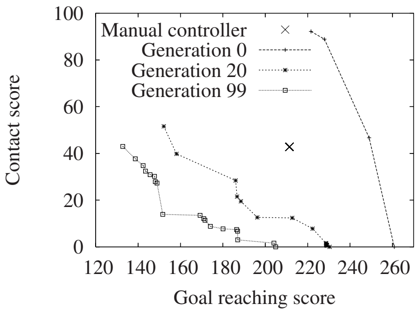


Fig. 2. Left: Snapshot of the simple environment with textured block obstacles. Right: Evolution of the best algorithms along the generations. Only the Pareto front for each generation is shown here.

Different kinds of controllers emerged, from slow ones only covering a few meters to fast ones quickly reaching the goal but bouncing on obstacles rather than avoiding them. The interesting point is that all those controllers use either an optical flow based technique, which is the most straightforward, or a Gabor filter detecting near obstacles. The evolution didn't find the ideal compromise between obstacle avoidance and speed but it selected reasonable and usable visual primitives for the environment efficiently. The resulting trajectories for the reference controller and two evolved ones are shown on Fig. 3.

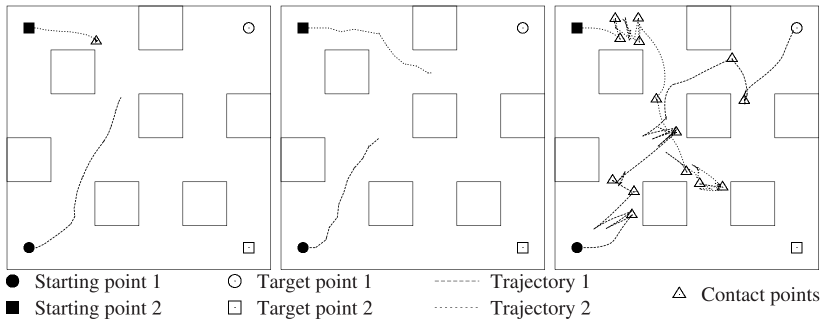


Fig. 3. Left: Trajectory of the reference controller. Middle: Trajectory of an evolved controller with a careful behavior. Right: Trajectory of a faster evolved controller bouncing on obstacles.

The two corresponding evolved controllers are shown on Fig. 4. It is interesting to note that these controllers use only a few primitives. In this kind of experiment, several primitives are seldom selected: edge detectors for example (Laplacian or Sobel) are almost never used, probably because the integral computation step is not adapted for the extraction of this kind of feature. On the

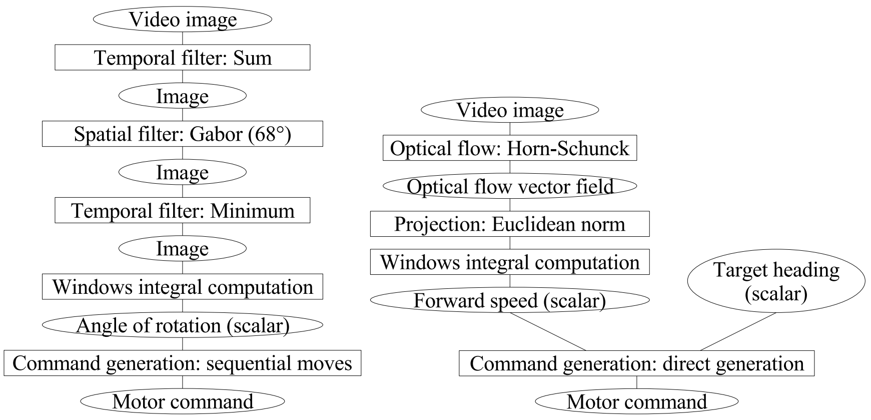

Video image
Temporal filter: Sum
Image
Spatial filter: Gabor (68°)
Image
Temporal filter: Minimum
Image
Windows integral computation
Angle of rotation (scalar)
Command generation: sequential moves
Motor command
Video image
Optical flow: Horn-Schunck
Optical flow vector field
Projection: Euclidean norm
Windows integral computation
Forward speed (scalar)
Target heading (scalar)
Command generation: direct generation
Motor command

Fig. 4. Left: Algorithm of an evolved controller with a careful behavior. Right: Algorithm of a faster evolved controller.

contrary, optical flow is very often used but mainly to detect near obstacles because the flow computation is very imprecise at high speed or with a rotary movement. Gabor filter is also often selected as it can detect textured obstacles at a given distance.

4.2 Experiment in a More Realistic Environment

In this experiment, the block obstacles have been replaced by bookshelves. The floor and the walls have different textures. The environment is less cluttered but the obstacles are larger than in the previous experiment. As there is more free space and the target location is nearer from the starting point, the experiment time is reduced to 30 seconds. We manually designed another reference controller to compare with the evolution results. It is also based on optical flow but it has been slightly changed to perform better in this environment. Figure 5 shows the environment and the performance of these controllers.

Once again, one of the evolved controllers avoids obstacles correctly but it is far from perfect as it doesn't reach the goal from the first starting point. Other evolved controllers reach the goal quickly but hit obstacles and walls several times on their way. This time, almost all the evolved controllers use direct integral computations on the image to detect obstacles. When approaching the bookshelves, the bottom of the shelves becomes more visible. Those areas are darker because the light comes from the ceiling in these experiments, thus with a simple integral computation in the upper part of the image, the robot brakes when approaching an obstacle. This obstacle detection technique is coupled with a target heading behavior and a seemingly random back and forth move that partially prevents the robot from getting stuck behind large obstacles. The combination of those three techniques achieves a rather good obstacle avoidance performance, as shown on Fig. 6.

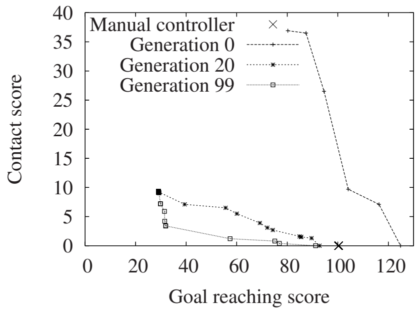


Fig. 5. Left: Snapshot of the environment with several furniture items. Right: Evolution of the best algorithms along the generations.

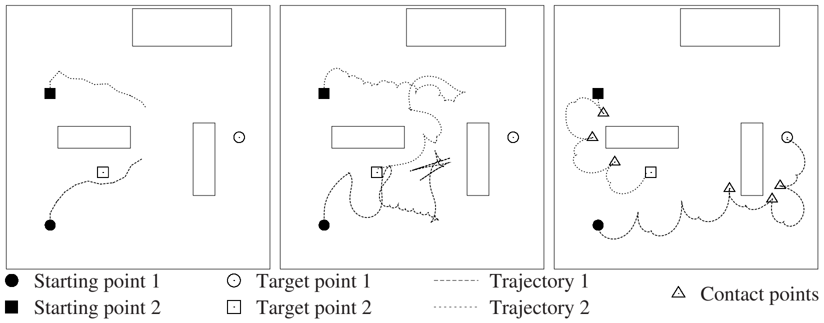


Fig. 6. Left: Trajectory of the hand-designed controller. Middle: Trajectory of an evolved controller avoiding obstacles. Right: Trajectory of another evolved controller quickly reaching the goal but not avoiding obstacles.

4.3 Discussion

The results of these two experiments show that the evolution process introduced in this paper is able to produce interesting solutions, generally outperforming hand-designed controllers. The evolution caused the emergence of original solutions to the obstacle avoidance problem depending on the context. The visual primitives selected in each case are coherent and can be used to avoid obstacles. This is promising for the next necessary step which will be the validation of the evolved algorithms on a real robot.

Nevertheless it should be noted that these performances are far from perfect. More, in other experiments with an even simpler environment where hand-designed controllers are really efficient, evolution gets stuck in local minima and achieves far worse performance. This underlines the major drawback of our approach at the moment: the population is much too small compared to the size of the search space, so evolution only explores a small part of it and hence misses many interesting solutions. We currently investigate solutions to overcome this problem without increasing the evolution time too much. The first possible solution would be to introduce a diversity metric to prevent premature convergence. This solution has been proved efficient on several problems but it is not straightforward to define a good diversity metric for tree based algorithms. An easier and probably more efficient solution would be to change population parameters. We could either split the population in several independent subpopulations or change the population size during the evolution to explore more solutions in the beginning. A combination of these two solutions is also possible and is likely to improve the results greatly. Another drawback is that our feature extraction operator is not adapted for linear or punctual features (edges or corners for instance). The addition of others feature extraction operators should increase the diversity of usable features.

5 Conclusion

In this paper, we used multi-objective genetic programming to create obstacle avoidance controllers making use of visual information. We use a simulation environment with computer rendered images to evaluate the candidate controllers. The evolution allowed the emergence of original strategies using relevant visual primitives in the environment. For future research, we intend to improve the results by controlling more precisely the population parameters of the evolution process. We also plan to design a more realistic simulation environment and robot model to evolve more robust controllers and to test them on a real robot. We will then adapt our system for the online selection of relevant controllers in order to develop strategies adapted to the visual context in real time. This will bring us closer to our goal of an adaptive and reactive robot able to face the complex and ever-changing environments of the real world.

References

1. Santos-Victor, J., Sandini, G., Curotto, F., Garibaldi, S.: Divergent stereo for robot navigation: learning from bees. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 434–439 (1993)
2. Ebner, M., Zell, A.: Centering behavior with a mobile robot using monocular foveated vision. Robotics and Autonomous Systems 32(4), 207–218 (2000)
3. Muratet, L., Doncieux, S., Brière, Y., Meyer, J.A.: A contribution to vision-based autonomous helicopter flight in urban environments. Robotics and Autonomous Systems 50(4), 195–209 (2005)
4. Hrabar, S.: Vision-Based 3D Navigation for an Autonomous Helicopter. PhD thesis, University of Southern California (2006)
5. Michels, J., Saxena, A., Ng, A.: High speed obstacle avoidance using monocular vision and reinforcement learning. In: Proceedings of the 22nd international conference on Machine learning, pp. 593–600 (2005)
6. Horswill, I.: Polly: A vision-based artificial agent. In: Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-1993), pp. 824–829 (1993)
7. Lorigo, L., Brooks, R., Grimson, W.: Visually-guided obstacle avoidance in unstructured environments. In: Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 373–379 (1997)

8. Ulrich, I., Nourbakhsh, I.: Appearance-based obstacle detection with monocular color vision. In: Proceedings of AAAI Conference, pp. 866–871 (2000)
9. Harvey, I., Husbands, P., Cliff, D.: Seeing the Light: Artificial Evolution, Real Vision. In: From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, pp. 392–401 (1994)
10. Marocco, D., Floreano, D.: Active vision and feature selection in evolutionary behavioral systems. From Animals to Animats 7, 247–255 (2002)
11. Mataric, M., Cliff, D.: Challenges in evolving controllers for physical robots. Robotics and Autonomous Systems 19(1), 67–83 (1996)
12. Walker, J., Garrett, S., Wilson, M.: Evolving controllers for real robots: A survey of the literature. Adaptive Behavior 11(3), 179–203 (2003)
13. Ebner, M., Zell, A.: Evolving a task specific image operator. In: Evolutionary image analysis, signal processing and telecommunications: First european workshop, evoiasp, pp. 74–89 (1999)
14. Trujillo, L., Olague, G.: Synthesis of interest point detectors through genetic programming. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 887–894 (2006)
15. Ebner, M., Zell, A.: Evolving a behavior-based control architecture-From simulations to the real world. In: Proceedings of Genetic and Evolutionary Computation Conference, vol. 2, pp. 1009–1014 (1999)
16. Pauplin, O., Louchet, J., Lutton, E., De La Fortelle, A.: Evolutionary Optimisation for Obstacle Detection and Avoidance in Mobile Robotics. Journal of Advanced Computational Intelligence and Intelligent Informatics 9(6), 622–629 (2005)
17. Olague, G., Puente, C.: Parisian evolution with honeybees for three-dimensional reconstruction. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 191–198 (2006)
18. Martin, M.: Evolving visual sonar: Depth from monocular images. Pattern Recognition Letters 27(11), 1174–1180 (2006)
19. Barron, J., Fleet, D., Beauchemin, S.: Performance of optical flow techniques. International Journal of Computer Vision 12(1), 43–77 (1994)
20. Koza, J.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge, MA, USA (1992)
21. Montana, D.: Strongly Typed Genetic Programming. Evolutionary Computation 3(2), 199–230 (1995)
22. Whigham, P.: Grammatically-based genetic programming. In: Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, pp. 33–41 (1995)
23. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

Generating SAT Local-Search Heuristics Using a GP Hyper-Heuristic Framework

Mohamed Bader-El-Den and Riccardo Poli

Department of Computing and Electronic Systems, University of Essex, UK

Abstract. We present GP-HH, a framework for evolving local-search 3-SAT heuristics based on GP. The aim is to obtain "disposable" heuristics which are evolved and used for a specific subset of instances of a problem. We test the heuristics evolved by GP-HH against well-known local-search heuristics on a variety of benchmark SAT problems. Results are very encouraging.

1 Introduction

Hyper-heuristics could simply be defined as "heuristics to choose other heuristics" [4]. A heuristic is considered as "rule of thumb" or "educated guess" that reduces the search required to find a solution. The difference between meta-heuristics and hyper-heuristics is that the former operate directly on the targeted problem search space with the goal of finding optimal or near optimal solutions. The latter, instead, operate on the heuristics search space (which consists of the heuristics used to solve the targeted problem). The goal then is finding or generating high-quality heuristics for a target problem, for a certain class of instances of a problem, or even for a particular instance.

There are two main classes of hyper-heuristics. In a first class of hyper-heuristic systems, which we term *HH-Class 1*, the system is provided with a list of preexisting heuristics for solving a certain problem. Then the hyper-heuristic system tries to discover what is the best sequence of application for these heuristics for the purpose of finding a solution. Different techniques have been used to build hyper-heuristic systems of this class. Algorithms used to achieve this include, for example: tabu search [5], case-based reasoning [6], genetic algorithms [7], ant-colony systems [22], and even algorithms inspired to marriage in honey-bees [1].

The second approach used to build hyper-heuristic systems aims at evolving new heuristics by making use of the *components* of known heuristics. We term this class *HH-Class 2*. This is the approach we will adopt also in this paper. The process starts simply by selecting a suitable set of heuristics that are known to be useful in solving a certain problem. However, instead of directly feeding these heuristics to the hyper-heuristic system (as an in *HH-Class 1* discussed above), the heuristics are first decomposed into their basic components. Different heuristics may share different basic components in their structure. However, during the decomposition process, information on how these components were connected with one another is lost. To avoid this problem, this information is captured by a grammar. So, in order to provide the hyper-heuristic systems with enough information on how to use components to create valid heuristics, one must first construct an appropriate grammar. Hence, in the hyper-heuristics in *HH-Class 2*, both the grammar and the heuristics components are given to the hyper-heuristic systems. The system then uses a suitable evolutionary algorithm to evolve new heuristics. For example, in recent work [3] genetic programming [13,14] was successfully used to evolve new heuristics in *HH-Class 3* for one-dimensional online bin packing problems. Very positive results with evolving offline bin-packing heuristics have recently been obtained in [16] where GP was used to evolve strategies to guide a fixed solver. In general, we can say that the *HH-Class 2* approach has more freedom to create new heuristics for a given problem than *HH-Class 1*. However, *HH-Class 1* is easier to implement since it does not require the decomposition of heuristics nor the use of a grammar.

The long term goal in our research is investigating the use of GP as a hyper-heuristic framework for evolving instance-dependent heuristics. That is, the aim is not to obtain general heuristics, but effectively “disposable” heuristics which are evolved and used for a specific instance of a problem. Here, we will make a first step in this direction, by exploring the evolution of heuristics which are specialised to solve *specific subsets of instances* of a problem. In particular we evolve heuristics specialised to solve SAT problems with a fixed number of variables. We do this with a grammar based strongly-typed GP hyper-heuristic system, which we call *GP-HH*.

2 SAT Problem

The target in the satisfiability problem (SAT) is to determine whether it is possible to set the variables of a given Boolean expression in such a way to make the expression true. The expression is said to be satisfiable if such an assignment exists. If the expression is satisfiable, we often want to know the assignment that satisfies it. The expression is typically represented in Conjunctive Normal Form (CNF), i.e., as a conjunction of clauses, where each clause is a disjunction of variables or negated variables.

There are many algorithms for solving SAT. Incomplete algorithms attempt to guess an assignment that satisfies a formula. So, if they fail, one cannot know whether that’s because the formula is unsatisfiable or simply because the algorithm did not run for long enough. Complete algorithms, instead, effectively *prove* whether a formula is satisfiable or not. So, their response is conclusive. They are in most cases based on backtracking. That is, they select a variable, assign a value to it, simplify the formula based on this value, then recursively check if the simplified formula is satisfiable. If this is the case, the original formula is satisfiable and the problem is solved. Otherwise, the same recursive check is done using the opposite truth value for the variable originally selected.

The best complete SAT solvers are instantiations of the Davis Putnam Logemann Loveland procedure [8]. Incomplete algorithms are often based on local

Algorithm 1. General algorithm for SAT stochastic local search heuristics

```
$L$ = initialise the list of variables randomly
for $i = 0$ to MaxFlips do
   if $L$ satisfies formula $F$ then
      return $L$
   end if
   select variable $V$ using some selection heuristic
   flip $V$ in $L$
end for
return no assignement satisfying $F$ found
```

search heuristics (see Section 2.1). These algorithms can be extremely fast, but success cannot be guaranteed. On the contrary, complete algorithms guarantee success, but they computational load can be considerable, and, so, they cannot be used for large SAT instances.

2.1 Stochastic Local-Search Heuristics

Stochastic local-search heuristics have been widely used since in the early 90s for solving the SAT problem following the successes of GSAT [21]. The main idea behind these heuristics is to try to get an educated guess as to which variable will most likely, when flipped, give us a solution or to move us one step closer to a solution. Normally the heuristic starts by randomly initialising all the variables in the CNF formula. It then flips one variable at a time, until either a solution is reached or the maximum number of flips allowed has been exceeded. Algorithm 1 shows the general structure of a typical local-search heuristic for the SAT problem. The algorithm is normally repeatedly restarted for a certain number of times if it is not successful.

2.2 Evolutionary Algorithms and SAT Problem

Different evolutionary techniques have been applied to the SAT problem. There are two main research directions: direct evolution and evolution of heuristics.

An example of methods in the first direction – direct evolution – is FlipGA which was introduced by Marchiori and Rossi in [15]. There a genetic algorithm was used to generate offspring solutions to SAT using the standard genetic operators. However, offspring were then improved by means of local search methods. The same authors later proposed, ASAP, a variant of FlipGA [17]. A good overview of other algorithms of this type is provided in [12].

The second direction, which we also adopt in this paper, is to use evolutionary techniques to automatically evolve local search heuristics. A successful example of this is the CLASS system developed by Fukunaga [9,10]. The process of evolving new heuristics in the CLASS system is based on five conditional branching cases (if-then-else rules) for combining heuristics. Effectively CLASS can be considered as a very special type of the genetic programming system where these rules are used instead of the standard GP operators (crossover and mutation). The results of the evolved heuristics were competitive with a number of human-designed heuristics. However, the evolved heuristics were relatively slow. This is because the conditional branching operations used evaluate two heuristics first and they then select the output of one to decide which variable to flip. Also, restricting evolution to use only conditional branching did not give the CLASS system enough freedom to evolve heuristics radically different from the human-designed heuristics (effectively, the evolved heuristic are made up by a number of nested heuristics). Another example of system that evolves SAT heuristics is the STAGE system introduced by Boyan and Moore in [2]. STAGE tries to improve the local search performance by learning (online) a function that predicts the output of the heuristic based on some characteristics seen during the search.

3 GP-HH for SAT

To construct a grammar suitable to guide GP-HH in the solution of SAT problems, we used a number of the well-know local-search heuristics. We decomposed these heuristics into their basic components. The heuristics considered are the following:

– GSAT: [21] which, at each iteration, flips the variable with the highest gain score, where the gain of the variable is the difference between the total number of satisfied clauses after flipping the variable and the current number of satisfied clauses. The gain is negative if flipping the variable reduces the total number of satisfied clauses.
– HSAT: [11] In GSAT more than one variable may present the maximum gain. GSAT chooses among such variables randomly. HSAT, instead, uses a more sophisticated strategy. Its selects the variable with the maximum age, where the age of the variable is the number of flips since it is was last flipped. So, the most recently flipped variable has an age of zero.
– GWSAT: [19] with probability p selects a variable occurring in some unsatisfied clauses while with probability $(1-p)$ flips the variable with maximum gain as in GSAT.
– WalkSat: [20] starts by selecting one of the unsatisfied clauses C. Then it flips randomly one of the variables that have a gain score of 0 (leading to a “zero-damage” flip). If none of the variables in C has a “zero-damage” characteristic, it selects with probability p the variable with the maximum score gain, and with probability $(1-p)$ a random variable in C.

We have designed a simple but flexible grammar, which gives GP-HH enough freedom to evolve really new heuristics. By analysing the previous heuristics, we classified the main components of these heuristics into two main groups. The first group of components, Group 1, returns *a variable* from an input list of variables (e.g., the selection of a random variable from the list or of the variable with highest gain score). The second group, Group 2, returns *a list of variables* from the CNF formula (e.g., the selection of a random unsatisfied clause which, effectively, returns a list of variables).

After trying different grammar representations we decided to design the grammar in such a way to produce nested functions to avoid using variables for passing data from a function to another. The aim was to reduce the constraints on the crossover and mutation operators, and to make the GP tree representing each individual simpler. The grammar we used and its components are shown in Figure 1.

In Group 2 we have two more components which are not directly taken from the list of heuristics above. The first, which we call ALL_USC (which stands for all unsatisfied clauses), returns a list of non-repeated variables found in all the unsatisfied clauses. We found that this component performed well especially on instances with a relatively small number of variables, as will be shown later. The second additional component, which we call RAND_USC (which stands for random unsatisfied clause), returns the variables in a randomly selected clause. The main difference between RAND_USC and USC, which also returns a random unsatisfied clause, is that USC returns the same unsatisfied clause during the course of the execution of a heuristic, while RAND_USC randomly selects a different clause each time it is invoked.

The primitive SCR_Z selects a zero-damage variable (as in WalkSAT). We placed this component in group 2. It returns the input list if no variable with zero score gain is found. If, instead, a zero-damage variable is found, it returns a list which includes this variable only.

Most primitives which accept a list as input are provided in two versions: one with a single list argument, and one with a list and an object of type op. The non-terminal symbol op in the grammar specifies how to break ties between variables whenever multiple variables in a list satisfy a selection criterion. When op is not provided, a default tie-breaking strategy is used. For example, in MAX_SCR – the component that returns the variable with the highest score gain – if multiple variables have the same highest score, the first variable is returned by default. However, if the optional parameter op is provided and it is TIE_AGE, the tie will be broken by favouring the variable which has least recently been flipped. In some cases a specific option may have no meaning with a particular component. For example, TIE_SCR breaks ties by favouring the variable with highest score. Naturally, when used in conjunction with MAX_SCR this option has no effect.

We also included probabilistic branching components (IFV and IFL) in our heuristics. We classify branching components on the basis of their return type. For example, if the branch is between selecting a random variable from a list and selecting the variable with the highest gain score, we consider this probabilistic branching component as in Group 1 since it returns a variable. The parameter prob represents the probability of returning the first argument of an IFV or an IFL primitive.

The grammar in Figure 1 could describe any of the heuristics discussed above. For example, a statement describing the GWSAT heuristic with a noise parameter of 0.5 could be written as `FLIP IFV 50, MAX_SCR ALL, TIE_RAND, RANDOM USC`, where ALL returns all the variables in the CNF formula, TIE_RAND stands for "break ties randomly", MAX_SCR selects a variable with highest score

```
start →   FLIP v
v     →   RANDOM l
        | MAX_SCR l | MAX_SCR l, op
        | IFV prob, v, v
        | MIN_SCR l | MIN_SCR l, op
        | MAX_AGE l | MAX_AGE l, op
l     →   ALL | ALL_USC
        | RAND_USC | USC
        | IFL prob, l, l
        | SCR_Z l | SCR_Z l, op
op    →   TIE_RAND | TIE_AGE
        | TIE_SCR | NOT_ZERO_AGE
prob  →   20 | 40 | 50 |
          70 | 80 | 90
```

Fig. 1. The grammar used for evolving heuristics for SAT using GP-HH

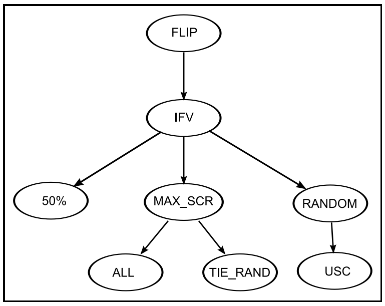


Fig. 2. GWSAT heuristic represented using the grammar adopted in GP-HH

and `RANDOM` selects a random variable from `USC` (unsatisfied clause). A tree representation of this individual is shown in Figure 2.

4 Experimental Setup

We have implemented the full genetic programming hyper-heuristic framework for the SAT problem in C++ compiled with the gcc compiler. The system consists of two main parts: a grammar based GP engine and a SAT engine for handling SAT formulas.

The GP-HH system was applied to solve benchmark cases taken from the uniform random 3-SAT library SatLib.[1] All the problems in our benchmarks were satisfiable uniform random 3-SAT problems with 20, 50, 75 and 100 variables.

[1] A full set of benchmarks is available from `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

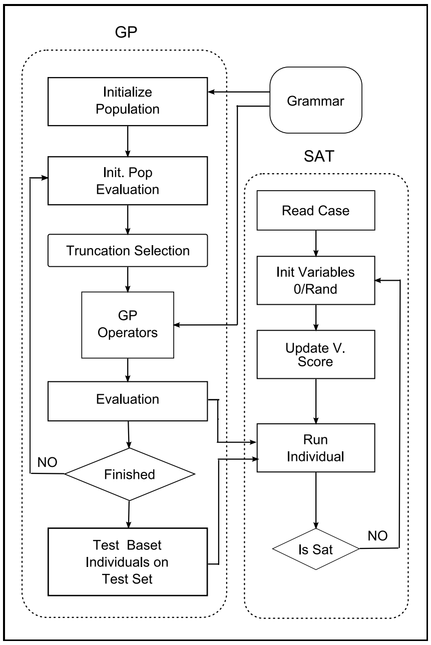


Fig. 3. Operations and interactions in the GP-HH framework

To reiterate, the objective of the experiments was to evolve a separate heuristic that best performs on SAT instances of a given size, and not a general heuristics for 3-SAT problem. So, we are not trying to evolve heuristics that compete with general SAT solvers, although, as it will become clear later, unexpectedly we obtained solvers with a considerable degree of generality.

Normally a local-search heuristic starts by randomly initialising all the variables in the formula to either zero or one. We did this in testing. However, during evolution we started the evolved heuristics with all variables set to zero. This may have slightly reduced the total number of solved cases and may even have slightly increased the mean number of flips required by each heuristics in each run. We used this approach, however, because it reduces the randomness in the evolutionary process and makes it easier to compare results. Once again, this was done only during the evolution of heuristics, while in testing we initialised all the variables randomly, as customary.

The GP system initialises the population by using the grammar and selecting random primitives out of the functions and terminals that are consistent with the grammar. So, all initial heuristics are guaranteed to be syntactically valid SAT heuristic. The population is then manipulated by the following operators:

– We use truncation selection, where only the best 40% of the population is allowed to reproduce.

Table 1. GP parameters

SAT set	Population size	Crossover rate	Mutation rate	Fitness cases	Max Flips
uf20	300	35%	1%	80	1000
uf50	250	40%	1%	100	4000
uf75	300	40%	1%	40	10000
uf100	250	40%	1%	100	12000

- Offspring are created using a specialised form of crossover. A random crossover point is selected in the first parent, then the grammar is used to select the crossover point from the second parent. It is randomly selected from all valid crossover points. If no point is available, the process is repeated again from the beginning until crossover is successful.
- Single point mutation is applied to 1% of the population. Again the grammar is used to ensure that all individuals are valid heuristics throughout the course of evolution.
- Individuals that have not been effected by any genetic operator are not evaluated again to reduce the computation cost of the evolution phase.

Figure 3 shows how the framework works and how the interaction between the two main engines, GP and SAT, operates.

As we mentioned before, we apply GP-HH to discover high-quality SAT solvers specialised for SAT instances of a particular size. So, we pass to the system sets of SAT instances all with the same number of variables. These form a training set of fitness cases on which individuals are tested. The fitness of each individual is based on three factors: a) how many cases have been solved out of the given fitness cases (SAT instances), b) the mean number of flips needed in the solved cases, and c) how many primitives (nodes) are present in an individual. Table 1 summarises the GP parameters used for each set of benchmarks.

During the evaluation of the initial population, we use only a fraction of the training set. Also, the number of maximum flips allowed is smaller, than during the other generations. This is done to reduce the computation load involved with the evaluation of the initial population. Since this is randomly generated, a high percentage of individuals have very low performance. These time saving techniques help filter them out quickly.

Although the initial population in GP-HH is randomly generated and includes no handcrafted heuristics, individuals representing GSAT, HSAT and GWSAT were created in the initialisation in almost all experiments we did. This is because of their simple representation with our grammar. This gave evolved heuristic a chance to start competing with previously known good heuristics from the beginning. In some cases the standard heuristics dominated the early generations of runs. Nonetheless, GP was always able to eventually discover new and better heuristics, despite our using in all our training and testing sets hard SAT instances, where the clause-to-variable ratio is grater than or equal to 4.3. None of the instance used in testing and comparing the evolved heuristics have been used in the evolution phase.

5 Results

Evolving heuristics for SAT is hard, with each GP-HH run taking between a few hours to several days (for the biggest training sets) to complete. So, we cannot provide here a statistical analysis of GP-HH runs. All we can say is that most of our runs successfully evolved high-quality heuristics for the SAT instances in their training set. We feel that this deficiency is acceptable, since this is one of those cases where one is more interested in the *results* of a set of runs rather than the runs themselves, since the results of our runs are actual problem solvers. These we can study in great detail. So, in this section we present some of the results of the evolved heuristics for each instance set of the 3-SAT problem. We also compare the performance of these heuristics with that of well-known local-search SAT heuristics.

We start by showing a typical example of the heuristics evolved using the GP-HH framework. Figure 4 shows one of the best performing heuristics evolved for the 50-variables instance set (brackets were introduced to increase readability). As one can see evolved heuristics are significantly more complicated than the standard heuristics we started from (e.g., GWSAT). So, a manual analysis of how the component steps of an evolved heuristic contribute to its overall performance is very difficult.

```
  FLIP( IFV( 90, IFV( 40, MAX_SCR( ALL,
TIE_RAND), IFV( 70, RANDOM(USC)), IFV( 80,
 MAX_SCR( RAN_USC, NOT_ZERO_AGE), IFV( 20,
MAX_SCR( ALL, TIE_RAND), MAX_SCR( RAND_USC,
TIE_AGE))))), IFV( 80, IFV( 50, MAX_SCR( ALL,
  TIE_AGE), MAX_SCR( RAND_USC, TIE_RAND)),
     MAX_SCR( IFL( 70, ALL_USC, USC)
             NOT_ZERO_AGE)))
```

Fig. 4. SAT heuristics evolved by GP-HH. Training set taken from the uf50 benchmark set.

However, it is possible to characterise the performance of SAT local search heuristics using certain numerical measures [18]. Depth and mobility are, perhaps, the two most important ones. *Depth* measures how many clauses remain unsatisfied during the execution of a heuristic. *Mobility* is a measure of how rapidly the heuristic moves in the search space. In general it is desirable to have algorithms with large mobility values which indicate that the heuristic is moving rapidly in the search space. Instead, it is better to have small values of depth, indicating that the average number of unsatisfied clauses is small during the course of execution of the heuristic.

Table 2 compares the depth and mobility of the GP-HH evolved heuristics against depth and mobility of reference human-designed heuristics. The comparison is done on the uf100-0953 SATLib benchmark, which consists of SAT instances with 100 variables and 430 clauses. The results for GSAT, HSAT and

Table 2. Comparison of evolved and known SAT solvers on the uf100-953 instance set

Solver	Mean depth	Mean mobility	Mean flips
GSAT()	2.13	5.7	99,006
WSAT(0.5)	5.65	15.7	9,421
Novelty(0.5)	4.76	18.9	4,122
GPHH100a	5.23	35,2	6,864
GPHH50a	8.17	42.9	11,154

WSAT are taken form [18]. In this table we show two of the local search heuristics evolved using GP-HH, `GPHH100a`, which was evolved using 100-variable instances, and `GPHH50a`, which was evolved on 50-variable instances. In both cases SAT training instances were taken from the SATLib benchmark library. The results show that `GPHH100a` performs better than GSAT, HSAT and WSAT in terms of mobility and the average the number for flips used. However, GSAT and HSAT have lower (better) values of depth. This is because they use a very large number of flips, which cause these algorithms to have a smaller average number of unsatisfied clauses. `GPHH100a` did not outperform the Novelty heuristic, but the results are very close. We think this is a good result because `Novelty` is an extremely high performing heuristics and it wasn't one of the heuristics decomposed to construct our GP-HH grammar. So, we hope that by including components from `Novelty` in future research we may be able to further improve GP-HH. Table 2 also shows the performance of `GPHH50a`, that was trained on 50-variable instances. Despite this, it appears to perform rather well also on instances with 100 variable, showing some generalisation capability.

Some benchmark suites consisting of a number of SAT instances with between 30 and 100 variables were used in [12], where comparative results between a number of heuristics, some of which evolutionary, were presented. These suites have been used in a number of other studies. So, we chose the same suites to perform a wider range of tests on our evolved heuristics. In particular, we used Suite A, which encompasses instance with 30, 40, 50 and 100 variables, and Suite B, which includes instances with 50, 75 and 100 variables. More details can be found in [12].

In Tables 3 and 4 we provide comparative results of the GP-HH heuristics against other state-of-the-art evolutionary heuristics and human-designed heuristics on Suites A and B. The results of the GP-HH evolved heuristics are averages of 5 runs on the benchmark sets. In Tables 3 and 4 the results of the FlipGA and WSAT are taken from [12], while in Table 4 the results for `Novelty+` and `C2-D3` are taken from [10]. The number of runs of these heuristics on the suites varied from 4 to 10. Note that we are testing evolved heuristics on *all* the instances in the suites. So, for example, heuristics evolved for 50 variable instances are also tested on the 75 and 100 variables instances. This gives us an indication of how general the heuristics are, though a thorough analysis of this issue is beyond the scope of this study.

In Table 3 and 4 two measures of the heuristics performance are shown: the success rate (SR) on the set and the average number of flips (AF) used by each

Table 3. Results for benchmark suite A. SR=success rate, AF=average number of flips (out of a maximum of 300,000). Results for FlipGA and WSAT are taken from [12].

	$n = 30$		$n = 50$		$n = 100$	
	SR	AF	SR	AF	SR	AF
FlipGA	1.0	25,490	1.0	127,300	0.87	116,653
WSAT	1.0	1,631	1.0	15,384	0.8	19,680
GPHH100a	1.0	1,864	1.0	12,872	0.92	54,257
GPHH50a	1.0	2,035	1.0	16,273	0.84	24,549
GPHH20a	1.0	1,457	0.95	18,384	0.66	32,781

Table 4. Results for benchmark suite B

	$n = 50$		$n = 75$		$n = 100$	
	SR	AF	SR	AF	SR	AF
FlipGA	1.0	103,800	0.82	29,818	0.57	20,675
WSAT	0.95	16,603	0.84	33,722	0.6	23,853
Novelty+	N/A	N/A	0.966	17,018	0.716	34,849
C2-D3	N/A	N/A	0.972	19,646	0.786	40,085
GPHH100a	0.96	12,527	0.93	27,975	0.74	41,284
GPHH75a	1.0	18,936	0.95	26,571	0.59	29,495
GPHH50a	0.97	11,751	0.81	36,215	0.46	22,648

heuristic. The results show that the heuristics evolved by GP-HH performed well compared to most local-search heuristics, outperforming some. The tables also show that the heuristics evolved by GP-HH outperformed FlipGA in terms of both the success rate and average number of flips.

From the results it can also be noticed that in some cases heuristics evolved for a instances with a larger number of variables have a considerable degree of generality, performing well also on problems with a smaller number variables.

6 Conclusion

In this paper we presented GP-HH, a framework for evolving "disposable" heuristics for the SAT problem, i.e., heuristics that are relatively fast to evolve and are specialised to solve specific sets of instances of the problem. We presented a comparison between GP-HH and other well-known evolutionary and local-search heuristics. The results show that the heuristics produced by GP-HH are competitive with these.

GP-HH produced heuristics that are on par with some of the best-known SAT solvers. We consider this a success. However, the heuristics evolved using CLASS2 are slightly better than the ones evolved by GP-HH. As mentioned in [10], these heuristics are slower than ours. This is because of the use of conditional branching as a GP primitive. As mentioned in Section 2.2 in most cases this requires to run two heuristics. We don't use this form of conditional branching (our branching instructions are probabilistic branches). So, GP-HH heuristics are faster than CLASS2 ones. Also the CLASS2 system used a large training sets and much longer evolutionary runs compared to GP-HH. It remains to be explored if by including more components in the grammar (e.g., those from `Novelty`), performing longer runs and feeding the GP-HH with a set of the well performing heuristics in the initial population, GP-HH could outperform CLASS2. This will be the target of our future research.

Furthermore, in future work we intend to test and evolve heuristics for a wider range of SAT problems. We also want to study the behaviour of GP-HH in more detail. In addition, we aim to further speed up evolution. Like most other GP systems, GP-HH populations include a large numbers of repeated individuals. So, a natural speed-up technique is to avoid the evaluation of repeated individuals. Additional savings could be obtained by avoiding the evaluation of repeated subtrees. We will also apply GP-HH to different combinatorial optimisation problems, e.g., job shop scheduling.

Acknowledgements

The authors acknowledge financial support from EPSRC (grants EP/C523377/1 and EP/C523385/1).

References

1. Abbass, H.A.: MBO: Marriage in honey bees optimization - A haplometrosis polygynous swarming approach. In: Proceedings of the, Congress on Evolutionary Computation CEC2001, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30. pp. 207–214, IEEE Press, Los Alamitos (2001)
2. Boyan, J., Moore, A.: Learning evaluation functions to improve optimization by local search. Journal of Machine Learning Research 1, 77–112 (2000)
3. Burke, E.K., Hyde, M.R., Kendall, G.: Evolving bin packing heuristics with genetic programming. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 860–869. Springer, Heidelberg (2006)
4. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)
5. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6), 451–470 (2003)
6. Burke, E.K., Petrovic, S., Qu, R.: Case-based heuristic selection for timetabling problems. Journal of Scheduling 9(2), 115–132 (2006)
7. Cowling, P., Kendall, G., Han, L.: An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Fogel, D.B., El-Sharkawi, M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.) Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, pp. 1185–1190. IEEE Press, Los Alamitos (2002)
8. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM 5(7), 394–397 (1962)

9. Fukunaga, A.: Automated discovery of composite SAT variable selection heuristics. In: Proceedings of the National Conference on Artificial Intelligence, pp. 641–648. AAAI, Menlo Park (2002)
10. Fukunaga, A.: Evolving local search heuristics for SAT using genetic programming. In: Deb, K., al., e. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 483–494. Springer, Heidelberg (2004)
11. Gent, I.P., Walsh, T.: Towards an understanding of hill-climbing procedures for sat. In: Proc. of AAAI-1993, Washington, DC, pp. 28–33 (1993)
12. Gottlieb, J., Marchiori, E., Rossi, C.: Evolutionary algorithms for the satisfiability problem. Evol. Comput. 10(1), 35–50 (2002)
13. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
14. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)
15. Marchiori, E., Rossi, C.: A flipping genetic algorithm for hard 3-SAT problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H, Honavar, V., Jakiela, M., Smith, R.E., (eds). Proceedings of the Genetic and Evolutionary Computation Conference Orlando, Florida, USA, 13-17, 1999, vol. 1, pp. 393–400, Morgan Kaufmann, San Francisco (1999)
16. Poli, R., Woodward, J., Burke, E.: A histogram-matching approach to the evolution of bin-packing strategies. In: Proceedings of the IEEE Congress on Evolutionary Computation, Singapore (accepted, 2007)
17. Rossi, C., Marchiori, E., Kok, J.N.: An adaptive evolutionary algorithm for the satisfiability problem. SAC 1, 463–469 (2000)
18. Schuurmans, D., Southey, F.: Local search characteristics of incomplete SAT procedures. Artificial Intelligence 132(2), 121–150 (2001)
19. Selman, B., Kautz, H.: Domain-independent extensions to GSAT: solving large structured satisfiability problems. In: Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI-1993), Chambéry, France (1993)
20. Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. In: Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 1994), Seattle, pp. 337–343 (1994)
21. Selman, B., Levesque, H.J., Mitchell, D.: A new method for solving hard satisfiability problems. In: Rosenbloom, P., Szolovits, P. (eds.) Proceedings of the Tenth National Conference on Artificial Intelligence, Menlo Park, California, pp. 440–446. AAAI Press, Menlo Park (1992)
22. Silva, D.L., O'Brien, R., Soubeiga, E.: An ant algorithm hyperheuristic for the project presentation scheduling problem. In: Fogel, D.B., El-Sharkawi, M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.) Proceedings of the 2005 IEEE Congress on Evolutionary Computation, pp. 92–99 (2005)

Magnetic Resonance Image Segmentation Based on Two-Dimensional Exponential Entropy and a Parameter Free PSO

Amir Nakib, Yann Cooren, Hamouche Oulhadj, and Patrick Siarry

Université de Paris XII, Laboratoire Images, Signaux et Systèmes Intelligents
(LISSI, E. A. 3956), 61 avenue du Général De Gaulle
94010 Créteil, France
{nakib,cooren,oulhadj,siarry}@univ-paris12.fr

Abstract. In this paper, a magnetic resonance image (MRI) segmentation method based on two-dimensional exponential entropy (2DEE) and parameter free particle swarm optimization (PSO) is proposed. The 2DEE technique does not consider only the distribution of the gray level information but also takes advantage of the spatial information using the 2D-histogram. The problem with this method is its time-consuming computation that is an obstacle in real time applications for instance. We propose to use a parameter free PSO algorithm called TRIBES, that was proved efficient for combinatorial and non convex optimization. The experiments on segmentation of MRI images proved that the proposed method can achieve a satisfactory segmentation with a low computation cost.

Keywords: image segmentation, two-dimensional exponential entropy, particle swarm optimization, tribes, parameter free.

1 Introduction

The increasing need for analyzing the brain magnetic resonance images (MRI) allowed to establish MRI segmentation as an important research field. For instance, in order to make easy the evaluation of the ventricular space evolution, a multilevel MRI segmentation is required. In this paper, we consider the problem of detecting the ventricular space from MRI of the brain. The image segmentation problem at hand is difficult because of the common occurrence of peri-ventricular lesions in MRI of even normal aging subjects, which locally alter the appearance of the white matter surrounding the ventricular space.

The segmentation problem has received a great deal of attention, thus any attempt to survey the literature would be too space-consuming. The most popular segmentation methods (tissue classification methods) may be found in [1] to [13]. The common class of parametric methods used in brain MRI segmentation is based on an expectation-maximization framework. This class of methods is based on the assumption that a mixture Gaussian distribution is assumed as a model for the voxel intensity probability distribution. However, in most cases, the distribution is far from being Gaussian. Many authors tried to overcome this problem by regularizing the misclassification error through spatially constraining the segmentation process with prior information from a probabilistic atlas [4]. However, the method becomes very sensitive to the correct alignment of the atlas with the image and too time consuming. Actually, doctors do not want to spend a lot of time waiting the result of segmentation, because of the large number of subjects. That induces the need for a fast segmentation algorithm.

Many authors have applied to brain MRI classical segmentation methods, details are given in [3], [6], and [7] to [13]. In order to overcome the problem of these methods, some post-classification methods were proposed [7].

The main contribution of the work we present here is a novel method for brain MRI segmentation based on an information measure, defined in [8], called exponential entropy (EE). The EE information measure solves the different problems related to the use of the classical Shannon entropy, pointed out in [9], i.e. Shannon's entropic description is not defined for distributions that include probabilities of 0. To avoid the problem of the spatial distribution, we defined a two-dimensional histogram, that takes into account the pixel spatial distribution. We also extended the EE to the two-dimensional and multilevel case.

As the computation complexity of the problem at hand exponentially increases with the increase of the number of classes, a fast optimization metaheuristic is needed to search for the optimal solution. Most metaheuristics have the drawback of having parameters which must be set by the user. According to the values given to these parameters, the algorithm is more or less efficient. However, there are many applications for which the user of the algorithm has no time to waste with parameter tuning. Practically, if the values of the objective function result from an experimental time costly process, it would be not possible to lead tests on the values of parameters, particularly in industrial applications. Tuning the parameters requires a minimum of experience about the used algorithm, so, it would be difficult and time consuming for a novice user to find the optimal set of parameters.

In this paper, we propose to use a parameter free PSO algorithm, called TRIBES, that does not need any parameter fitting [14]. Many authors tried to make the PSO algorithm free of parameters [15], [16] and [17]. But the first really parameter free algorithm, called TRIBES, was proposed by Clerc [14].

This paper is outlined as follows: in the next section, the computation of the two-dimensional histogram is presented. In section 3, definition of the exponential entropy is given and the extension of the exponential entropy to the two-dimensional case is presented. A quick description of the TRIBES parameter free Particle Swarm Optimization algorithm is given in section 4. The proposed segmentation algorithm is presented in section 5. Experimental results are discussed in section 6. Finally, we conclude in the last section.

2 Two-Dimensional Histogram

The two-dimensional (2D) histogram [18] of a given image is computed as follows. One calculates the average gray-level value of the neighborhood of each pixel. Let $w(x, y)$ be the averaged image of $f(x, y)$ using a window of size 3x3 defined by:

$$w(x,y) = \left\lfloor \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x+i, y+j) \right\rfloor . \tag{1}$$

where $\lfloor x \rfloor$ denotes the integer part of the number x. In order to solve the frontier problem we disregard the top and bottom rows and the left and right columns. Then the 2D histogram is constructed using expression (2).

$$h(i,j) = \text{Cardinal}(f(x,y) = i \text{ and } w(x,y) = j) / \text{image size} . \tag{2}$$

The joint probability is given by:

$$p_{ij} = h(i,j) , \tag{3}$$

where $i, j \in \{0,1,2,\ldots,255\}$.

The 2D histogram plane is represented in figure 1: the first and the second quadrant denote the background and the objects respectively, the third and the fourth quadrant contain information about noise and edges alone, they are not considered here. A threshold vector is (s, t), where s, for $g(x, y)$, represents the threshold of the average gray-level of the pixel neighborhoods and t, for $f(x, y)$, represents the threshold of the gray level of the pixel. The quadrants containing the background and the objects (first and second) are considered to be independent probability distributions; values in each case must be normalized in order to have a total probability equal to 1. In the case of image segmentation into N classes, a posteriori class probabilities are given by:

$$P_{m-1}\left[a_{n-1}, a_n\right] = \sum_{i=s_{n-1}}^{s_n - 1} \sum_{j=t_{n-1}}^{t_n - 1} p_{ij} \tag{.4}$$

$$P_m\left[a_n, a_{n+1}\right] = \sum_{i=s_n}^{s_{n+1} - 1} \sum_{j=t_n}^{t_{n+1} - 1} p_{ij} . \tag{5}$$

where $a_n \equiv (s_n, t_n)$; n=1,…,N; m=2,…,N and N is the number of classes.

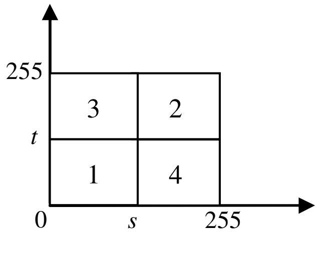


Fig. 1. Two-dimensional histogram plane, where s and t are the thresholds for $w(x,y)$ and $f(x,y)$, respectively

3 Two-Dimensional Exponential Entropy

We define the 2D exponential entropy (2DEE) by:

$$H_\alpha = \left(\sum_i \sum_j p_{ij}^\alpha \right)^{1/(1-\alpha)} , \tag{6}$$

where $\alpha \in \Re$ and $\alpha \neq 1$.

Thus the exponential entropies associated with different image classes' distributions are defined below:

- The 2DEE of the class *m*-1 can be computed through:

$$H_\alpha^{(m-1)}\left[a_{n-1}, a_n\right] = \left(\sum_{i=s_{n-1}}^{s_n - 1} \sum_{j=t_{n-1}}^{t_n - 1} \left(\frac{p_{ij}}{P_m\left[a_{n-1}, a_n\right]} \right)^\alpha \right)^{1/(1-\alpha)} . \tag{7}$$

- The 2DEE of the class *m* can be computed through:

$$H_\alpha^{(m)}\left[a_n, a_{n+1}\right] = \left(\sum_{i=s_n}^{s_{n+1}-1} \sum_{j=t_n}^{t_{n+1}-1} \left(\frac{p_{ij}}{P_m\left[a_n, a_{n+1}\right]} \right)^\alpha \right)^{1/(1-\alpha)} . \tag{8}$$

For the convenience of illustration, two vectors $(s_0, t_0) = (0, 0)$ and $(s_N, t_N) = (255, 255)$ were added, where $t_0 < t_1 < t_2 < \ldots < t_N$ and $s_0 < s_1 < s_2 < \ldots < s_N$.

Then the total 2DEE is:

$$H_\alpha^T\left[a_0, \ldots, a_N\right] = \sum_{i=0}^{N-1} H_\alpha^{(i+1)}\left[a_i, a_{i+1}\right] \tag{9}$$

According to the maximum entropy principle, the optimal vectors $(a_{1,\ldots,N-1}^*) \equiv \left((s_1^*, t_1^*), \ldots, (s_{N-1}^*, t_{N-1}^*)\right)$ should meet:

$$H_\alpha^T\left(a_{1,\ldots,N-1}^*\right) = \max\left\{H_\alpha^T(a_{1,\ldots,N-1})\right\} \tag{10}$$

where: $0 < s_1 < s_2 < \ldots. < s_{N-1} < 255$ and $0 < t_1 < t_2 < \ldots. < t_{N-1} < 255$.

In the case of one threshold (*N*=2) the computational complexity for determining the optimal vector (s^*, t^*) is $O(L^4)$, where *L* is the total number of gray-levels (usually

256). However, it is too time-consuming in the case of multilevel thresholding. For the n-thresholding problem, it requires $O(L^{2n+2})$. In this paper, we further present a parameter free PSO algorithm for solving $\arg\max\left\{H_{\alpha}^{T}\left[(s_1,t_1),(s_2,t_2),...,(s_{N-1},t_{N-1})\right]\right\}$ efficiently.

4 Parameter Free PSO Algorithm (TRIBES)

The Particle Swarm Optimization (PSO) is a population based stochastic technique developed by Kennedy and Eberhart (1995). PSO has similarities with the genetic algorithms: a population of potential solutions is used in the search. However there is no evolution operator in PSO. The technique starts with a random initialization of a swarm of particles in the search space. Each particle is modeled by its position in the search space and its velocity. At each time step, all particles adjust their positions and velocities, thus their trajectories, according to their best locations and the location of the best particle of the swarm, in the global version of the algorithm, or of the neighbors, in the local version. Here appears the social behavior of the particles. Indeed, each individual is influenced not only by its own experience but also by the experience of other particles.

TRIBES is an adaptive algorithm of which parameters change according to the swarm behavior. In TRIBES, the user only has to define the objective function and the stopping criterion. The method incorporates rules defining how the structure of the swarm must be modified and also how a given particle must behave, according to the information gradually collected during the optimization process.

However, it must be pointed out that TRIBES, like all competing optimization algorithms, cannot solve with certainty all the problems. Moreover, TRIBES is a stochastic algorithm, thus results given by the algorithm are probabilistic. The aim of TRIBES is to be an algorithm which is efficient enough in most cases and which permits to the users to gain time by avoiding the fitting of parameters.

4.1 Swarm's Structure and Communication

The swarm is structured in different "tribes" of variable size. The space search is simultaneously explored and all tribes exchange results in order to find the global optimum. The algorithm includes two different types of communication: intra-tribe communication and inter-tribes communication, more details about these types of communication are given in [14].

To set rules to modify the swarm's structure, quality qualifiers are defined for each particle and likewise for the tribes. These qualifiers allow defining two rules: removal of a particle and generation of particles. These structural adaptations are not done at all iterations. In practice, if *NL* is information links number at the moment of the last adaptation, the next adaptation will occur after *NL*/2 iterations. For more details see [14].

4.2 Swarm Evolution

The swarm is initialized by only one particle, that represents a single tribe. A second tribe is created if, at the first iteration, the initial particle does not improve its location. The same process is then applied for the next iterations. The size of the swarm increases until promising areas are found. In other words, the capacity of the swarm to explore increases, but the time between successive adaptations decreases. Then, the swarm has more and more chances to find a good solution between two adaptations. This can be seen as a strategy of displacement. Other implemented strategies are described below.

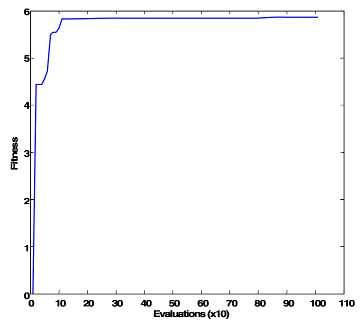
4.3 Strategies of Displacement

The second strategy to adapt the swarm to the found results is by selecting a different strategy of displacement of each particle according to its recent past. Then the algorithm chooses to call for the best strategy of displacement in order to move the particle to the best possible location, that can be reached.

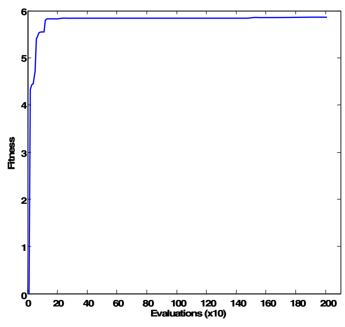
TRIBES tries to overcome an important problem of metaheuristics: the fitting of parameters. TRIBES frees users of defining parameters by adapting the structure of the swarm and the strategies of displacement of the particles. The particles use their own history and the history of the swarm to decide the way of their move and the organization of the swarm in view of approaching as efficiently as possible the global optimum. Fig. 2 shows a summary of TRIBES process.

```
1. Initialization of a population of particles with random positions and
      velocities.
2. Evaluate the objective function for each particle and compute $g$.
For each individual i, $p_i$ is initialized at $X_i$.
3. Repeat until the stopping criterion is met
      3.1.  Determination of status of all particles
      3.2.  Choice of the displacement strategies
      3.3.  Update the velocities and the positions of the particles.
      3.4.  Evaluate the objective function $H_\alpha^T\left[a_0,...,a_N\right]$ for each
            individual.
      3.5.  Compute the new $p_i$ and $g$.
              If $n<NL$
                    - Determination of tribes qualities
                    - Swarm's adaptations
                    - Computation of $NL$
              End if
4. Show the best solution.
```

Fig. 2. Principle of TRIBES, where g is the best location reached by the swarm, p_i is the best location for particle *I*, X_i the position vector of the particle *i*, *NL* is the number of information links at the last structure of the swarm, and *n* is the number of iterations since the last adaptation of the swarm.



(a)



(b)

Fig. 3. Example of the evolution of the fitness function in logarithmic scale (for image of Fig. 4 (a)) for : (a) 2000 evaluations, (b) 1000 evaluations. The curves are the result of the averaging of 25 runs.

5 The Proposed Image Segmentation Algorithm

The proposed image segmentation algorithm is based on the maximization of the total 2DEE using TRIBES. The method exploits the particle swarm approach to solve the segmentation problem expressed by (10). The algorithm does not require any special initialization. The number of evaluations was used as stopping criterion. Looking at our experiments (Fig. 3), the value of the fitness function does not increase significantly after 1000 evaluations of the objective function, that explains our decision to fix the maximum number of evaluations of the objective function at 1000.

6 Experimental Results and Discussion

In this section, we discuss the selection of the optimal thresholds and the presentation of some MR images. The performances of the method are compared to those of five other methods, over the segmentation of a synthetic images. The results on MRI segmentation were compared to those provided by the 2D Shannon entropy (2DSE) method [11]. Here, are presented only the results in the case of four and five classes' segmentation.

The value of the optimal threshold depends on the 2DEE order (α). In order to find the optimal value (α^*), the well known uniformity criterion is used. This criterion is given by:

$$U_{(\alpha)} = \left(\frac{1-2N}{M}\right) \sum_{j=0}^{N} \sum_{i \in C_j} (f_i - \mu_i)^2 \Big/ (f_{\max} - f_{\min})^2 \ . \tag{11}$$

where N is the number of thresholds, C_j the jth class, M the number of pixels in the image, f_i the gray level of pixel i, μ_i the mean gray level of pixels in jth class, f_{max} and f_{min} the maximum and the minimum gray levels of pixels in the image, respectively. U has a positive value and lies between 0 and 1. When U is close to 1, the uniformity is very good and vice versa.

6.1 Comparison to Other Methods

We compared the performance of the proposed method to those of five other methods: EM algorithm based method (EM) [20], one method based on valley-emphasis (VE) [21], the well known Otsu method [9], the classical Kapur *et al.* method [9], and Sahoo *et al.* method based on 2D Tsallis entropy (TE) [22]. The comparison is based on synthetic images, noised with different degrees of noise (Fig. 3). To measure these performances, the misclassification error (*ME*) criterion was used [9]. *ME* is defined in terms of correlation of the images with human observation. *ME* is expressed by:

$$ME(\%) = \left(1 - \frac{|B_O \cap B_T| + |F_O \cap F_T|}{|B_O| + |F_O|}\right) \times 100 \qquad (12)$$

where background and foreground are denoted by B_O and F_O for the original image, and by B_T and F_T for the thresholded image, respectively. In the best case of ideal thresholding, *ME* is equal to 0% and, in the worst case, *ME* value is 100%.

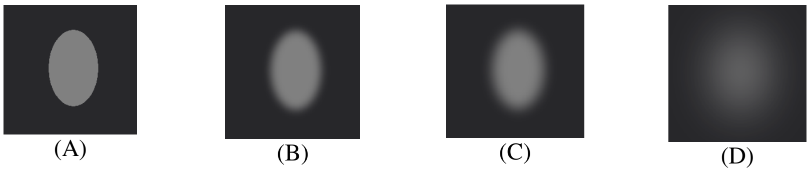


Fig. 4. (A) Original synthetic image, (B) to (D) noised images

Table 1. Performance evaluation of the proposed method compared to competing methods

Test images of Fig. 4	Segmentation methods						
	Otsu	Kapur	EM	VE	TE	EE2D	
	ME(%)	*ME(%)*	*ME(%)*	*ME(%)*	*ME(%)*	*ME(%)*	α
Image B	0.45	5.61	8.68	0.34	0.64	**0.23**	0.4
Image C	0.88	4.50	12.50	0.63	1.12	**0.56**	0.4
Image D	12.22	4.97	28.87	11.59	12.90	**3.57**	0.6

The quantitative comparison of the results provided by our method and the five other methods, based on segmentation of synthetic images, is presented on table 1. As it can be seen, the proposed method provides better results than the other methods, only VE method provides a better performance in the case of image B.

Table 2. Experimental results for image in Fig. 4 (a)

Number of classes (*N*)	Time (s)	Speed gain factor
3	14.8	$106.\,10^4$
4	19.6	687.10^8
5	26.7	194.10^{16}

6.2 Examples of Results and Discussion

The obtained results through the application of our segmentation algorithm are illustrated with two brain MRI. Fig. 5 shows the original images and their multilevel classification (segmented) version when N=4 and 5. The results in the case of a sane subject are in Fig. 5 (c) and (e); those in the case of an atrophy pathology are shown in Fig. 5 (d) and (f). Our goal is to detect the different spaces and the white matter surrounding the ventricular space quickly. In order to quantify the performance of the optimization algorithm, we define the speed gain factor, that corresponds to the ratio of the number of the exhaustive search solutions to the evaluation number of the objective function.

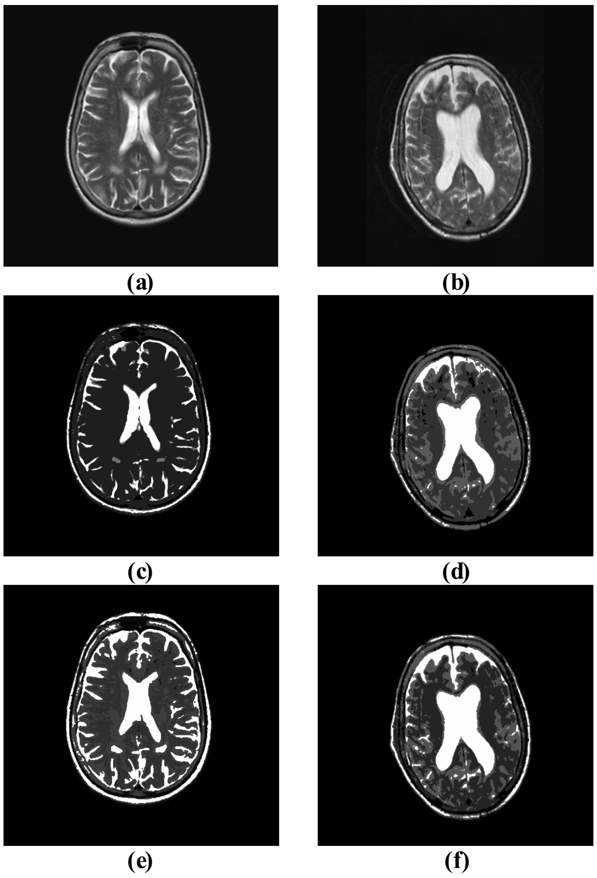


Fig. 5. Segmentation of sane and pathologic MRI. (a) Original image of a sane brain, (b) Original image of a pathologic brain, (c) 4 classes segmented image T=(30, 112, 134) with α=0.1, where T is the threshold vector, (d) 4 classes segmented image T=(49, 88, 180) with α=0.3, (e) 5 classes segmented image T=(10, 47, 61, 97) with α=0.4, (f) 5 classes segmented image T=(36, 82, 138, 153) with α= 0.2.

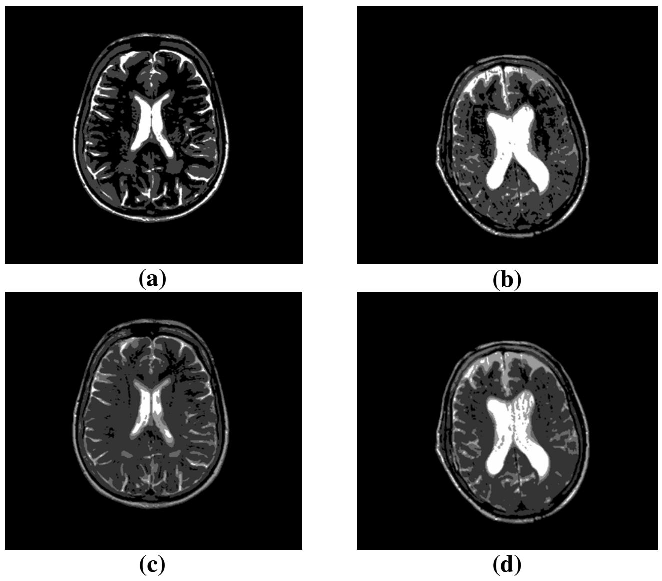


Fig. 6. 2DSE segmentation results. (a) 4 classes segmented image *T*(64, 128, 191), where *T* is the threshold vector, (b) 4 classes segmented image *T*(65, 131, 192), (c) 5 classes segmented image *T*(52, 102, 152, 203), (d) 5 classes segmented image *T*(52, 103, 155, 205).

The number of points for which the criterion function must be evaluated, in the case of an exhaustive search, is $\left(L!/\left((L+1-N)!(N-1)!\right)\right)^2$, where L is the total number of gray-levels (usually 256). For instance, when L=256 and N=2, the number of objective function evaluations is 65536 and, when L= 256 and N=3, it is 32640^2! [14]. Table 1 shows the experimental results obtained on the image of Fig. 5 (a). The speed gain factor and the time values show effectiveness of TRIBES algorithm and confirm that our method is fast compared to those in [1] to [7], where the result is obtained after more than 120s [7]. As it can be seen, in table 2, the speed gain factor increases by a factor higher than 10^4 when one class is added to the problem.

Fig. 6 shows the results obtained via the application of 2DSE. One notices that the results provided by our method are more homogeneous than those provided by 2DSE. This can be seen clearly, for instance, through the comparison of the detected white matter, between Fig. 5 (f) and Fig. 6 (d).

7 Conclusion

In this paper, we proposed a new fast approach to find the optimal thresholds, based on 2DEE to avoid the problems related to the use of Shannon entropy to segment images. We also proposed to use a parameter free PSO algorithm and our experiments proved that TRIBES can be used as a black box optimization tool to solve a segmentation problem. The use of TRIBES allows to avoid the parameter tuning step that requires a minimum of experience about the used algorithm.

It is clearly seen from the experimental results that the presented method is more efficient than the classical 2DSE and using TRIBES allows to obtain good results quickly. However, the use of the method to segment other kinds of images does not provide good segmentation results when the images are strongly noised. In the work in progress we use a multiobjective optimization based on parameter free PSO in order to add information to segment noised images.

Acknowledgements. The authors would like to thank Dr. Raphael Blanc, from Neurosurgery service of “Centre Hospitalier Universitaire de Mondor”, Créteil, France, for supplying the MR images.

References

1. Drapaca, C.S., Cardenas, V., Studholme, C.: Segmentation of tissue boundary evolution from brain MR image sequences using multi-phase level sets. Computer Vision and Image Understanding 100, 312–329 (2005)
2. Qiao, Y., Hu, Q., Qian, G., Luo, S., Nowinski, W.L.: Thresholding based on variance and intensity contrast. Pattern Recognition 40, 596–608 (2007)
3. Warfield, S.K., Kaus, M., Jolesz, F.A., Kikinis, R.: Adaptive template moderate spatially varying statistical classification. Medical Image analysis 4, 43–55 (2000)
4. Murgasova, M., Dyet, L., Edwards, D., Rutherford, M., Hajnal, J.V., Rueckert, D.: Segmentation of Brain MRI in Young Children. In: 9th Int. Conf. MICCAI Copenhagen, pp. 687–694 (2006)
5. Song, Z., Tustison, N., Avants, B., Gee, J.C.: Integrated Graph Cuts for Brain MRI Segmentation. In: 9th Int. Conf. MICCAI, Copenhagen, Denmark, pp. 831–838 (2006)
6. Kamber, M., Shinghal, R., Collins, D.L., Francis, G.S., Evans, A.C.: Model-based segmentation of multiple sclerosis lesions in magnetic resonance brain images. IEEE Trans. on Med. Imaging 14, 442–453 (2000)
7. Cocosco, C.A., Zijdenbos, A.P., Evans, A.C.: A fully automatic and robust brain MRI Tissue Classification. Medical Image Analysis 7, 513–527 (2003)
8. Zografos, K., Nadarajah, S.: Survival Exponential Entropies. IEEE Trans. on Information Theory 51, 1239–1246 (2005)
9. Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13, 146–165 (2004)
10. Tao, W., Tian, J., Liu, J.: Image segmentation by three level thresholding based on maximum fuzzy entropy and genetic algorithm. Pattern Recognition Letters 24, 3069–3078 (2004)
11. Peng-Yeng, Y.: Multilevel minimum cross entropy threshold selection based on particle swarm optimization. Applied Mathematics and Computation 184, 503–513 (2007)
12. Zahara, E., Fan, S.S., Tsai, D.: Optimal multi-thresholding using a hybrid optimisation approach. Pattern Recognition Letters 26, 1082–1095 (2004)
13. Synder, W., Bilbro, G.: Optimal thresholding: A new approach. Pattern Recognition Letters 11, 803–810 (1990)
14. Clerc, M.: TRIBES - Un exemple d’optimisation par essaim particulaire sans paramètres de contrôle. In: OEP 2003, Paris (2003)
15. Ye, X.F., Zhang, W.J., Yang, Z.L.: Adaptive Particle Swarm Optimization on Individual Level. Int. Conf. on Signal Processing (ICSP), Beijing, China, 1215–1218 (2002)

16. Zhang, W., Liu, Y., Clerc, M.: An adaptive PSO algorithm for real power optimization. In: APSCOM (Advances in Power System Control Operation and Management), S6: Application of Artificial Intelligence Technique (part I), Hong Kong, pp. 302–307 (2003)
17. Yasuda, K., Iwasaki., N.: Adaptive particle swarm optimization using velocity information of swarm. In: IEEE Conference on System, Man and Cybernetics, The Hague, Netherlands, pp. 3475–3481 (2004)
18. Nakib, A., Oulhadj, H., Siarry, P.: Microscopic image segmentation based on two-dimensional exponential entropy with hybrid microcanonical annealing. In: Proceedings of Int. Conf. IAPR- MVA2007, Tokyo, pp. 420–423 (2007)
19. Nakib, A., Oulhadj, H., Siarry, P.: Image histogram thresholding based on multiobjective optimization. Signal processing 87, 2516–2534 (2007)
20. Bazi, Y., Bruzzone, L., Melgani, F.: Image thresholding based on the EM algorithm and the generalized Gaussian distribution. Patter Recognition Journal 40, 619–634 (2007)
21. Ng, H.: Automatic thresholding for defect detection. Pattern Recognition Letters 27, 1644–1649 (2006)
22. Sahoo, K.P., Arora, G.: Image thresholding using two dimensional Tsallis-Havrda-Charvat entropy. Pattern Recognition Letters 27, 520–528 (2006)

Mimetic Variations on Stigmergic Swarm Paintings

Paulo Urbano

Faculdade de Ciências da Universidade de Lisboa
Campo Grande 1749-016 Lisboa, Portugal
pub@di.fc.ul.pt

Abstract. This paper explores artificial collective artistic work inspired by natural phenomena, namely the use of pheromone substances for mass recruitment in ants. Our goal is to look for innovative patterns using techniques derived from Artificial Life. We will play 2 variations, based on imitation, on a society of anonymous and homogeneous artificial micro-painters (the Colombines). In the Colombines model the virtual canvas, besides being a computational space for depositing paint, is also a pheromone medium, mirroring the painting patterns and influencing the painters' behaviour. More, the micro-painters do not exchange information directly with each other, they are simply attracted towards non-painting areas of the canvas—the non-painted "tableaux" patches diffuse an "environmental produced" chemical and the painters prefer to follow the chemical gradient. Thus, this form of stigmergic communication simply influences the artistic agents movements. We will expand the Colombines basic model adding direct communication between the micro-artists: they will imitate the colour of others. In the first variation, they will imitate the colour of who ever they interact with and in the second one they will have a force attribute and colour imitation will depend on the force relationship between them.

1 Introduction

The study of biological self-organization [1] has revealed that numerous sophisticated pattern formation, decision-making, and collective behaviour, are the emergent result of the interaction of very simply behaviours performed by masses of individuals relying only on local information. In particular, successful problem solving by social insects made models of their collective mechanisms particularly attractive [4,5]. The dissemination of Artificial Life has been an important influence in the media arts [18]. Our goal is to explore the artistic possibilities of artificial collective artists that rely on auto-organization, furthering on previous work [17,18].

There are already examples of collective artistic pieces made by natural and real agents. Examples of flocking based artwork include interactive musicians [2] and interactive video installations [3,13]. L. Moura [4] has used a small group of robot-painters inspired by ants' behaviour that move randomly in a limited space. Stimulated by the local perception of the painting they may leave a trace with one of their coloured pens. The painters rely on stigmergic interaction [6,15] in order to create confused patterns with some spots of the same colour. Colour has the pheromone role:

a spot dominated by a certain colour has the capacity to stimulate the painter-robot to add some paint of the same colour. Monmarché et al. [10] have also designed groups of painters inspired by ants' pheromone behaviour and the paintings were evolved using an interactive genetic algorithm. It is based on a competition between ants: the virtual artists try to superimpose their colours on traces made by others, creating a dynamic painting which is never finished. Ants have the capability to "sniff" the painted colour and react appropriately. The group is composed by a small number of individuals (less than 10). Monmarché et al. [10] have also applied their swarm algorithm to music. Greenfield [7] introduced a non-interactive genetic algorithm to evolve swarm paintings. In [17] we developed swarm painters, the Colombines, where the environment is responsible for the production and diffusion of pheromones which guide the movement of the painters. One of the differences from the other ant-paintings is that the artistic individuals are not charged for pheromone production, (the environment is responsible for that task). More, the diffusion process does not occur on any of the ant paintings we have referred. We introduced also populations of numerous agents: we have experimented with groups composed of up to 2000 individuals working in the same artistic piece. Greenfield furthered the Colombines style, introducing a multiple pheromone model [8].

Following the work of Kaplan [9] and Shoham and Tennenholtz [14] on convention emergence in multiagent systems, in [18] we have applied a collective mechanism of emergence of random convention sequences, to the generation of collective paintings. We introduced the Gaugants, a society of micro-painters where consensus (collective choice) around some attributes (colour and orientation) is the source of artistic pattern and where the continuously changing collective choices are the source of diversity and non-homogeneity. Both the Colombines and Gaugants were implemented in Netlogo, a derivation of Starlogo [12].

We will play 2 variations, based on imitation, on the Colombines model referred before. We will expand the Colombines model, where stigmergic communication simply influences the artistic agents movements, adding direct communication between the micro-artists: they will imitate the colour of others. In the first variation, they will imitate the colour of who ever they interact with and in the second one they will have a force attribute and colour imitation will depend on the force relationship between them. These variations allow us to explore the possibilities of pattern formation in swarm societies.

The remainder of the paper is organized as follows: In section 2 we remember the Colombines painters; in section 3 we introduce the first mimetic variation on the basic model: besides moving towards the unpainted parts of the canvas they imitate the colour of their neighbours which are captured inside a varying perception radius. In section 4 we describe the Force-Mimetic Colombines and their artistic pieces. In the final section we conclude.

2 The Colombines

The Colombines are a swarm of small and homogeneous artificial micro-painters, individually very simple, which are able to paint a bi-dimensional virtual canvas, composed of small cells.

The canvas is bi-dimensional space with a toroidal format, divided in small squared sections, called patches or cells, it is a kind of grided paper, with no borders, folded in every direction, in which two types of virtual materials coexist: paint and a chemical signal. Each patch can have a certain colour and can have a certain quantity of chemical. There is a fixed colour (usually grey) for the background. Any other colour corresponds to paint.

The non-painted cells have more attraction power (more chemical). Therefore, every cell has the potential ability to release chemical, but only the non-painted cells (the background ones) are chemical producers. The squared canvas is a kind of chemical medium where every cell is permanently diffusing chemical to their immediate neighbours, independently of being painted or not. The chemical evaporates at a constant rate. Without evaporation, the attraction power decay of recently painted spots will last more time, disorientating the painters, attracting them to painted spots. Foremost, the evaporation phenomenon increases the painters' efficiency: the painting will be completed sooner.

The cells behaviour is the following: 1) if it is not painted increase its own chemical quantity by a certain amount, otherwise the chemical level is maintained intact; 2) diffuses a percentage of its chemical to their 8 immediate neighbours; 3) delete a percentage of its chemical (evaporation). The chemical constant produced by non-painted cells, the evaporation and diffusion taxes are parameters modifiable by the user.

Initially, we launch these painters in a non-painted background, each one occupying a particular cell, and they will move along, depositing a trace of ink, until the canvas is completely fulfilled. Note that each painter is constrained to paint only non-painted cells and when there isn't any non-painted cell left, the artistic work cannot change and is considered finished.

Our micro-painters have a very limited perception field—they have an orientation and have access just to the three cells in front of them. Each painter is created with a particular colour and they never change to another colour. It's the empty spots that guide the painters. They prefer to move towards empty spots.

If each Lilliputian painter just acted on its own, without any interactions, either with the world or with the others, interesting phenomena would never arise. They do no more than moving on the virtual canvas, visiting preferentially cells with more amount of chemical, (preferring to move towards non-painted spots) and painting cells still unpainted, leaving traces of colour behind them. In case of identical chemical values in their neighbouring cells they have a tendency to preserve its current direction. Each Colombine has a position (real Cartesian coordinates), an orientation (0..360), and can only inhabit one cell, the one that corresponds to their coordinates. They see just their own cell and also the three cells immediately in front of them. On the other hand, the painters are created with a particular colour that is never going to be changed.

The behaviour of each Colombine is the following: 1) it senses the three immediate cells in front of him and chooses the one with more chemical, changing his orientation towards that winning cell and moving to it; 2) if that cell is not yet painted, stamps his colour on it, otherwise, does not paint it. In detail, the painter senses his three forward neighbouring cells and if there is no better patch than the one in front he remains with the same orientation and go forward one step (rounding his coordinates). If the left path is the most attractive he rotates 45 degrees to the left and moves forward one unity, rounding both position coordinates; the same happens when he prefers the right cell: he rotates to the right 45 degrees, moving forward one unity, rounding his coordinates. The round operation influences the patterns generated, as we will see later.

The evolution of the collective artistic work happens in the following way. Initially, the virtual canvas is grey and each patch has an identical quantity of chemical (normally 0). We create a colony of Colombines, each one with its own colour and orientation, distributing them in the environment. The painting process will begin in a sequence of iterations until every patch is painted completing the plastic work. Each iteration is divided in two steps: in the first, every cell executes its behaviour (chemical production, diffusion and evaporation); in the second step, the Colombines move, attracted by chemical, depositing paint. The paintings are only declared finished when there are no grey patches, but, alternatively, we could finish the collective work after a random or fixed number of iterations.

2.1 Dynamics Responsible for Pattern Emergence

The canvas can be seen as a dynamical chemical landscape, in permanent mutation—there is a constant interaction between chemical distribution and the painters' behaviour. The chemical world is information floating both in the painted and background patches. There is a strong circularity: On one hand, the chemical information guides the movement of the Colombines, attracting them toward non-painted spots, On the other hand, their painting activity change the information landscape, in an permanent auto-catalytic interaction. The patterns, the coloured forms, are the by-product of the collaboration between the Colombines and their chemical environment. Figure1 illustrates how Colombines pattern emerges.

We have two painters, one white and one black. They have an initial orientation (black moves east and white goes south). They both tend to preserve their directions. The black suddenly changes direction, avoiding the trace left by the white painter. After a while the white painter reaches his own trace and avoids it, changing direction and having to avoid later the black trace and the painting progresses. Sometimes, the painters have to cross already painted spots, due to the fact that the three patches ahead are already painted and they cannot escape them.

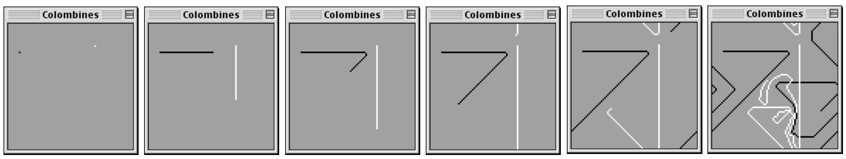


Fig. 1. The interaction between two painters. Illustration of the tendency to conserve direction and to avoid painted patches.

In figure 2 we show a painting progress taken in 4 different instants, made by 300 micro-painters. Initially the Colombines are scattered randomly on the "tableaux", starting with a colour randomly chosen from a list of 140. Notice that we can find spots with the same colour due to the fact that a painter can be on a non-painted area which is surrounded by traces, constraining him to be inside, painting that enclosed

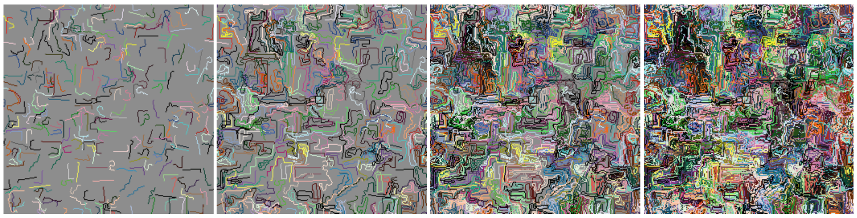


Fig. 2. The progress of a painting made by 300 Colombines. Each one has a colour chosen randomly from a 140 possibilities.

spot. Population size is important because as density increases the possibility of encountering very soon traces of other painters also increases and constrains immediately our micro-artist that begin to fold their patterns, creating smaller spots with the same colour as we are going to see. In this painting the "tableaux" has a dimension 131*131 patches.

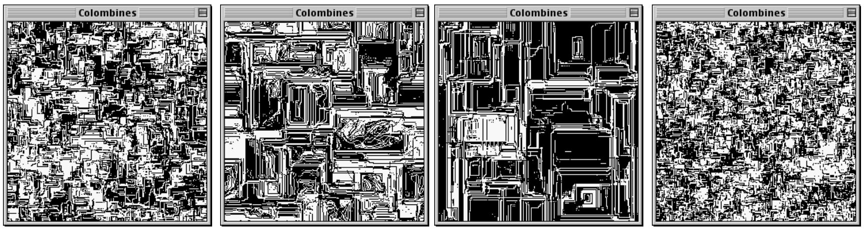


Fig. 3. Four Colombine black and white paintings. From left to right: Alfama of Glass, Opening the Head of Pacheco Pereira, Coimbra of Xanana and In the Roof of Hugo Pratt.

In figure 3 we show four examples of finished paintings made by societies of Colombines of different sizes (from left to right, 1000, 100, 50 and 2000 painters) in a world of 125*125 patches. There are only black and white painters equitably distributed by each of the colours and which are randomly scattered on the "tableau". The painters were created with random orientations. If we increase the number of micro-painters, the possibility of encountering traces also increases. The resulting effect is that the spots with the same colour have a smaller area and we find less rectilinear traces.

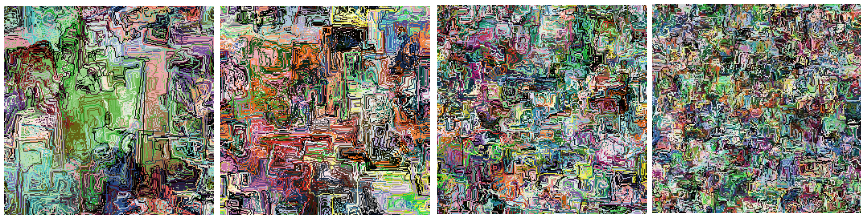


Fig. 4. Four Colombine coloured paintings. From left to right: 50, 100, 500 and 1000 painters with a random initial colour.

In figure 4 we vary population size and show some coloured paintings, where initially each painter has a random colour chosen from 140 possibilities (0..139). We can conclude that there is a Colombines style made by painters trying to avoid painting spots even if they never do not change their colours.

3 First Variation: Simple Mimetic Colombines

We will now introduce a variation in the Colombines behaviour. Now they will interact directly with each other exchanging information. They will imitate the colour of who ever they happen to meet.

3.1 Simple Imitation in Convention Emergence

We will first describe the simple imitation behaviour in the context of convention emergence developed in lexical [9] and social rules [14] formation research. Suppose we have a number of agents which are able to make their own decisions, and they can interact with each other, influencing and being influenced by others. With time, a winning option can eventually emerge and a consensus is therefore attained. The main goal is to reach a global consensual choice through decentralised mechanisms based on self-organisation. In every model each agent has only local access to the society, which is composed of anonymous agents.

In the research of convention emergence, initial situations correspond to situations of maximal competition. In the literature, two initial situations of maximal competition are considered. In the first one we have only two possible choices for being the convention adopted, where each one is initially adopted by 50% of the population, and in the second one we have a different initial choice per individual.

The interaction is based on a series of dialogues involving a pair of agents. In each dialogue, two of the society members are randomly chosen for interaction, the hearing and speaking elements. In what concerns performance analysis, we are interested especially in the average convergence velocity for several simulations and its variation with the number of agents. The convergence velocity is the number of dialogues necessary for reaching a global consensus, starting with options that are equally distributed among agents—no option dominates in the initial population.

In the imitation game, during a unilateral dialogue, the speaking agent indicates to the hearing agent the option it is currently using, and the latter adopts it immediately. Starting with 2 or N options equally distributed in the population (N agents), nothing directs the group towards convergence, as every option can increase its influence with equal probability. In general, convergence is assured after an important series of oscillations in a time quadratic with the number of agents.

3.2 Simple Mimetic Colombines

In convention emergence we want behaviours with fast convergence and the simple imitation behaviour surely does not fit this goal. But it is perhaps well suited for our artistic purposes. If convergence on some attribute, colour for example, is too fast, it means that soon we will have only one colour in the canvas, even starting from a situation of maximal divergence. A slow convergence means we will have diversity and full homogeneity will arrive late.

This way we will change the Colombines introducing a vision radius, defining a perception area on each micro-painter. They continue to sense their own patch and the three patches ahead for chemical but they can sense other painters inside the vision radius. In each painting step, besides following the gradient and depositing paint, they will pick a random neighbour (someone inside the perception area) and imitate unconditionally its colour.

The convergence time, towards a consensual situation on colour will increase with the number of painters involved and it will increase with shorter perception radius. Note also that movement depends on chemical production and diffusion, and they depend on the painted spots and movement conditions painters' positions and theirs neighbourhoods. Everything is related and dependent.

In figure 5 we show three snapshots describing the evolution of a painting made by 50 painters with a vision radius of 15. Initially there are 25 pink painters and 25 red ones. The pink colour wins the convergence game and the painting result is a red pattern on a pink background. Remember that the non-painted background colour was grey. Therefore, the resulting pattern will be the result of a balance between the number of painters and the vision-radius. We will try to show in the following figures, the paintings that can result from the tuning of those 2 parameters.

In figure 6 we show 3 paintings for different population and radius-vision sizes. All this populations have mimetic-stigmergic painters of 2 initial colours (blue and red) and we begin with 50% reds and 50% blues.

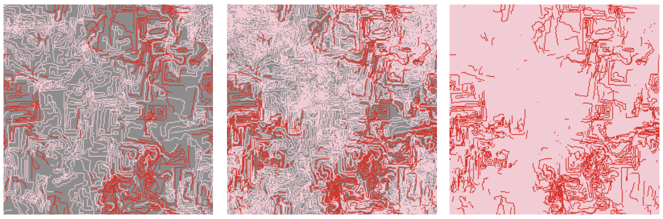


Fig. 5. Evolution of a painting with 50 painters (two colours (red and pink) equally distributed, 25 pinks and 25 reds. The vision radius is 15 units.

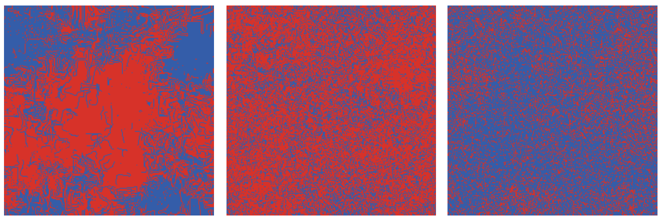


Fig. 6. 3 paintings with 100, 500 and 1000 painters with 15, 30 and 30 of vision-radius respectively. Only 2 colours are competing, blue and red.

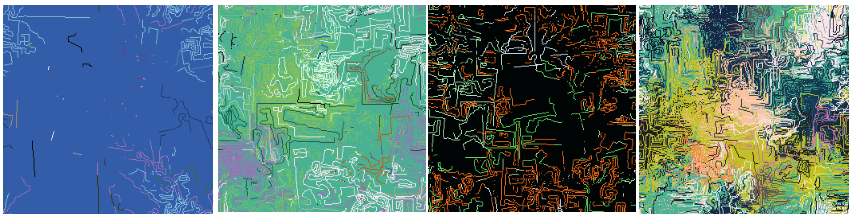


Fig. 7. The first painting made with 50 painters with a random initial colour and a vision radius of 10. The other 3 paintings are made by a group of 100 painters where we vary the vision-radius, respectively (20, 30 and 10).

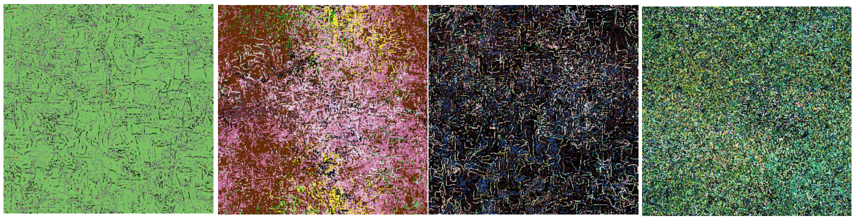


Fig. 8. Four mimetic-estigmergic multicoloured paintings made from groups of 200, 500, 500 and 1000 micropainters with vison-radius of 50, 20, 30 and30.

In figure 7 and 8 we show 4+4 paintings for different 140-coloured populations and different vision-radius. We know that in the simple imitation behaviour, after some time, two colours begin to dominate and finally one of them wins, but eventually there is not enough time to achieve convergence and also it can happen that when convergence is achieved there are not many non-painted spots left. Therefore, we can find 2 or 3 dominant colours. In conclusion, with this mimetic variation on stigmergic painters we have arrived to create new patterns that were not possible with non mimetic painters.

4 Second Variation: Imitation Based on Force: Consensual Evolution

In the context of convention emergence research we have designed a very successful behaviour in what concerns speed of convergence and its variation with the number of agents involved [17]. We have introduced force as a new attribute of agents, besides choice. The force attribute transforms each dialogue in a conflict interaction. The main point of conflict interactions is that agents only imitate other agents as strong or stronger than them.

4.1 Double Imitation of Stronger Agents with Reinforcement

In this behaviour, we will have imitation of option and of force. During unilateral encounters, we have again one speaking and one hearing agents. The speaker will tell the other how much force it has and what is its option. The behaviour is divided in two parts, in the first one force is compared and there will be imitation eventually; in the following part, there is positive reinforcement.

Part 1: In case the hearing agent looses (it has less or the same force of its interlocutor) it will imitate both the force and colour of the speaking agent. If the speaking agent is weaker than the hearing agent, this one will conserve both force and option. This way, the new recruited agents will have their force increased and will be more powerful recruiters than they were before the dialogue started. Agents with more power impose their colours in a contagious way.

Part 2: if they had the same choice or option when they met, the hearing agent will now reinforce its force in 1 unit, independently of loosing or winning the interaction in part 1.

In sum, the stronger ones recruit weaker agents (these will be at least as strong as the winners imitating their choices, and they can even overpass them in case their options were the same) enlarging the influence of their options.
This behaviour manifests fast convergence, even faster than a known behaviour well studied in convention emergence research (positive reinforcement with score) [9, 14], which has a complexity of nlogn, where n is the number of agents.

So, we have a very effective behaviour for consensual choice formation, which is well-suited for decentralized convention formation, unlike the simple imitation, but let's try to apply it to artistic creation, and specifically to swarm painting.

4.2 ForceMimeticColombines

We will now apply this successful behaviour, concerning convergence velocity, to our micro-painters. What we called option will be the colour attribute. So, initially we distribute the ForceMimeticColombines randomly on the "tableaux" assigning them random initial colours and 0 of force. Thus, each time a painter sees another (depending on the vision-radius) it compares its force with its interlocutor. In case, it is weaker or it has the same force, it will imitate both the stronger force and colour; if it has more force, no mimetism. After there is the reinforcement phase—if their colours were the same when they met the "hearing" painter will reinforce its strength in one unit.

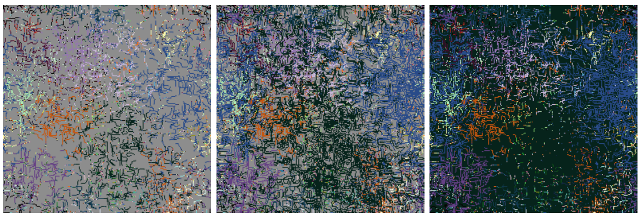


Fig. 9. Evolution of a painting with 1000 painters (random chosen from 140 possibilities). The vision radius is 20 units.

The difference towards the simple mimetic painters is on the speed of collectively choosing a colour. The consensual colour will be found earlier with the ForceMimeticColombines, which is perhaps not very desirable for artistic societies composed of a small number of agents, unless they have a small vision-radius. Otherwise, one of

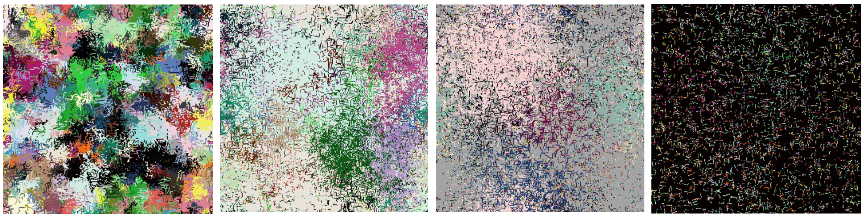


Fig. 10. Four paintings made with 2000 painters with different vision-radius (from left to right, 5, 15, 30, 50

the colour dominates very early and once it dominates the whole population nothing will change until the end, which means a lot of homogeneous pattern—a super minimalism; a big background colour and some Colombine style traces on it,

In figure 10 we show images made by societies of numerous painters (2000) where colour convergence varies with the vison-radius attribute. We can compare these pieces with simple mimetic ones made by 1000 or 2000 thousand painters and the difference is obvious. Those exposed in figure 10 have much more spots with the same colour due to the fact that imitation is more effective. We can turn the knobs of population size and vision-radius in order to obtain a wide variation on pattern besides the fact that the collective choices on colour are different from simulation to simulation.

We finish this section comparing this work with previous work with consensual behaviour, the Gaugants model [18]. In the Gaugants model there was no stigmergic interactions and there was an imitation behaviour of colour and orientation. As a collective choice was obtained very fast we had to create the figure of a dissident and contagious painter that after seeing enough clones of itself decides to change its colour and orientation attributes and increase its force, imposing its new attributes (in a very contagious way) to its painter colleagues.

5 Conclusions and Future Work

We have mixed two coordinating models based on auto-organization in order to generate collective artificial paintings. The first model is based on stigmergy (Colombines) and the second one is based on mimetic behaviours derived for convention emergence (Gaugants). The movement of painters is controlled by the stigmergic interaction, a chemical produced by non-painted "tableaux" cells) attracts painters, and the colour used by agents depends on mimetic interactions. We have applied two mimetic variations on the Colombine models, simple imitation which is not very effective in what concerns fast convergence towards collective choice on colour and an algorithm developed by the author [16] where there is a recruitment based on the force of agents, which is very effective. These 2 variations expand the stigmergic model creating new patterns, generating innovative swarm paintings.

We are going to continue exploring new ways of coordinating swarms and of generating artificial art.

If you have problems seeing the images you can consult www.di.fc.ul.pt/~ pub/ Evo07 where you can generate images and see images at their real scale.

References

[1] Bonabeau, E., Dorigo, M., Theraulaz, Z.: Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute, Studies in the Sciences of Complexity (1999)

[2] Blackwell, T.M.: Swarm Music: Improvised Music with Multi-Swarms. Artificial Intelligence and the Simulation of Behaviour, University of Wales (2003)

[3] Boyd, J.E., Hushlak, G., Jacob, C.J.: Swarm Art: interactive art from swarm intelli-gence. In: Proceedings of the 12th annual ACM International Conference on Multimedia, NY USA (2004)

[4] Camazine, S., Deneubourg, J.-L., Franks, N., Sneyd, J., Theraulaz, Z., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2001)

[5] Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans. Syst. Man. Cybern. B 26, 29–41 (1996)

[6] Grassé, P.-P.: Termitologia, Tome II." Fondation des Sociétés. Construction, Paris, Masson (1984)

[7] Greenfield, G.: Evolutionary methods for ant colony paintings. In: Proceedings of the 7th Conference on Short and Medium Span Bridges, Montréal, Québec, Canada (2006)

[8] Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.): EvoWorkshops 2005. LNCS, vol. 3449. Springer, Heidelberg (2005)

[9] Kaplan, F.: L'Émergence d'un Lexique dans une Population d'Agents Autonomes These de Doctorat de L'Université de, Paris VI (2000)

[10] Monmarché, N., Slimane, Mohamed, Venturini, G.: Interactive Evolution of Ant Paintings. IEEE Congress on Evolutionary Computation, Los Alamitos (2003)

[11] Moura, L.: Swarm Paintings. Architopia: Art, Architecture, Science (ed. Maubant) Institut d'Art Contemporaine (2002)

[12] Resnick, M.: Turtles, Termites and Traffic Jams: explorations in massively parallel microworlds. MIT Press, Cambridge (1994)

[13] Shiffman, D.: Swarm. Emerging Technologies Exhibition. Siggraph, Los Angeles, LA, USA

[14] Shoham, Y., Tennenholtz, M.: Emergent conventions in multi-agents systems: initial experiments results and observations. In: Proceedings of the 3rd International Conference on Principles of Knowledge and Reasoning (1992)

[15] Theraulaz, G., Bonabeau, E.: A Brief History of Stigmergy. Artificial Life 2 (1993)

[16] Urbano, P.: Jogos Descentralizados de Consenso ou de Consenso em Consenso. Tese de Doutoramento, Universidade de Lisboa (2004)

[17] Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.): EvoWorkshops 2005. LNCS, vol. 3449. Springer, Heidelberg (2005)

[18] Urbano, P.: Consensual Paintings. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 622–632. Springer, Heidelberg (2006)

[19] Whitelaw, M.: Metacreation: Art and Artificial Art. MIT Press, Cambridge (2004)

Minimal and Necessary Conditions for the Emergence of Species-Specific Recognition Patterns

Nicolas Brodu

INRIA-IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
Concordia University, 1455 de Maisonneuve Blvd. W. Montreal, QC H3G 1M8, Canada
nicolas.brodu@free.fr

Abstract. A simple mechanism is presented for the emergence of recognition patterns that are used by individuals to find each other and mate. The genetic component determines the brain of an individual, a machine learning architecture which is then used to transmit knowledge. Thanks to the interactions between the genetic and the knowledge parts the agents get to use species-specific recognition patterns, starting from an initial condition where the species are not distinguishable. Several machine learning architectures are investigated, as well as the influence of space and asynchronous genetic algorithm operations. Agents selecting each other for mating based on their limited recognition capacities is all that is needed for the emergence of species-specific recognition patterns: the transition between symbols to sequences with an intrinsic role within the species.

1 Introduction

Computer simulations are powerful tools to analyze the emergence of language [1], but despite the progress they entail [2] the field remains controversial [3,4,5,1]. The work introduced here is about interactions between communication and reproduction. Previous related work have studied for example the knowledge transmission of the categorization of object attributes [6], or have introduced specific mappings between meaning and symbols [7]. The present work does not rely on any a priori semantic concepts. The model is stripped down to the bare minimum: genetic reproduction, and basic learning capabilities. The model also omit social interactions [8,9] and ecology [10]. Yet complex patterns emerge for the mutual recognition of individuals belonging to the same species. The goal is then to find the necessary and sufficient conditions for the emergence of these mutual recognition patterns. These patterns may perhaps serve as basis for a protolanguage [11,5], which may then be extended into a full-featured language thanks to social interactions [9]. This work is about how some of the precursor patterns may form in the first place, the transition from isolated symbols to sequences, not about the later two transitions to a full-featured language.

Communication is imperfect and takes the form of strings of symbolic values. Each individual emits a string, and it is presented with the strings from the other individuals. The task is then to find a suitable mate, which necessarily implies the formation of specific patterns for higher than random recognition rates. By design less symbols are available than the number of species, so mono-symbol sequences cannot uniquely identify a species and more complex patterns are required. As time passes the agents get to recognize each other better, using these more elaborated strings.

Genetics alone cannot determine the recognition patterns for the simplest models. Knowledge transmission alone cannot solve the problem either: there are few and noisy learning instances. Hence the combination of both is necessary for the agents to agree on more complex recognition patterns. Species are initially indistinguishable. The individuals who could find a mate may teach the others, according to the algorithm presented in Section 2 and 3. Genetics act on the brain structure. Several simple machine learning models are compared so as to determine the minimal conditions for the emergence of the recognition patterns. These models are detailed Section 4. Results are presented in Section 5, and the role of space is then investigated in Section 6, together with the influence of synchronizing or not the genetic algorithm operations in time. Section 7 concludes this work and proposes possible extensions.

2 The Model

The goal of this experiment is to investigate the minimal and necessary conditions for the emergence of species recognition patterns. A simulation model is built accordingly: Each agent is equated to an AI model for performing string recognition and symbol production. The parameters of this AI model form the genome for the agent (See Fig. 1). Several AI models are used, they are presented in Section 3. Each agent produces a "song". Each agent then selects a mate according to how much it likes the other agents songs. Only agents choosing a mate from the same species may reproduce. The symbols that are used to build the songs are assumed to be available and identical for all agents. Agents

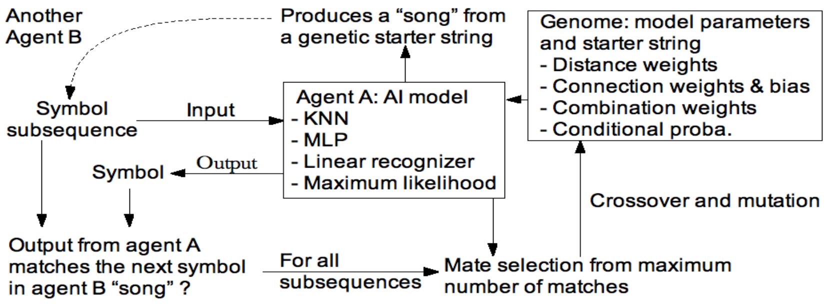

Another Agent B
Produces a "song" from a genetic starter string
Genome: model parameters and starter string
- Distance weights
- Connection weights & bias
- Combination weights
- Conditional proba.
Symbol subsequence
Input
Agent A: AI model
- KNN
- MLP
- Linear recognizer
- Maximum likelihood
Symbol
Output
Crossover and mutation
Output from agent A matches the next symbol in agent B "song" ?
For all subsequences
Mate selection from maximum number of matches

Fig. 1. The simulation model

communicate only through the sequence exchange. In particular, there is no way for an agent to assert the species of another agent except by inferring this information from the symbol sequence of the other agent. So, this model operates on the transition from isolated symbols to recognition patterns ; it aims at providing some reasons why a structure may appear in the symbol sequences. In a first time, the model is simplistic with discrete time (synchronous genetic algorithm) and no spatial structure. Section 6 then extends the model to continuous time (asynchronous genetic algorithm) and three-dimensions, in order to alleviate the fairly strong assumptions of the discrete model.

3 The Genetic Algorithm/Machine Learning Interactions

The core of the algorithm is a feedback loop between the genetic and the machine learning algorithm components: Each individual must recognize a mate for the selection process, amongst all individuals from all species, and only successful individuals may reproduce. There is thus no explicit fitness function. Each individual produces a "song" that will be presented to the others during a "mating parade" (see Fig. 1). The discrete time, synchronous genetic algorithm is sketched in Fig. 2.

```
for each generation
  All individuals produce a "song"
  for each individual
    listen to all other songs, choose one mate
    if same species => put in reproducers set
  for maximum R reproducers at random per species
    mate to produce a child
    child trains on both parent songs
    selected partner trains on reproducer song
    replace one individual at random by the new child
```

Fig. 2. The core algorithm where both genetic and knowledge components interact. The main text introduces the algorithm details, like R which is the maximum turnover rate.

More formally, a song is a series of symbols, $(s)_i$ with $i = 1..M$ and M the maximum song size. Random series are presented to the first generation in order to bootstrap the experiment. A moving window of size N inputs is applied to each substring $(s)_j$ of each training song, with $j = i \dots i+N-1$. The individual is trained to produce s_{i+N} the next symbol of the sequence for each such substring (see Fig. 3). For the first symbols $i = -N \dots -1$ of the sequence there are not enough previous symbols to fill the moving window and genetically determined symbols fill the substring (Fig. 3). The generation of the songs is the natural reverse operation: The genetic starter string is presented and the individual is asked to produce a symbol based on its previous knowledge.

Some genetic starter strings G are better suited than others for learning some sequences. Suppose $G = AAAA$ and the task is to learn the song AB. In this case, the system will generate two conflicting training instances $AAAA \rightarrow A$ and

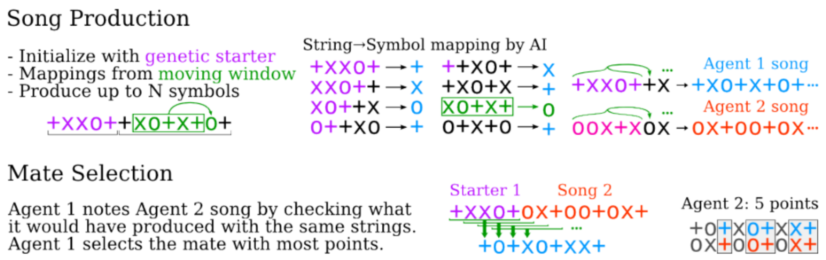


Fig. 3. Song production and mate selection from symbol sequences

$AAAA \rightarrow B$. On the other hand if $G = CDDC$ there is no conflict. The genetic part thus has an influence on how well it is possible to learn some sequences. Conversely there are several possible genetic starters which are all equally suited with respect to learning a particular song. Hence the genome does not determine the species song, it merely defines for each individual a subspace of all possible sequences for which there is no conflict. Any song within these subspaces intersection can be learned equally well by all members of the species, even if they have different genetic material.

In the mate selection task each agent classifies the candidate songs by order of preference. Each candidate song is fed as input, possibly after alteration by imperfect communication (symbols are modified at random with a predetermined probability). The individual then estimates what symbol it would have produced for each substring (see Fig. 3). When the symbols match the candidate gets one point. Each individual selects a mate with maximal points, choosing at random for ex-aequo candidates. The agent may reproduce if the selected partner belongs to the same species.

The mating process is straightforward: crossover and mutation of the AI model parameters between the parents so as to produce the offspring. But at this point the new child has no training and is thus unable to produce its own sequence at the next generation. The parents songs are used so a child starts with only two training instances (possibility imperfectly communicated). But now, if all individuals from one generation successfully reproduce they are all killed and replaced by their offspring, destroying the knowledge that accumulated with time. In order to eliminate this risk a maximum number of children R is introduced in each species. With this setup the individuals get a chance to survive more than one generation and accumulate knowledge. However no mechanism has yet been introduced that would allow this accumulation (only one that prevents the non-accumulation). In order to get more training instances a selected mate also trains on the reproducer which selected it (Fig. 2). Thus, as time passes, agents now have a chance to accumulate knowledge.

4 The Different Machine Learning Models

Any machine learning model may be used in the algorithm in Fig. 2. This model receives as training instances (sequence, symbol) pairs, and must predict a symbol when presented other (possibly unknown) sequences. The models chosen for this study are simple ones, exploiting different information, since the goal is to investigate what are the minimal conditions for the emergence of mutual recognition patterns.

4.1 The Linear Classifier

As often for categorical data (the symbols) each input is duplicated into L entries, with L the size of the alphabet. Each of these entries is here set to $+1$ or -1 if the corresponding input matches that entry value. For example: with $L = 3$ symbols ABC, the input string $ABCA$ is mapped into the 12 entries vector $I = (+1, -1, -1, -1, +1, -1, -1, -1, +1, +1, -1, -1)$. Similarly the vector P for the symbol to predict contains S outputs.

The training instances are converted in this (I, P) format. For T training instances the I vectors form a $N.L \times T$ matrix A and the P vectors a $L \times T$ matrix B. The least squared error solution for the equation $WA = B$ gives the weights W that are used by the linear classifier. Then, for a new unknown instance J, the predicted vector is $P = WJ$. The output symbol for J is extracted from P as the one with largest entry. For example suppose $P = (0.68, -0.87, 0.05)$, then the symbol A is returned. When two symbols have equal value one is chosen at random, which makes the machine learning algorithm occasionally non-deterministic. This is acceptable in our context, especially since any song may be altered randomly later on by imperfect communication anyway. The linear classifier model has no genetic component in addition to the initial sequence of symbols presented in the previous section.

4.2 The 2-Layer Perceptron (MLP)

The same setup as for the linear classifier is reused for mapping the symbols to categorical data. A 2-layer perceptron then processes the input data. More precisely, the $N.L$ categorical entries are connected to the input neurons. This input layer is connected to 10 hidden neurons with sigmoidal transfer function ($f(x) = x/(1 + \text{abs}(x))$ is used here for its reduced computational costs compared to the more usual tanh, see [12]). An output layer with linear activation functions finally maps the results of the hidden layer to the L output categorical entries. The training set is formed as before. The MLP is trained simply by performing 30 steps of batch gradient descent with a learning rate of 0.1 over all known instances. The MLP initial connection weights and biases before learning form an additional genetic component, together with the initial sequence of symbols presented in the previous section. When the training set is fixed (i.e. when individuals have agreed on a unique species recognition pattern) then individuals who have a genetic information (initial weights) that is better suited to this training set have an advantage over the others since they need less training, hence a Baldwin effect is expected [13].

4.3 The K-nearest Neighbors (KNN) Classifier, with $K = 5$

The KNN model represents a simple form of learning by imitation of previously observed instances. [14] notes that "simple models of cultural transmission solely based on imitation are not sufficient to permit linguistic co-ordination". However, as mentioned in the introduction, the current work in not about the emergence of a full featured language, just about the emergence of recognition patterns. The more elaborated mechanisms that would additionally be necessary to turn these precursor patterns into a full language are out of scope. However the only way to assert whether the KNN model – imitating previous instances – is sufficient for the emergence of the recognition patterns, is to test it in practice.

Each training sequence $S = (s)_i$ with $i = 1 \ldots N$ is kept with the associated next symbol X, forming a pair (S, X). When a symbol has to be predicted from an unknown input sequence Σ, the distance between Σ and each known S is computed. That distance is simply the number of differences between Σ and S. For example, $ABCD$ and $AACD$ are at distance 1, $ABCD$ and $BBCA$ are at distance 2, etc. The K nearest S are then selected, with ex-aequo chosen at random if necessary. Then, for each of the up to K neighbors, the symbol X associated to that neighbor S is given a weight v_k. This weight v_k depends on the neighbor distance order k. Summing over all neighbors, the output symbol with the largest total weight wins the selection: It is returned by the classifier as the result of predicting the sequence Σ.

In this model the votes $(v_k)_{k=1\ldots K}$ associated to each of the K neighbors (in distance order) are genetic parameters in addition to the initial sequence of symbols presented in Section 2. $K = 5$ has been chosen for this study, though with the voting mechanism it may happen than some of the v_k become null during the genetic evolution and thus reduce the effective value of K.

4.4 The Assembly of Maximum Likelihood (ML) Estimators

For a sequence $S = (s)_i$, a ML learner seeks to maximize the probability of this sequence $p(s_1 ... s_N|t)$ over all possible output symbols t. Unlike the more usual approach of maximizing $p(t|s_1 ... s_N)$, the probability of obtaining t given the observed sequence, the maximum likelihood approach discriminates between competing sequences. The probabilities are noted from the samples, but unfortunately estimating $p(s_1 ... s_N|t)$ requires monitoring L^{N+1} combinations (one L^N for each t), with L the number of symbols. A simple yet limited solution is to consider that inputs are independent, simplifying $p(s_1 ... s_N|t)$ into $\prod_{i=1\ldots N} p(s_i|t)$, hence reducing the complexity to $N \times K^2$ combinations. An intermediary solution allowing one level of dependence has been chosen for this work. Inputs are gathered into mutually independent groups (assumption A1). A main input is chosen in each group, and the other group members are assumed to be independent conditionally to this input (assumption A2).

Example: Suppose $N = 5$, with two groups $\{s_1, s_2, s_3\}$ and $\{s_4, s_5\}$, and with s_1 and s_4 the group leaders. In this case:

- $p(s_1, s_2, s_3, s_4, s_5|t) = p(s_1, s_2, s_3|t).p(s_4, s_5|t)$ Using A1
- $p(s_1, s_2, s_3, s_4, s_5|t) = p(s_2, s_3|s_1, t).p(s_1|t).p(s_5|s_4, t).p(s_4|t)p(s_1, s_2, s_3, s_4, s_5|t) = p(s_2|s1, t).p(s_3|s_1, t).p(s_1|t).p(s_5|s_4, t).p(s_4|t)$ With A2

One level of dependence is thus kept, while maintaining the number of combinations to monitor in $O(K^3)$ instead of K^{N+1}: one K^3 for each $p(s_j|s_i, t)$ with s_i a group leader and s_j in that group. An assembly of maximum likelihood predictors was introduced so as to deal with more complex songs: Several predictors are maintained in parallel, each with its own conditional dependence assumptions on the inputs. The final predicted symbol is simply the result of a majority vote amongst the predictors.

For each possible output symbol t, $p(s_1...s_N|t)$ is computed using the decomposition presented above. The symbol with maximum likelihood value is selected. In the case where some $p(s_i|t)$ were not observed, the selection operates between outputs with less unknown $p(s_i|t)$. This is equivalent to still noting known subsets of inputs when the whole sequence is unknown. When all input combinations are unknown no output is selected and majority is then voted amongst the other predictors in the assembly (which use different grouping assumptions). When all predictors fail the song simply stops. Ex-aequo situations are solved by choosing one candidate solution at random.

In this study 3 ML estimators are gathered in an assembly. The grouping information for the conditional dependence between the inputs form an additional genetic material together with the initial starter sequences.

5 Results for the Synchronous Case

As all species receive equal treatment the results can be averaged over all species to give synthetic indicators for the whole population. The experiments in this section use: 6 species, 49 individuals per species, a moving window size of $N = 5$ inputs, a maximum song size of $M = 10$ symbols, and a maximum turnover rate of 20% (fixing R in Fig. 2). A first experiment is performed using 3 symbols. There is a probability of 0.01 that each time a symbol is transmitted it is replaced by another one at random. A second experiment reduces the number of symbols to 2, and a third experiment studies the effect of removing the transmission errors. 20 batches of runs are performed with the same random seeds for each experiment, and repeated again for all 4 machine learning algorithms. The results are plotted in Fig. 4-7.

Fig. 4 highlights the failure of the ML model to produce recognition patterns. The simplest linear classifier is less efficient than the KNN and MLP models in the noisy scenarii (left and middle plots). The number of symbols does not seem to influence much the models, except for the ML recognizer. An hypothesis would be a lack of training examples so to produce reliable statistics in the ML model, with more symbols meaning more combinations hence even less instances for each combination. Experiments performed where all individuals listen to all the species songs tend to confirm this hypothesis by improving the performance of the ML model.

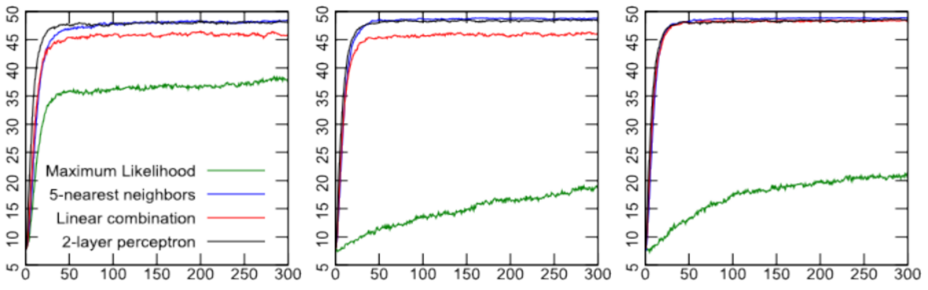


Fig. 4. Evolution of the number of reproducers vs. number of generations, when 2 symbols are in use (left), 3 symbols (middle), and 3 symbols with no transmission error (right)

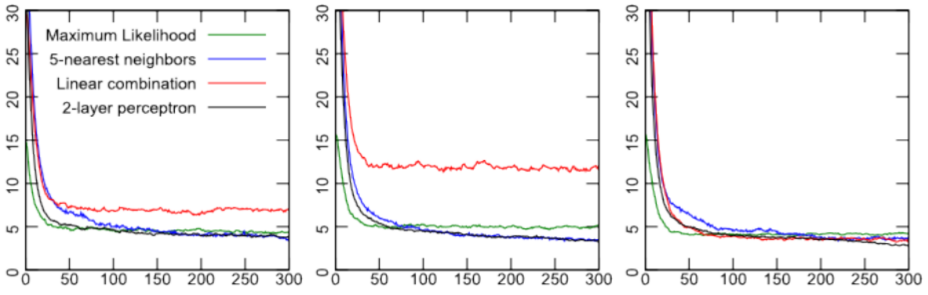


Fig. 5. Evolution of the number of songs vs. the number of generations, when 2 symbols are in use (left), 3 symbols (middle), and 3 symbols with no transmission error (right)

Fig. 5 introduces the number of songs used in each species. It is not obvious whether the present scenario converges or not to a unique song for each species, given the limited number of training instances for the children and the transmission errors. Powerful AI models may also learn several songs. Once again the KNN and MLP models are relatively insensitive to both noise and number of symbols. The linear recognizer is too sensitive to noise, as is apparent from both Fig. 4 and Fig 5.

Fig. 6 shows the repartition of the individuals using the few songs that are present in each species. In this synchronous scenario the dominant song is shared by a large majority of the individuals. The remaining songs are variants emitted probably by individuals without enough training (like the children). Some examples of dominant songs produced at the end of the 300 generations (with symbols noted as numbers) are the obvious mono-symbol sequences like 2222222222, etc., the cycle-2 patterns like 0101010101, and other repetitive patterns like 0110110110, 1100011000, 1201201201, etc. The patterns may also be more complex, like 2221102212: Even though the fixed window size of N inputs would eventually make the trailing sequence in these patterns cyclic, the genetic starter string must be taken into account for determining the first symbols, which are

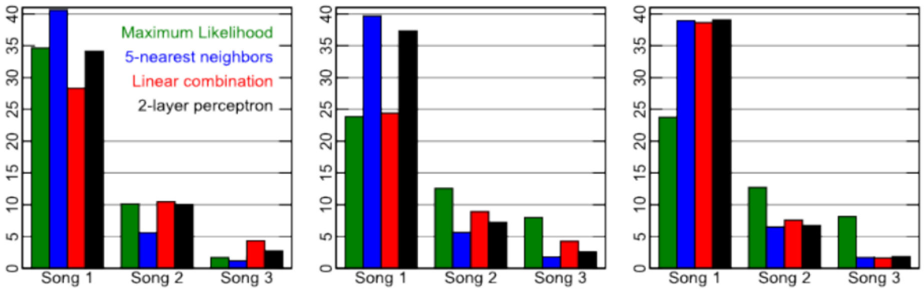


Fig. 6. Number of users for the three main songs in each species, when 2 symbols are in use (left), 3 symbols (middle), and 3 symbols with no transmission error (right)

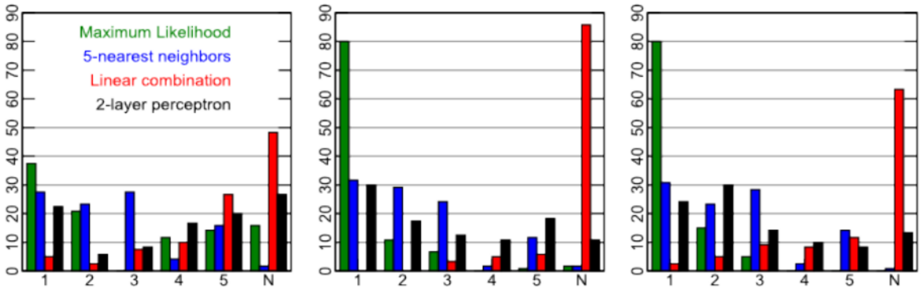


Fig. 7. Percentage of songs with given cycle lengths in abscissa (N = no cycle), when 2 symbols are in use (left), 3 symbols (middle), and 3 symbols with no transmission error (right)

thus not part of the eventual cycle, but nevertheless included in the pattern recognition between individuals.

Fig. 7 displays the repartition of the songs according to their cycle length. The number of acyclic (over the first symbols) songs is highest for the linear recognizer, possibly due to the aforementioned sensitivity of that model. The ML model fails to produce distinctive patterns for each species, which corroborates Fig. 4: that model could not produce the more complex songs, necessary to overcome the symbol limit. Fig. 5, right, shows that the MLP and the KNN have similar performances. Fig. 7 shows however that the KNN classifier makes use of simpler recognition sequences on average, while the MLP produces a more diverse complexity repartition.

In order to investigate what are the intrinsic capabilities of each algorithm, a simple solution is to disable the genetic or the knowledge transmission part. Without knowledge transmission only the genetic structure may evolve, and without the genetic algorithm the initial agents may only learn from each other without producing new children. Fig. 8 proves that both components are necessary for the emergence of efficient recognition patterns, though the two most successful models (KNN and MLP) still exhibit limited capabilities with only

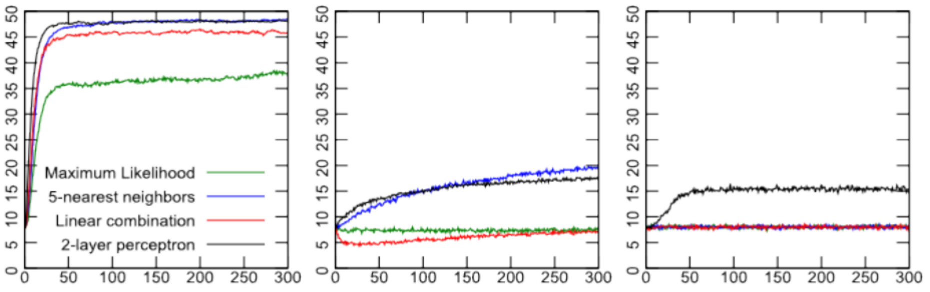


Fig. 8. Evolution of the number of reproducers vs. number of generations, with only knowledge transmission (middle) and with only the genetic algorithm (right). The left plot from Fig. 2 is reproduced with a similar scale to ease comparison.

one component active. The interactions between the genetic and the knowledge transmission parts, however, are necessary for producing real recognition patterns: the levels obtained with the partial cases correspond to less than half the population successfully recognizing each other.

6 Results for the Asynchronous Case

The synchronous selection operation without spatial organization is useful for determining the respective influence of the models, but it imposes a fairly severe constraint on the genetic algorithm. Moreover these assumptions go against the goal of analyzing the minimal conditions for the emergence of the recognition patterns. A more general framework is thus needed, where the influence of the spatial distribution of agents may be studied, together with the possibility for the agents to reproduce at any time. Fig. 9 is a capture of the 3D simulation, with

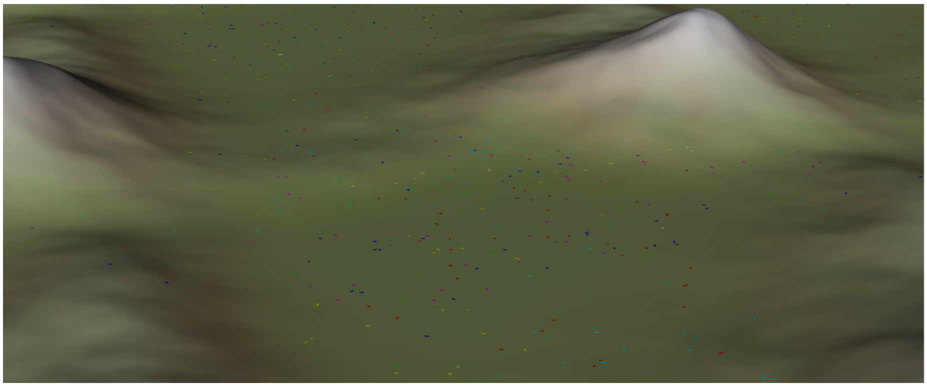


Fig. 9. Three-dimensional environment with an asynchronous genetic algorithm

continuous space and time. The agents are embodied as vehicles with definite mass, position and velocity, and wander in the world with the aim of avoiding collisions. No further AI is given to the agents. Each agent chooses a mate as before, but only amongst neighbors present within a predetermined radius. The influence of space on the simulation is studied by varying the search radius. The agents reproduce at their own rhythm, determined by a frequency and a phase. Each agent has its own phase so the genetic algorithm operations are performed asynchronously in time.

Another change from the basic experiment is necessary due to the spatial localization: a minimum delay between reproduction events. This minimum delay ensures that a child has some time to move away from its parent, and that isolated mates don't reproduce too fast independently of the rest of the species. The asynchronous aspect is also enhanced, since the delays are randomly set for each reproduction event. A negative learning was finally introduced in the scenario, with agents learning instances that do not lead to a mating operation as bad sequences, with the hypothesis that it could improve the species recognition.

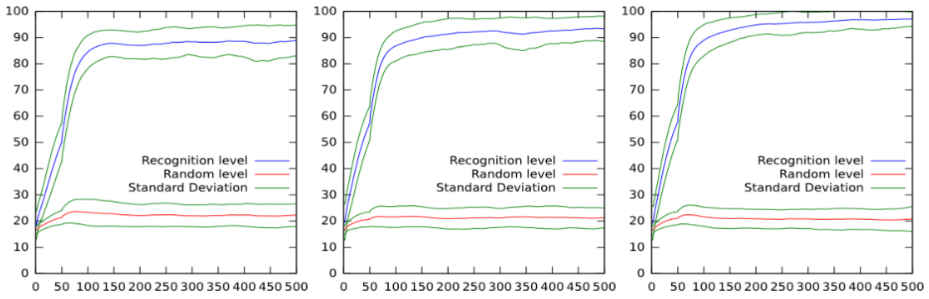


Fig. 10. Evolution of the recognition level vs. simulation time, when 2 symbols are in use (left), 3 symbols (middle), and 3 symbols with no transmission error (right)

Results for the recognition capabilities are given in Fig. 10, using the KNN learning algorithm, by averaging the results for 6 species over 20 experiments. These plots are the equivalent of Fig. 4 in the present asynchronous scenario.

The base random level is computed by checking how much agents from the same species were present in each neighborhood at each reproduction event; it gives the chance an agent would select another one from the same species at random. As before, at the beginning of the simulation the agents start with no prior knowledge and do not better than random. As time passes, the average recognition level over the past 50 time units is monitored, and increases up to a point where the agents in each species can recognize each other with a good accuracy.

Figure 11 shows the influence of space on the convergence to recognition patterns, as well as the influence of the negative learning and the transmission error. The negative learning does not have a significant effect on the agent performances. However space is found to be a major factor: When the search radius

	Small radius	Medium radius	Normal radius (Fig.10 middle)	No negative learning	No error (Fig. 10 right)
Mean Rec. Level	69.9	85.0	93.3	92.8	97.1
Dev. Rec. Level	10.2	6.96	4.76	4.70	2.75
Mean Num. Songs	13.1	11.6	11.4	10.5	4.31
Dev. Num. Songs	8.92	8.74	7.96	7.58	4.20

Fig. 11. Recognition levels and number of songs in each species at the end of the run, for various asynchronous scenario configurations, with 3 symbols and 6 species

is too small the individuals from the same species do not learn to recognize each other as efficiently, possibly due to the agents using different recognition patterns at different places, as is reflected by an increased number of songs. The effect of removing the 1% transmission error is however clearly visible: A better recognition rate, and much less diversity in the patterns used within a species. This contrasts from Fig. 5 in the synchronous case, where the number of songs was not noticeably affected by the removal of the transmission error.

7 Conclusion

A framework was presented where the genetic component and the knowledge acquired during an agent lifetime interact with each other: The genetic material defines the innate processing power of an individual, its capabilities for learning. In turn, the knowledge an agent acquires directly influences its success at reproduction. Both parts may be transmitted to the next generations: the genetic component using a crossover/mutation algorithm, and the knowledge using machine learning techniques built according to these genetic instructions.

The main findings of this work may be summarized by:

1. The learning mechanism needs to be simple and robust (failure of the linear model, Fig. 4 and 5, and of the ML classifier, Fig. 4).
2. Too powerful models are sufficient, but not necessary: the KNN model is simpler than the MLP and converges to the same performances (Fig. 4).
3. Complex recognition patterns are produced for free (Fig. 7, cycle lengths).
4. Asynchronous reproduction events in continuous time do not seem to alter the performances of the KNN model (Fig. 10).
5. However when the agents are too spatially isolated the recognition performance drops (Fig. 11).

The original problem of determining the minimal and sufficient conditions for the emergence of mutual recognition patterns can now be answered. According to the present study results, it seems that good candidate conditions are: 1. A turnover of agents in the genetic algorithm so as to produce new patterns, and 2. A limited form of knowledge transmission by imitating previous instances. In particular imperfect transmission is not a necessary condition (though no error improves performances), but a sufficient spatial distribution may be necessary. Additional experiments using more species (up to 12) and symbols (2,3,4) produce similar results that are not included here due to space restrictions.

An extension to this work could be the introduction of an external environment, allowing more advanced forms of communication, like stigmergy. The current setup has been restricted on purpose to a bare-bones model where the agents interactions are strictly controlled. Yet, this prevents group effects and other collective behaviors that would be a natural extension to this framework. Another direction of research would be to investigate the influence of the learned part on the genetic component, the Baldwin effect [13]. Visual inspection suggests that in the current setup the genetic starter strings resemble the species specific song at the end of the training, but the more general question is why this is so and whether this is always necessarily the case. For example, for some AI architecture a genetic starter similar to the dominant species song introduces more training substrings, hence provide a selective advantage over individuals without the correct starter sequence. For more elaborated AI algorithms, and also for more complex environments with "social" interactions not restricted to choosing a mate based on its song, it is possible that the Baldwin effect operates in more complicated ways.

In any case, the current experiments have shown that very few preconditions are needed for the emergence of species-specific recognition patterns. What this shows is that the transition from isolated symbols to precursor sequences for more elaborated forms of communication, like language, is not exceptional. The more interesting question of how the precursor patterns may then turn or not into these advanced form of communication is, however, an open question.

References

1. Wang, W.S.-Y., Ke, J., Minett, W.J.: Computational Studies of Language Evolution. In: Computational linguistics and Beyond, pp. 65–106. Huang, C.R. & Lenders, W (2004)
2. Wagner, K., Reggia, J.A., Uriagereka, J., Wilkinson, G.S.: Progress in the simulation of emergent communication and language. Adaptive Behavior 11(1), 37–69 (2003)
3. Harris, R.A.: Review of "The language instinct" by S. Pinker. The Globe & Mail (1994)
4. Smith, J.M., Szathmáry, E.: The major transitions in evolution. Oxford Univ. Press, Oxford (1995)
5. Calvin, W.H., Bickerton, D.: Lingua ex Machina: Reconciling Darwin and Chomsky with the human brain. MIT Press, Cambridge (2000)
6. Acerbi, A., Parisi, D.: Cultural Transmission Between and Within Generations. J. of Artificial Societies and Social Simulation 9(1)9 (2006)
7. Gong, T., Wang, S.-Y.W.: Computational Modeling on Language Emergence: A Coevolution Model of Lexicon, Syntax, and Social Structure. Lang. and Linguistics 6(1), 1–41 (2005)
8. Beecher, D.M., Burt, M.J.: The Role of Social Interaction in Bird Song Learning. Cur. Dir. in Psychological Science 13(6), 224–228 (2004)

9. Dessalles, J.-L.: From protolanguage to language: model of a transition. Marges linguistiques 11, 142–152 (2006)
10. Slabbekoorn, H., Smith, T.B.: Bird song, ecology and speciation. In: Philosophical trans. of the Royal Society of London. Series B, Bio. Sciences. 357,1420, pp. 493–503 (April 2002)
11. Jackendoff, R.: Possible stages in the evolution of the language capacity. Trends in Cognitive Sciences 3 (July 7, 1999)
12. Elliott, D.L.: A Better Activation Function for Artificial Neural Networks. Technical Report 93-8, Institute for Systems Research, University of Maryland (1993)
13. Baldwin, J.M.: A New Factor in Evolution. American Nat. 30, 441–451, 536-553 (1896)
14. Oudeyer, P.Y., Kaplan, F.: Language evolution as a Darwinian process: computational studies. Cogn Process 8, 21–35 (2007)

Artificial Ants for the Optimization of Virtual Keyboard Arrangement for Disabled People

Sonia Colas, Nicolas Monmarché, Pierre Gaucher, and Mohamed Slimane

Laboratoire d'Informatique de l'Université de Tours,
64 avenue Jean Portalis, 37200 Tours, France

Abstract. Many physically impaired people can meet difficulties when using a computer and particularly with the keyboard. Often, a virtual keyboard can improve the usability of the computer system by handicapped people and can be adapted to their disabilities. From a combinatorial point of view, artificial ants have been studied to improve classical keyboard arrangements. This paper present how this method can fit the first problem and then become a tool to build new "tailor-made" keyboards. Then, the artificial ants meta-heuristic can be used as often as needed as an optimizer that take into account user's activities of writing.

1 Introduction

Handicapped persons are dependent on their computer to work and communicate. Many of them are heavily disable and computers are the best way to give them a feeling of autonomy.

Internet has now become to play a prominent role in commercial services (on-line shopping, train or plan reservation,...) but also in administrative services. These on-line services can be perceived as a quick mean to get documents, informations or goods. In order to make internet access easier for handicapped persons, several laws have been voted to improve web sites compliance with WCAG 1.0 (Web Content Accessibility Guidelines) norm [1] from the Web Accessibility Initiative (WAI) [2]. Since 1998, USA have started to consider the problem of accessibility and more recently Europe have decided to oblige institutional web sites to become accessible. This kind of law have only been approved in France during February 2005.

A poll of INSEE [3] has shown that 8 millions of motor handicapped people and 6 millions of sensory handicapped people are living in France. About 20 % of the first group are affected with serious paralysis and as population is getting older, this tendency will not stop soon.

In order to reach the Internet, handicapped people first have to be able to use their computer alone. They often have to use specialized devices, called assistive technologies, adapted to their specific disabilities. For instance blind people can use braille terminals or voice synthesis instead of the screen and a standard keyboard. Some motor handicapped persons can use a classical computer mouse but others could prefer other pointing devices like touch-pads or game's joysticks[1].

[1] Virtual Projects : JoyMouse. At http://www.vp-soft.com/software/joymouse.php

In order to assist people with special needs, four kinds of keyboard-like systems can be found:

– modified and improved real keyboard (hardware)
– software that simulates a standard keyboard on screen (virtual keyboards)
– software that simulates an improved keyboard on screen (virtual keyboards with static or dynamic improvement of keys'position)
– software that simulates a keyboard on screen and which suggests words (virtual keyboards with prediction ability)

In our work, we focus our effort on the third kind of keyboards, i.e. a virtual keyboards for which keys'positions are chosen in order to minimize the typing energy of the user.

The remainder of this paper is organized as follows: next section quickly describes various real and virtual keyboards, then section 3 deals with artificial ants and previous works about keyboards. In section 4, we describe our method for the specific case of virtual keyboards and we give results of our experiments in section 5.

2 Overview of Assistive Technologies to Take Place of Keyboards

2.1 Hardware Solutions

There exists a lot of modified keyboards that can help disabled persons to enter textual data in computer. For instance, the *Contoured* keyboard (fig. 1.a) proposed by Kinesis Ergo[2] has been designed to reduce arms and hands moves while

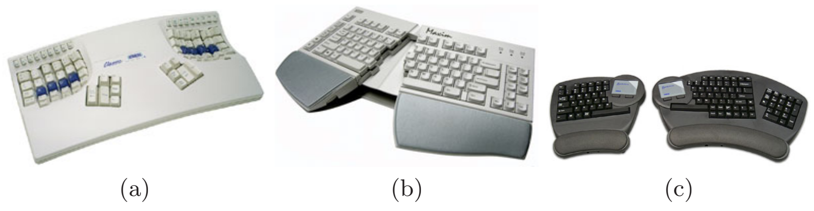


Fig. 1. Modified keyboards (Kinesis Ergo): contoured keyboard (a), maxim keyboard (b) and evolution keyboard (c)

typing. Keys are clustered in two groups, one for each hand. The *Maxim* keyboard (fig. 1.b) displays an example of keyboard that can take various angle values between the two arms (wists can rest on mobile parts, as often provided with keyboards). The *Evolution* keyboard if made of two parts, totally independent.

These kind of keyboards can of course be used by ordinary people who are trying to increase their typing efficiency or their everyday comfort. But more

[2] Kinesis Corporation: Computer ergonomics. At http://www.kinesis-ergo.com

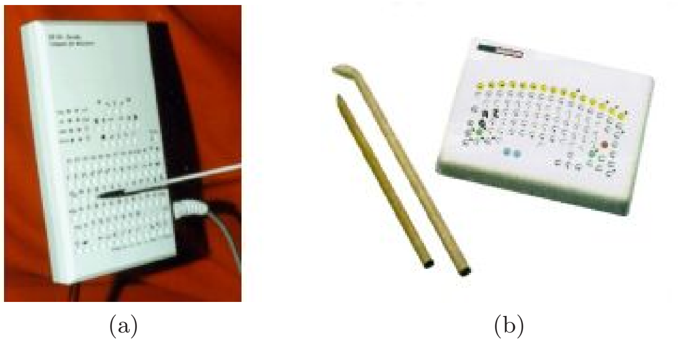


Fig. 2. Specific keyboards for handicapped people using a stem with mouth: M32h keyboard by SEVEKE (a) and GT keyboard by GORLO (b)

specific keyboards also exist. For instance small keyboards which are used with a mouth stick: they must be sensitive to low pressure and most frequently used keys must be near the center of the keyboard (see fig.2).

Other specific keyboards can also be found: for only one hand (fig.3.a) or with an improved separation of keys (fig.3.b).

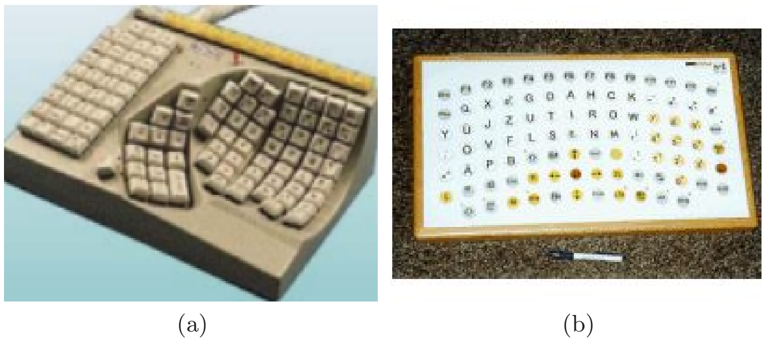


Fig. 3. Other examples of adapted keyboards: ergonomic keyboard MALTRON for right hand (A) and improved keyboard SUMO (b)

2.2 Virtuals Keyboards

For a simulated keyboard, also called virtual or screen keyboard, it is necessary to move a cursor to select a key. Cursor moves can be obtained by a mouse, joystick or any physical input device. Even if this keyboard is displayed on screen, for a given user (i.e. with his/her particular handicap and particular tasks to perform) a bad arrangement of keys can slow down his/her typesetting rate since the pointer moves are similar to the finger moves of a single finger user.

Sometimes, a mouse click is even not possible to obtain from the user and the click must be automatic. Several virtual keyboards, such as Windows XP virtual keyboard, Click-N-Type keyboard [4], CVK keyboard [5], ScreenDors 2000 keyboard [6] or Keystrokes for Apple computers [7] propose this "autoclick" functionality: the key is selected if the cursor does not move for a given time.

Two methods can be used to scroll up and down the key list: linearly and automatically (one dimension) or bidimensional (within rows and columns). In case of particular serious handicap, the user can only use the first method and has to wait that the correct key is present under the cursor, it is then sometimes possible to first select a group of keys and after the key inside the group. For instance (see fig.4), keys can be virtually clustered in Microsoft virtual keyboard as in Clavicom keyboard [8].

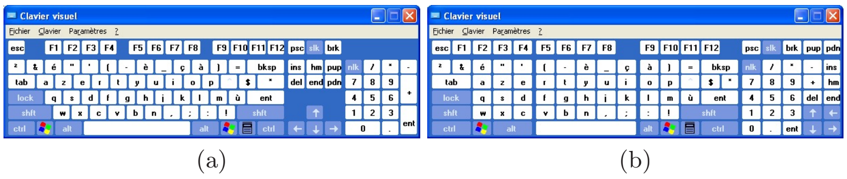


Fig. 4. Microsoft Windows XP virtual keyboard: standard configuration (a), and block configuration (b) for automatic scrolling

Several improvements are often present: CVK keyboard can zoom on the selected key, Click-N-Type keyboard can spell scrolled keys. It's often possible to use sound to verify typing (CVK, Clavicom, Wivik [9]). Moreover, virtual keyboards can modify their display characteristics: size of keyboard/keys can vary (ScreenDoor and Wivik).

In order to improve the typing rate, keyboards are equipped with a prediction system (except Microsoft windows XP virtual keyboard): CVK, Vitipi [10], Wivik, Clavicom, ScreenDoors, Keystrokes and Click-N-Type all use a word prediction system. Thanks to a dictionary, they can provide a word list from the first selected letters. Skippy keyboard uses the probability to use a word after another one to sort the proposed list. A multi-lingual dictionary is used in Vitipi but without any grammatical rules. At the opposite, the HandiAs system [11] can use French grammar but can also be extended to foreign grammars. Sometimes, the dictionary can be automatically filed with new words (ScreenDoors).

Figure 5 displays the CVK keyboard with its word prediction system. This keyboard is an open source software and can be embedded in any Windows application (here the notepad).

As words can be predicted, letters probability can also be exploited: for instance with Keyglasses [12] and Sibylettre [13] virtual keyboards. With Keyglasses (see fig.6), the most probable letters that could be typed appears with transparency around the last typed one (in order to reduce the distance to reach these keys). This can be particularly useful for people using a pointing device (like a mouse). With Sibylettre, after each letter, the keyboard is re-organized

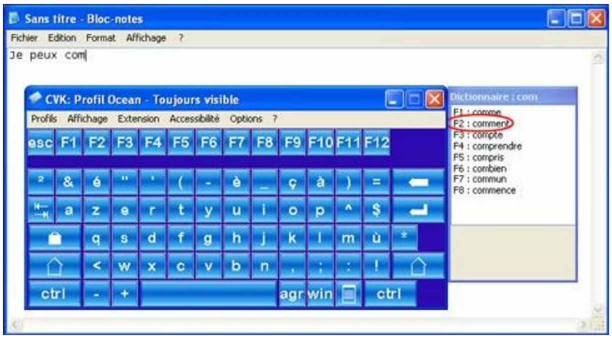


Fig. 5. Words prediction system with CVK keyboard

in order to first present the most probable letters. This can be useful for people using a linearly scrolling of keys but, of course, if the keyboard is modified at each key stroke, this can be a problem when the user selects letters with a pointing device.

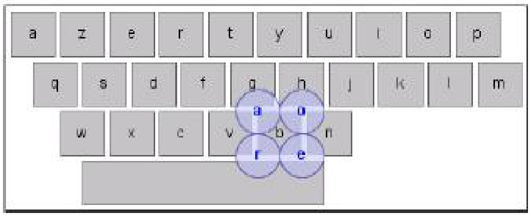


Fig. 6. Virtual keyboards with letters prediction system: Keyglasses keyboard after the letter 'b'

In our work, we try to improve the typing rate (or typing tiredness) by reorganizing the keyboard according to user habits.

3 Artificial Ants for Combinatorial Optimization

Since 70's, new paradigms inspired by natural systems have arisen in computer science. One of the recent successful idea is the artificial ant paradigm: artificial ant-agents are used to solve collectively hard problems while they use only simple interactions within the colony or with their environment. These works are referred as Swarm intelligence methods and most of the published ones are in the field of combinatorial optimization [14,15].

The ant-colony paradigm has met a great success with combinatorial optimization because problem's underlying graph structure is not too far from the food network that ants can build and exploit with the help of a global shared memory known as pheromones. Artificial ants can build solutions to the problem while they move between vertices of the graph and reinforce edges that belong to promising solution. Then, this positive feedback will orient other ants toward these regions of the search space. The multiple interactions of ants throught this global shared memory tend to collectively provide the emergence of good solutions.

Regarding to the keyboard arrangement problem, a recent work has shown that artificial ants can be efficiently used to find the best arrangement of keyboards [16]. However, this work has two major limitations: (1) once the optimum keyboard for a given language has been found, it is of no use to run the optimization tool once more (unless we consider new keys or new languages) and (2) even if actual keyboards are not optimal, new arrangements, even if they are better, are not spread at all by the keyboard manufacturers or adopted by users. Then, with the adaptation of visual keyboard to disable people, we think that these two limitations would be bypassed: disabilities are most often different from on user to another one (a different keyboard for each user) and if the advantage of a new keyboard can be felt for every days' typing tasks, a user could have more motivation to use a personalized keyboard.

We can notice that genetic algorithms have been recently introduced to tackle this optimization task, for diseable people [17] or for air traffic controller [18].

4 Optimization of a Virtual Keyboard by Artificial Ants

The derived combinatorial problem can then be formulated as follows:

– a keyboard of $m \times n$ keys is considered,
– a set of texts that have to be typeset, or a set of document that are usually produced by the considered user in a given context (i.e. emails, financial reports,...),

We need to know the best arrangement of the keyboard that minimizes the total length of pointer's moves for the given set of texts. Of course, we can note that this problem can be exactly considered as a traveler salesman problem and consequently is NP-complete.

The work of ants is then to assign symbols (i.e. letters or keys) to keyboard locations. To make this assignment, they take into account pheromone quantities as a global information built by the whole colony. At this stage, different possibilities of using pheromones are possible:

– Pheromone quantities are used by ants to choose the next symbol to assign at each time step of their solution building,
– or, pheromones are used by ants to choose, for each symbol, a location on the empty keyboard.

Ant decisions are also influenced by randomness in the same way that they can do in real life when exploring ground around their nest.

Let us notice that we will only consider the first semantic direction for pheromones' role in the following of this paper. This is due to the fact that first experiments of the second method have quickly shown that assigning a pheromone value on the link between a symbol and a location on the keyboard is not a good idea. We have noticed that a small change in location can induce large irregularities in objective function: therefore it is difficult for ants to converge gradually. Then we have decided to consider pheromones between symbols in order to concentrate ants on the order of symbols to be assigned to keyboard locations.

The general framework of the algorithm is given in algorithm 1. This framework is quite classic with this class of population based algorithm: ants build solutions, best solutions are used to update the global memory and this mechanism is repeated for a given number of iterations.

Algorithm 1. Keyboard arrangement optimization with artificial ants

```
1: Initialize pheromone values $\tau^0_{i,j}$
2: for $T^{\max}$ iterations do
3:    for all ant $k$ do
4:       Build a keyboard arrangement $\mathcal{K}_k$
5:       Evaluate the quality of $\mathcal{K}_k$
6:    end for
7:    Update pheromone values according to quality of new solutions and pheromone
      natural evaporation
8: end for
9: Return the best keyboard arrangement found since the beginning
```

4.1 Solution Building

During each iteration $T \in \{1, \ldots, T^{\max}\}$, each ant k builds a solution (i.e. a keyboard) $\mathcal{K}_k$ and uses a collective memory called pheromones in their natural world. Artificial pheromones are real values that are used by artificial ants to build a solution. We note $\tau_{i,j}$, the pheromone quantity between symbol i and symbol j. They also use a local information $\eta_{i,j}$ called desirability which represents a kind of heuristic value specific to each problem instance. This value is calculated once at the beginning of one run whereas pheromones are modified and evolve along the $T^{\max}$ iterations.

At time t of iteration T the ant has built the partial solution $\mathcal{K}_k(t)$ with the last symbol assigned i and performs a new assignment of a symbol j using the following probability:

$$P^{(t)}(i,j) = P_e^{(T)} \times \frac{\tau^{\alpha}_{i,j} \times \eta_{i,j}}{\sum\limits_{l \in N(\mathcal{K}_k(t))} \tau^{\alpha}_{i,l} \times \eta_{i,l}} + (1 - P_e^{(T)}) \times \begin{cases} 1 \text{ if } j = \arg \max\limits_{l \in N(\mathcal{K}_k(t))} \{\tau^{\alpha}_{i,l} \times \eta_{i,l}\} \\ 0 \text{ else} \end{cases} \tag{1}$$

where j is chosen in $N(\mathcal{K}_k(t))$ which corresponds to the set of symbols that are still not assigned in the partial solution $\mathcal{K}_k(t)$. The exponent α is used as a parameter to scale the relative importance of pheromones against desirability.

$P_e^{(T)}$ is the exploration/exploitation probability, which is often a constant in ant algorithms, but here varies along the iterations : $P_e^{(T)} = 0.8(T/T^{\max})$ in order to increase exploitation *vs* exploration behavior of ants as the iterations are increasing.

At time $t = 0$ the ant starts from a factious symbol only used as a start node. At time $t = m \times n$ the ant has completed its solution building and the obtained keyboard can be evaluated. Figure 7 shows a small example of keyboard assignment.

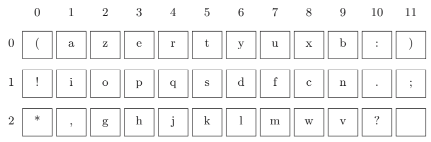


Fig. 7. Keyboard of size 12×3

4.2 Solution Evaluation

The quality $Q(\mathcal{K}, S)$ of an arrangement $\mathcal{K}$ is calculated according to a sequences of symbols $S = \{s[1], s[2], \ldots, s[|S|]\}$ of length $|S|$ and corresponds to the total length of moves that are necessary to typeset the sequence on a given keyboard $\mathcal{K}$.

$$Q(\mathcal{K}, S) = \frac{100}{|S|} \sum_{i=1}^{|S|-1} d_\varepsilon(s[i], s[i+1]) \tag{2}$$

where $d_\varepsilon(x, y)$ stands for the euclidean distance between keys x and y of coordinates $(x_1, x_2) \in \{0, \ldots, m-1\}^2$ and $(y_1, y_2) \in \{0, \ldots, n-1\}^2$ on the keyboard $\mathcal{K}$:

$$d_\varepsilon(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

4.3 Pheromones Update

Once all the ants have built their solution, each pheromone value is updated according to the following rule:

$$\tau_{i,j} \leftarrow (1 - \rho)\tau_{i,j} + \rho \sum_{k=1}^{A} \Delta_{i,j}^k \tag{3}$$

ρ is called evaporation coefficient and Δ is computed in order to favor good choices of ants:

$$\Delta_{i,j}^k = \begin{cases} \dfrac{\min\limits_{l \in \{1,\ldots,A\}} \{Q(\mathcal{K}_l, S)\}}{Q(\mathcal{K}_k, S) \times (\mathrm{Rank}(\mathcal{K}_k) + 1)} & \text{if symbol } j \text{ follows symbol } i \text{ in keyboard } \mathcal{K}_k \\ 0 & \text{else} \end{cases} \tag{4}$$

with A the number of ants. In this case, $\Delta_{i,j}^k$ is increased each time there is one ant with the symbol j after symbol i in its keyboard and proportionally to the relative quality of the keyboard and its rank in a decreasing order ofquality ($\text{Rank}(\mathcal{K}_{\text{best}}) = 1$). If no ant have decided to set j after i, the pheromone quantity is only decreased according to the parameter ρ. Finally, pheromones are kept inside bounds $[\tau_{\min}; \tau_{\max}]$.

5 Experimental Results

5.1 Experimental Settings

In order to study the algorithm performances, we have built a set of 12 different documents of 4 types (see table 1). For all these documents, the same character set is used:

```
azertyuiopmlkjhgfdsqwxcvbn,;:!.?&"'(-_)][$* 1234567890{}\n
```

and the same keyboard size is used: 20×3.

Table 1. 12 documents used as testbed

Name	Type	Lengths
S_1, S_2, S_3	blog extracts	$1564, 3269, 2370$
S_4, S_5, S_6	C programming code	$1943, 3495, 2934$
S_7, S_8, S_9	tales of Grimm extracts	$8802, 9425, 6193$
S_{10}, S_{11}, S_{12}	newspaper extracts	$4103, 2105, 4701$

Parameters of ants have been chosen as follows:

- pheromone bounds: $[\tau_{\min}; \tau_{\max}] = [0.1; 0.9]$,
- initial pheromone value: $\tau_{i,j}^0 = \tau_{\max}$ if $i \neq j$ and $\tau_{i,i}^0 = 0.0$,
- evaporation rate : $\rho = 0.01$,
- desirability $\eta_{i,j}$ is computed as the relative frequency of co-occurrence of symbols i and j in the considered text,
- number of ants = number of symbols to set on the keyboard (57 in our case),
- number of iteration : 500 (each ant builds 500 keyboards).

5.2 Parameter Study

We have introduced several measures to study ants'work at iteration T but we only focus in this paper on pheromone entropy:

$$\text{Ent}(T) = -\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{|\tau_{i,j} - \bar{\tau}|}{\max\limits_{u,v}\{\tau_{u,v}\} - \min\limits_{u,v}\{\tau_{u,v}\}} \log \frac{|\tau_{i,j} - \bar{\tau}|}{\max\limits_{u,v}\{\tau_{u,v}\} - \min\limits_{u,v}\{\tau_{u,v}\}} \qquad (5)$$

Figure 8 shows the evolution of keyboard quality, pheromone values and pheromone entropy for document S_1 (blog). Three cases of α are compared :

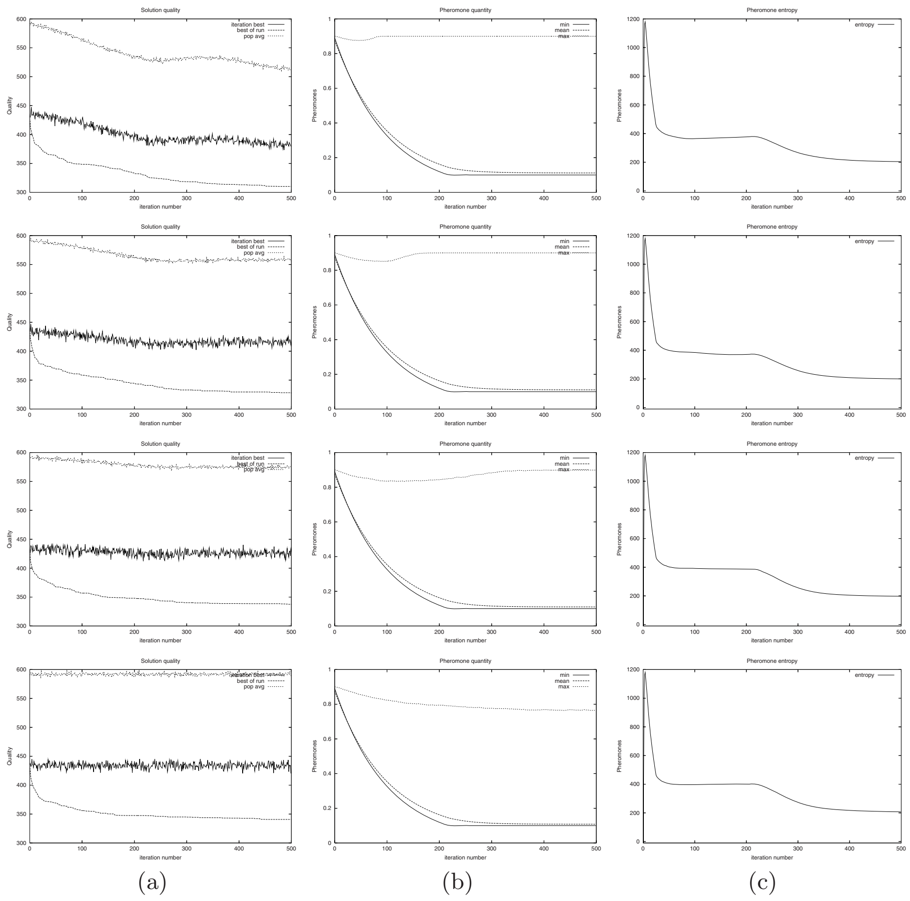


Fig. 8. Plots for document S_1 (blog): (a) evolution of keyboard quality (best of run, best of iteration, mean over the population), (b) pheromone values (max, min and mean over all the edges) and (c) pheromone entropy. First row: $\alpha = 2$, second row: $\alpha = 1$, third row: $\alpha = 0.5$ and fourth row: $\alpha = 0$.

$\alpha = 1$ (pheromones and desirability are of same importance in ant's probabilistic decision rule), $\alpha = 0.5$ (pheromones are twice less used than desirability) and $\alpha = 0$ (pheromones are not used!). Each plot is the average of 30 independent runs. Plots obtained for other documents are very similar. We can notice that the pheromone entropy curve (column (c)) is more or less similar for each value of α: pheromone entropy does only depend on the mean pheromone value which decreases similarly (when most of the edges have reach the minimum value of $\tau_{\min}$ around iteration $T = 220$, see column (b)). But, we can see that if pheromones are not used (last row, in this case the algorithm is then similar to a greedy algorithm restarted 500 times), the mean quality of the population and the best keyboard for a given iteration remain constant while for $\alpha = 2.0$ the quality of the whole population increases (with a slow down around iteration $T = 220$).

Table 2. Relative mean performances of the best keyboard found for each document (rows) over all 12 documents (columns). results are given in %.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
S_1	1.89	5.43	1.47	37.24	73.75	29.69	1.90	4.30	2.90	3.40	3.19	3.50
S_2	9.84	1.66	0.73	44.49	80.19	29.87	3.68	5.37	5.78	6.17	6.77	4.53
S_3	11.77	7.29	0.20	39.42	82.88	26.97	6.07	8.82	7.23	7.30	7.75	7.28
S_4	21.57	19.86	15.13	1.09	42.07	10.46	17.49	20.04	18.21	17.28	16.87	17.94
S_5	50.35	47.34	45.06	38.04	0.00	33.10	47.63	46.64	42.21	49.04	50.38	47.10
S_6	74.34	70.69	69.15	90.06	109.08	0.00	73.23	72.12	70.19	71.02	69.74	70.33
S_7	8.39	7.65	4.12	41.56	74.31	30.33	0.25	4.31	3.95	5.27	6.59	5.29
S_8	7.83	5.26	1.80	38.73	75.70	25.43	1.48	2.53	2.53	5.21	5.22	4.74
S_9	8.35	8.04	4.50	41.61	76.80	31.86	2.18	5.31	0.60	7.53	7.15	5.80
S_{10}	7.24	5.94	1.64	36.87	74.87	25.55	2.85	5.07	4.10	1.29	4.01	2.84
S_{11}	6.81	7.32	2.68	41.08	78.09	25.35	3.01	5.69	4.79	3.85	1.27	3.72
S_{12}	6.16	7.46	2.39	42.00	87.65	31.73	2.51	5.56	4.52	3.91	5.07	1.41

We can conclude that pheromones are useful for ants'progression and if we would like to obtain still better results, we should try to keep pheromone entropy at the first level (during the 220 first iterations). To do this, we can play with parameters ρ.

5.3 Experimental Comparison

For each document $\{S_1, \ldots, S_{12}\}$ and each of the 30 independent runs, we have obtained a best virtual keyboard. This keyboard is then used to compute its quality when typing the 12 documents. Values reported in table 2 give the mean relative performance of one keyboard over the 12 documents. We can notice that for a given "keyboard" (*i.e.* which is learned with the given document), the relative best typing performances are obtained for the given document and more widely for the same class of documents (for instance, learning with S_5 is always the best way to type in S_5). This confirms us that it seems to be useful to train the keyboard with representative documents of user's activity. We see clearly that keyboards built from C documents are much more efficient for typing C documents.

6 Conclusion

In this paper, we have presented a work which is taking advantage of artificial ant optimization:

– the first work on artificial ants and keyboards [16] was interesting and well conducted but it was missing a real need, and in our opinion, helping disabled persons is a good one. Moreover, as needs of handicapped persons are always different from each other and because their capabilities are often in evolution (positive or negative) our method will be for many persons and many times!

– preliminary results presented in this paper show that the algorithm has the needed characteristics to follow our goal: an adaptive behavior according to simple document different types,
– other properties of ant algorithms can be useful, for instance we can provide an online-version that can dynamically modify the keyboard according to every day use and then follows the user's evolution in its habits.

Many experiments will be added to this first study. For instance in order to compare artificial ants with other population based metaheuristics. Performances can surely be improved by introducing a local search step. We will also improve the fitness computation by introducing classical measures or user models used in the field of adapted input devices and interfaces [19].

The next step of this work would be to provide a virtual keyboard with the ant algorithm inside. Then, it will be possible to perform experiments with real users. As we have seen it in section 2, it will not be possible to ignore to include a prediction mechanism.

References

1. Chisholm, W., Vanderheiden, G., Jacobs, I.: CSS Techniques for Web Content Accessibility Guidelines 1.0 (Novembre 2000) (last visit on 10/11/2006), http://www.w3.org/TR/WCAG10-CSS-TECHS/
2. WAI: Web Accessibility Initiative (2006) (last visit on 06/12/2006), http://www.w3.org/WAI
3. Mormiche, P.: le groupe de projet HID: Le handicap en institution, le devenir des pensionnaires entre 1998 et 2000. (septembre 2001)
4. Lake Software: Click-n-type (2007) (last visit on 30/01/2007), http://www.lakefolks.org/cnt/fr-intro.htm
5. In'Tech INFO: Custom Virtual Keyboard (CVK) (2007) (last visit on 30/01/2007), http://www.cvk.fr/
6. Madentec: Screendoors 2000 (2007) (last visit on 30/01/2007), http://www.madentec.com/products/screendoors.php
7. AssistiveWare: Keystrokes (2007) (last visit on 30/01/2007), http://www.assistiveware.com/keystrokes.php
8. Handicap International: Clavicom (2007) (last visit on 30/01/2007), http://www.handicap-icom.asso.fr/adaptations/aides_techniques/clavicom.html
9. Bloorview Kids Rehab: Wivik (2007) (last visit on 30/01/2007), http://www.wivik.com/
10. Boissière, P., Dours, D.: Vitipi: Un système d'aide à l'écriture basé sur un principe d'autoapprentissage et adapté à tous les handicaps moteurs. In: IFRATH Handicap 2000, AACCESS Reprographie (Paris 15 - 16 Juin 2000), pp. 81–96
11. Maurel, D., Fouche, B., Briffault, S.: Handias: Aider la communication en facilitant la saisie rapide de textes. In: IFRATH Handicap 2000, (Paris 15 - 16 Juin 2000), pp. 87–92 (2000)
12. Raynal, M.: Keyglasses: des touches semi-transparentes pour optimiser la saise de texte. In: Interaction Homme Machine IHM 2005 (2005)

13. Schadle, I., Antoine, J.Y., Le Pévédic, B., Poirier, F.: Sibylettre: prédiction de lettre pour la communication assistée. In: Revue d'Interaction Homme-Machine, RIHM vol. 3 (2002)
14. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
15. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA, USA (2004)
16. Eggers, J., Feillet, D., Kehl, S., Wagner, M.O., Yannou, B.: Optimization of the keyboard arrangement problem using an Ant Colony algorithm. European Journal of Operational Research 148(3), 672–686 (2003)
17. Raynal, M., Vigouroux, N.: Genetic Algorithm to Generate Optimized Soft Keyboard. In: Conference for Human-Computer Interaction (CHI 2005), Portland, Oregon (April 27, 2005), pp. 1729–1732. ACM, New York (2005)
18. Delahaye, D., Puechmorel, S.: Air Traffic Controller Keyboard Optimization by Artificial Evolution. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 177–188. Springer, Heidelberg (2004)
19. Soukoref, R., MacKenzie, I.: Theorical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard. Behavior and Information Technology 14(6), 370–379 (1995)

Self-organization and Evolution Combined to Address the Vehicle Routing Problem

Jean-Charles Créput and Abderrafiaâ Koukam

Systems and Transportation Laboratory,
University of Technology of Belfort-Montbeliard, 90010 Belfort, France
{Jean-Charles.Creput,Abder.Koukam}@utbm.fr

Abstract. The paper deals with a self-organizing system in a evolutionary framework applied to the Euclidean Vehicle Routing Problem (VRP). Theoretically, self-organization is intended to allow adaptation to noisy data as well as to confer robustness according to demand fluctuation. Evolution through selection is intended to guide a population based search toward near-optimal solutions. To implement such principles to address the VRP, the approach uses the standard self-organizing map algorithm as a main operator embedded in a evolutionary loop. We evaluate the approach on standard benchmark problems and show that it performs better, with respect to solution quality and/or computation time, than other self-organizing neural networks to the VRP presented in the literature. As well, it substantially reduces the gap to some classical Operations Research heuristics.

Keywords: Neural network, Self-organizing map, Evolutionary algorithm, Vehicle routing problem.

1 Introduction

In this paper we are concerned with the Vehicle Routing Problem (VRP) [3]. The VRP is defined on a set $V = \{v_0, v_1, ..., v_N\}$ of vertices, where vertex v_0 is a depot at which are based m identical vehicles of capacity Q, while the remaining N vertices represent customers, also called requests or demands. A non-negative cost, or travel time, is defined for each edge $(v_i, v_j) \in V \times V$. Each customer has a non-negative demand q_i and a non-negative service time s_i. A vehicle route is a circuit on vertices. The VRP consists of designing a set of m vehicle routes of least total cost, each starting and ending at the depot, such that each customer is visited exactly once by a vehicle, the total demand of any route does not exceed Q, and the total duration of any route does not exceed a preset bound D. As it is the most often done in practice [7][20], we shall be concerned in this paper with the Euclidean VRP, where each vertex v_i has a location in the plane, and where the travel cost is given by the Euclidean distance $d(v_i, v_j)$ for each edge $(v_i, v_j) \in V \times V$. Then, the main objective of the problem is the total route length.

This problem is one of the most widely studied problems in combinatorial optimization. It has a central place for the determination of efficient routes in distribution management. The problem is NP-hard. Then, for large instances, using heuristics is encouraged in that they have statistical or empirical guaranty to find good solutions possibly for large scale problems with several hundreds of customers. For example, ones of the most powerful Operations Research (OR) heuristics for the VRP, referred in the extensive surveys [7][14][20], are based on metaheuristic frameworks as the Tabu Search [6][28][32], simulated annealing, and population based methods, such as evolutionary algorithms [22][25], adaptive memory [31] and ant algorithms [26]. Other methods can hybridize several metaheuristics principles, as for example the very powerful active guided local search [22], which is maybe the overall winner approach considering both quality solution and computation time.

Here, we focus on the Euclidean VRP and propose a Euclidean solving approach. The method presented in this paper takes its origin in neural networks (NN) visual patterns that evolve and distort in the plane according to an underlying data distribution. The neural network considered in this paper is the self-organizing map (SOM) [19], which is often presented as a non supervised learning procedure performing a non parametric regression that reflects topological information of the input data. The underlying concept, that we call adaptive meshing, lets envisage application to many spatially distributed problems, as radio-mobile and terrestrial transportation dimensioning, clustering k-median, and unified clustering and routing problems [8][9][10][11][12]. By preserving density and topology of a data distribution, SOM allows positioning of facilities in accordance to the demand, respecting the inter-component network architecture. As well, its $O(N)$ spatial complexity theoretically allows application to very large instances. Furthermore, continuous visual feedback during simulations is naturally allowed.

To solve the combinatorial optimization problem, we present an evolutionary framework which incorporates self-organizing maps as internal operators. The approach is called memetic SOM by reference to memetic algorithms [24], which are hybrid evolutionary algorithms (EA) incorporating a neighborhood search. Furthermore, since the communication times at the level of selection is relatively small, the long running times of independent SOM processes favor parallel execution of the method on standard computer systems.

In the literature, many applications of neural networks have addressed the traveling salesman problem (TSP). For more information on this subject, we refer the reader to the extensive survey of Cochrane and Beasley [5]. However, extending SOM to the more complex vehicle routing problem remains a difficult task. Few works were carried out trying to extend SOM, or elastic nets, to the VRP. As far as we know, the most recent approaches are [15][18][21][23][29][30][33]. They are generally based on a complex modification of the internal learning law, altered by problem dependant penalties. Here, to apply SOM to the VRP and to improve performances as well, the standard SOM execution interleaves with other processes, or operators, following the evolutionary method. The standard SOM is a main operator embedded into an EA and combined with other greedy insertion operators, fitness evaluation and selection operators.

Evaluation of the proposed approach is performed against neural networks and some of the recent Operations Research heuristics presented in [7]. Mainly, we will try to show that the memetic SOM yields a substantial gain of accuracy in comparison to the previous SOM based approaches. Considering OR heuristics, memetic SOM does not compete with the most powerful ones, which beneficiate of the considerable effort spent over more than thirty years, but we claim that it substantially reduces the gap.

The paper is organized as follows. Section 2 presents the memetic SOM approach. Section 3 reports experiments carried out on the capacitated VRP, as well as on its time duration version. Evaluations against SOM based approaches and OR heuristics are performed. Finally, the last section is devoted to the conclusion and further research.

2 Evolutionary Algorithm Embedding SOM

2.1 Method Principle

To illustrate the "philosophy" of the SOM behavior, an example of its application to the TSP is illustrated in Fig. 1 on the bier127 instance from TSPLIB [27], at different steps of a long simulation run. The TSP can be seen as a VRP, with no capacity and no time duration constraint, and using a single vehicle. The example shows a tour construction using a ring network, which is a planar graph with a fixed number of vertices having a ring shape. The network dispatches its vertices among cities, or customers (dots in the figure), in a massively parallel manner. At the beginning, the local moves in the plane are performed with a great intensity in order to let the ring deploy toward cities, starting from scratch (a). Then, the intensity of moves slightly decreases in order to progressively freeze the vertices near cities (b-c). At a final step, customers have just to be assigned to their nearest vertex in the ring in order to generate a final tour ordering.

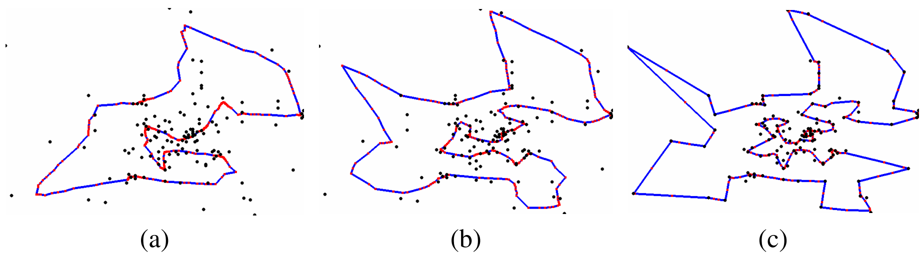


Fig. 1. A TSP tour construction by SOM using the TSPLIB bier127 instance

This massive parallelism, using an intermediate structure in the plane, differentiates the approach from classical Operations Research heuristics. Such methods operate (sequentially) on a graph, where the vertices stand for the customers. They generally model routes by a customer ordering, and apply local search operators performing customer swaps and/or arc exchanges between routes [7][14][20].

Here, routes are defined by an ordering of cluster centers, or neurons, moving in the plane, and having to be assigned to customers in a subsequent step. We think that separating the transportation network from the underlying customer demands has several positive aspects. One of the goals is to give the natural potential to deal with noisy or incomplete data, and with fluctuating demand. As well, we think that this independent structure more closely models usual transportation networks, which are physical interconnected infrastructures more or less adapted to the many potential customers. A characteristic of the method is that non standard solutions are admitted during the course of the optimization process. This allows to deal with combined clustering and routing problems, in a unified way. For example, [10][11][12] present a combination of the Euclidean k-median problem [1] and classical VRP. It consists of positioning bus stops according to customer locations (k-median problem) and generating vehicle routes among bus-stops (VRP). Bus-stops define clusters where customers are grouped and where they have to walk to take the bus.

To extend SOM applicability to complex problems, such as the VRP, it becomes an operator embedded into an evolutionary algorithm, yielding to what we call the memetic SOM algorithm. An advantage is that operators can be designed independently and then combined. The term used to describe and qualify our method focuses on the analogy with the memetic algorithm [24] which incorporates a local search into a standard evolutionary loop. The SOM algorithm stands for local search. It is used here as a long run process applied to a population of solutions, and interrupted during its progress by application of evolutionary operators. Here, no recombination, nor crossover, operators are considered. The method is a descendant approach just like evolution strategies [2] and memetic algorithms, and at the contrary of genetic algorithms [16] that are ascendant methods (solutions are constructed) centered around crossover. By following the two kind of spatial and natural metaphors of self-organization and evolution, and by using standard algorithm structures, such as SOM and EA, the goal is also to address the build of a simple and effective distributed method for the solving of hard combinatorial optimization problems.

2.2 The Kohonen's Self-Organizing Map

The standard self-organizing map [19] algorithm operates on a non directed graph $G = (A, E)$, called the network, where each vertex $n \in A$ is a neuron having a location $w_n = (x, y)$ in the plane. The set of neurons A is provided with the d_G induced canonical metric $d_G(n,n') = 1$ if and only if $(n,n') \in E$, and with the usual Euclidean distance $d(n, n')$.

The training procedure applies a given number of iterations *niter* to a graph network, the vertex coordinates of which being randomly initialized into an area delimiting the data set. Here, the data set is the set of demands, or customers. Each iteration follows four basic steps. At each iteration t, a point $p(t) \in \Re^2$ is randomly extracted from the data set (extraction step). Then, a competition between neurons against the input point $p(t)$ is performed to select the winner neuron n^* (competition step). Usually, it is the nearest neuron to $p(t)$.

$$w_n(t+1) = w_n(t) + \alpha(t).h_t(n^*, n).\left(p - w_n(t)\right) \tag{1}$$

Then, the learning law (1) (triggering step) is applied to n^* and to all neurons within a finite neighborhood of n^* of radius σ_t, in the sense of the topological distance d_G, using learning rate $\alpha(t)$ and function profile h_t. The function profile is given by the Gaussian in (2).

$$h_t\left(n^*,n\right)=\exp\left(-d_G\left(n^*,n\right)^2\Big/\sigma_t^2\right) \tag{2}$$

Finally, the learning rate $\alpha(t)$ and radius σ_t slowly decrease as geometric functions of time (decreasing step). To perform a decreasing run within t_{max} iterations, at each iteration t coefficients $\alpha(t)$ and σ_t are multiplied by $\exp\left(\ln\left(x_{final}\big/x_{init}\right)\Big/t_{\max}\right)$, with respectively $x = \alpha$ and $x = \sigma$, x_{init} and x_{final} being respectively the values at starting and final iteration. In our evolutionary algorithm, a SOM simulation becomes an operator specified by its running parameters $\left(\alpha_{init},\alpha_{final},\sigma_{init},\sigma_{final},t_{\max}\right)$.

2.3 Evolutionary Loop and Operators

A construction loop as well as an improvement loop are instantiated based on the following generic memetic loop structure, where the parameter setting has been determined empirically after a preliminary round of experiments :

```
Initialize population with Pop=50 individuals.
Gen = 0
While not Gen=N/2 generations are performed /* Gen=5N for long runs */.
        1. In construction mode only, apply a standard SOM operator, with parameters ($\alpha_{init}$, $\alpha_{final}$, $\sigma_{init}$, $\sigma_{final}$, $t_{\max}$)= (0.5, 0.5, 2×N/m, 4, Gen×niter), to each individual in population separately, performing niter=N/4 iterations by individual at each generation.
        2. In improvement mode only, apply a standard SOM operator, with parameters ($\alpha_{init}$, $\alpha_{final}$, $\sigma_{init}$, $\sigma_{final}$, $t_{\max}$)= (0.9, 0.5, 2×N/m, 0.5N/m, Gen×niter), to each individual in population separately, performing niter=1 iteration by individual at each generation.
        3. In improvement mode only, apply derived SOM operator, denoted SOMVRP, with parameters ($\alpha_{init}$, $\alpha_{final}$, $\sigma_{init}$, $\sigma_{final}$, $t_{\max}$)= (0.5, 0.5, 10, 4, Gen×niter), to each individual in population separately, performing niter=N/m iterations by individual at each generation.
        4. Apply mapping operator MAPPING to each individual in population to assign each demand to a closest vertex, then move vertices to demand locations.
        5. Apply fitness evaluation operator FITNESS to each individual in population.
        6. Save the best individual encountered.
        7. Apply selection operator SELECT.
        8. Apply elitist selection operator SELECT_ELIT.
        9. Apply derived operator SOMDVRP, to each individual in population separately to perform greedy insertion moves to the residual demands. */
        8. Gen = Gen +1
End while.
Report best individual encountered.
```

The memetic loop applies a set of operators to a population of *Pop* individuals, at each iteration (called a generation). A loop executes a fixed number of generations *Gen*, depending on the problem size *N*. The number of individuals is constant. One individual encapsulates exactly one solution, that is, a set of *m* routes, called the network. Each route is represented by a planar graph having a ring structure with $5 \times N/m$ vertices. To each route corresponds exactly one vehicle. Each ring has a vertex fixed at the depot location. Other vertices are free to move anywhere in the plane. Vertices are the variables which have to be located on, and assigned to, customer locations in order to define the different tour orderings which stand for the VRP solution. The number of vertices by vehicle corresponds to the maximum of customers a vehicle can visit. It has been adjusted empirically to allow a good compromise between number of customers visited, equilibration of route lengths and computation speed.

The construction loop starts its execution with solutions having randomly generated vertex coordinates into a rectangle area containing demands. The improvement loop starts with the single best previously constructed solution, which is duplicated in a new population. Mainly, the behavior consists of applying a SOM process and interrupting its progress, at each generation, by application of a mapping operator which generates a VRP solution by projecting vertices to customer locations. Other operators have a complementary role, trying to direct the search toward better quality solutions. The construction loop is responsible for creating an initial solution from scratch, whereas the improvement loop is responsible of local improvements on solutions. It follows that the SOM process embedded in the former loop performs a greater number of iterations by generation, using a larger neighborhood. The improvement loop performs very few and punctual moves at each generation.

The two loops are managed by a master loop controlling restart executions, from random solutions at each time. A complete run executes construction followed by improvement until all customers are inserted and all constraints satisfied, at least *NExec* times, and at most a predefined large number of times. For fast runs, we take *NExec* = 1 and *Gen* = *N*/2. For long runs, we take *NExec* = 5, *Gen* = 5*N*, the improvement phase being also restarted as soon as the fitness is improved.

The construction and improvement phases are illustrated in Fig. 2 on the c5 instance with 200 customers of the publicly available Christofides, Mingozzi, and Toth (CMT) [3] benchmark, showing the visual pattern moving in the plane. The construction phase is illustrated in (a-d). Two consecutive pictures show the network, at a given generation, as distorted by the SOM operator followed immediately by the admissible, or near admissible, solution generated by the mapping operator. This mapping operator creates a VRP solution by projecting nearest vertices to each customer. (a-b) present the network at the beginning of construction and (c-d) several generations later. In (e-f) is shown the network at different steps of the improvement phase, illustrating the local perturbations responsible for local improvements.

Details of operators are the followings:

1) Self-organizing map operator. It is the standard SOM applied to the graph network. It is denoted by its name and its internal parameters, as $SOM\left(\alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}, t_{\max}\right)$. One or more instances of the operator, with their own parameter values, can be combined. A SOM operator is executed

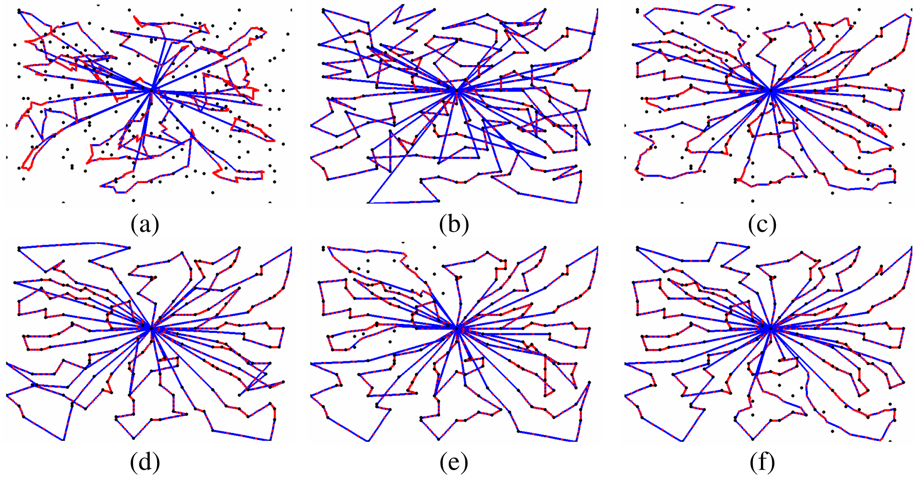


Fig. 2. Two phases algorithm with the CMT c5 instance. (a)-(d) Construction phase. (e)-(f) Improvement phase.

performing *niter* basic iterations by individual, at each generation. Parameter $t_{\max}$ is the number of iterations defining a long decreasing run performed in the stated generation number *Gen*, for each individual. Other parameters define the initial and final intensity and neighborhood for the learning law. The operator can be used to deploy the network toward customers in construction phase, or to introduce punctual moves to exit from local minima in improvement phase.

2) SOM derived operators. Two operators are derived from the SOM algorithm structure for dealing with the VRP. The first operator, denoted *SOMVRP,* is a standard SOM restricted to be applied on a randomly chosen vehicle, using customers already inserted into the vehicle/route. It helps eliminate remaining crossing edges in routes. While capacity constraint is greedily tackled by the mapping/assignment operator below, the second operator, denoted *SOMDVRP*, deals with the time duration constraint specifically. It performs few insertion moves of customers, not already assigned, to a vehicle vertex with least route time increase.

3) Mapping/assignment operator. This operator, denoted *MAPPING*, generates a VRP solution by inserting customers into routes and modifies the shape of the network accordingly. The operator greedily maps customers to their nearest vertex, considering vertices not already assigned, for which vehicle capacity constraint is satisfied. Then, the operator moves the vertices to the location of their assigned customer (if exist) and dispatches regularly (by translation) other vertices along edges formed by two consecutive customers in a route.

4) Fitness operator, denoted *FITNESS*. Once the assignment of customers to routes has been performed, this operator evaluates a scalar fitness value for each individual that has to be maximized and which is used by the selection operator. The value returned is $fitness = sat - 10^{-5} \times length,$ where *sat* is the number of customers that are successfully assigned to routes, and *length* is the length of the routes defined by the

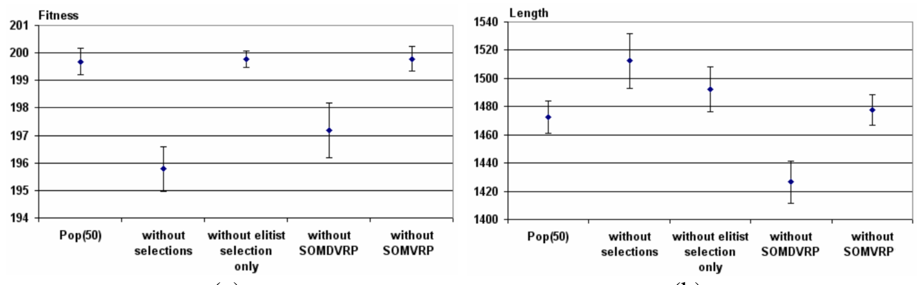
ordering of such customers mapped along the rings. Admissible solutions are the ones for which *sat* = *N*, *N* being the number of customers.

5) Selection operators. Based on fitness maximization, the operator denoted *SELECT* replaces *Pop*/5 worst individuals, which have the lowest fitness values in the population, by the same number of bests individuals, which have the highest fitness values in the population. An elitist version *SELECT_ELIT* replaces the *Pop*/10 worst individuals by the single best individual encountered during the current run.

3 Computational Results

3.1 Influence of the Main Algorithmic Components

Using the parameter settings given in the previous section, we apply the construction and improvement loops in sequence, one time each, and study the influence of the main algorithmic components, for a fixed population size of 50 individuals. The generation numbers *GenC* and *GenI* are set to *N*. The tests are done using the c10 instance of the Christofides, Mingozzi, and Toth (CMT) [3] benchmark, with 200 customers and time duration constraint, performing 10 runs by test with a chosen component being removed from the algorithm. Fig. 3(a) presents the mean fitness values obtained at the last generation within 95% confidence intervals. Fig. 3(b) presents the mean route lengths, as well in 95% confidence intervals.



(a) (b)

Fig. 3. Performances of memetic SOM removing algorithmic components. (a) Fitness values in 95 % confidence intervals. (b) Route lengths in 95 % confidence intervals.

Clearly, as shown in Fig. 3(a), selections operators have a great influence on the algorithmic performance. Without any selection operator, the algorithm behaves like executing 50 independent runs on a single individual. In that case, only 196 demands, on a total of 200, are satisfied on average. As well, the *SOMDVRP* operator, responsible for random insertions according to time duration constraint, plays an important role. Removing this component yields to solution with less than 200 customers successfully inserted, thus to non admissible solutions.

Removing elitist selection only, or *SOMVRP* operator responsible for single route improvement, yields to fitness values overlapping with the standard case *Pop* = 50. Then, to decide whether these two operators play a significant role, we need to look at the secondary fitness component, which is the total route length, in fact the main VRP objective to minimize. Fig. 3(b) clearly shows that removing the elitist selection has a non negligible impact on route length minimization. Removing the *SOMVRP* however has a weaker impact on this value, but this can be explained by the particular test case used, where time duration constraint is predominant. The standard Pop = 50 case wins in all cases, considering both fitness and length.

3.2 Comparative Evaluation

Evaluation of the memetic SOM approach is done against neural network algorithms and against some recent Operations Research heuristics. In the former case, we compare memetic SOM to the three representative approaches of Ghaziri [15], Modares et al. [23] and Schwardt and Dethloff [30]. Only these authors have made significant use of the publicly available Christofides, Mingozzi, and Toth (CMT) [3] test problems. Other neural network versions [18][21][29][33] are quite algorithmically similar or clearly worse performing. Only Ghaziri [15] addresses the time-duration version of VRP and solves almost all the corresponding CMT test cases. We also evaluate the approach against the Clarke and Wright construction heuristic [4], the Unified Tabu Search Algorithm (UTSA) [6], the Active Guided Evolution Strategy (AGES) [22], and the Very Large Neighborhood Search (VLNS) [13] approach. Considering the survey of Cordeau et al. [7], the UTSA approach has displayed good performances on solution quality and is considered very simple and flexible, but time consuming. Other recent OR heuristics present similar accuracy within less computation time. An exception is the AGES approach, which displays a very high performance considering both solution quality and computation time. At the opposite, VLNS appears to be the less performing OR heuristic presented in [7]. Results on the large test problems of Golden et al. [17], with up to 483 customers, will be mentioned briefly.

Numerical results on CMT instances are given in Table 1. The fourteen CMT instances of the capacitated VRP are composed of two subsets containing seven instances of each VRP types, with (D subset) or without time duration constraint (C subset). The first column “Name-size-veh” indicates the name, size and number of vehicles of the instance. Second column indicates the best known values that were obtained initially for a large part by [28] through a long run. The memetic SOM results are averaged on a basis of 10 runs. We used the two configurations “fast” and “long” of the algorithm, to adjust computation time. Approaches are mentioned in Table 1 using the name of the method and/or the author names and date of publication. For each method, the percentage deviation to the best known value of the mean solution is reported in column “%PDM” and the average computation time (if exist) in column “Min” in minutes. Some results that are best values over many runs are reported in column “Best”. Results in Table 1 show the improvement carried out by memetic SOM against earlier neural network approaches. Accuracy is substantially improved since the average deviation to the best known value is reduced from roughly 5 % to 3.39 % on average for fast runs, and to 1.20 % for long runs.

Here, all CMT test cases are addressed successfully. Modares et al. [23] and Schwardt and Dethloff [30] do not report computation time. Furthermore, they do not address the time duration version of the problem. Only Ghaziri [15] deals with it. Taking care of the rough approximation performed when comparing computation times, memetic SOM clearly improves solution quality. We also report in Table 1 results obtained by the Clarke and Wright construction heuristic [4], available at VRP Web http://neo.lcc.uma.es/radi-aeb/WebVRP/. Neural networks perform better than the construction heuristic, for more computation time. Finally, results obtained by the UTSA approach [6] are given in the two last columns. They illustrate the gap to such a recent Operations Research heuristic. With roughly 30-40 % computation time more on a similar computer, UTSA yields 0.56 % of average deviation, whereas memetic SOM 1.20 %. At the opposite, referring to [7], the best performing, but complicated, AGES [22] yields 0.03 % deviation to best known in 7.72 minutes on a Pentium IV (2000 MHz), and 0.07 % deviation in 0.27 minutes.

Considering Operations Research heuristics, we also used the large test problems of Golden et al. [17], with up to 483 customers. On all the tests, memetic SOM yields 2.70 % deviation in 39 minutes on a AMD Athlon (2000 MHz) computer, whereas UTSA yields 1.45 % in 56 minutes on a Pentium IV (2000 MHz), using results reported in [7]. Memetic SOM appears to be more accurate on instances with time duration constraint. In that case, the average deviation obtained was 1.96 % in 44 minutes, whereas UTSA yields 1.60 % in 38 minutes. On such instances, the proposed memetic SOM performs better than the VLNS [13] heuristic. It yields 3.45 % of average deviation in 10 minutes, whereas VLNS yields 3,76 % in 22 minutes normalized to the same computer.

Table 1. Numerical results for the CMT instances

		memetic SOM (fast)		memetic SOM (long)		Ghaziri (1996)		Modares et al. (1999)	Schwardt and Dethloff (2005)	Clarke and Wright	UTSA (Cordeau et al. 2001)	
Name-size-veh	Best	%PDM	Min[a]	%PDM	Min[a]	%PDM	Min[b]	%Best	%PDM	%PDM	%PDM	Min[c]
c1-50-5	524.61	2.37	0.03	0.00	2.19	2.78	0.90	2.36	1.43	11.44	0.00	2.32
c2-75-10	835.26	5.22	0.07	1.16	4.93			4.88	17.02	7.78	0.00	14.78
c3-100-8	826.14	1.52	0.11	0.15	8.17	8.14	6.50	4.46	3.78	7.35	0.00	11.67
c12-100-10	819.56	0.76	0.17	0.74	8.63	0.68	1.70		1.15	1.70	0.00	9.02
c11-120-7	1042.11	8.41	0.20	0.99	21.85	5.79	4.20	2.29	7.33	2.78	3.01	11.67
c4-150-12	1028.42	3.08	0.26	1.40	20.42	5.47	13.20	5.21	5.67	10.21	0.41	26.66
c5-199-17	1291.26	5.14	0.44	2.21	42.83	8.53	23.20	7.34	6.83	8.09	1.90	57.68
Average C		3.79	0.18	1.29	15.57	5.23	8.28	4.42	6.17	7.05	0.76	19.11
c6-50-6	555.43	1.46	0.03	0.22	2.10	1.06	4.30			11.34	0.00	3.03
c7-75-11	909.68	2.69	1.08	0.35	7.84					7.23	0.00	7.41
c8-100-9	865.94	1.82	0.12	0.00	8.62	3.28	18.40			12.47	0.00	10.93
c14-100-11	866.37	0.90	0.13	0.00	9.51	1.55	8.50			1.08	0.00	10.53
c13-120-11	1541.14	5.20	0.31	0.41	19.02	4.37	31.30			3.61	0.53	21.00
c9-150-14	1162.55	4.03	0.51	1.46	26.94	8.73	27.20			10.76	0.46	51.66
c10-199-18	1395.85	4.84	0.94	2.53	64.18	13.22	52.40			10.23	1.50	106.28
Average D		2.99	0.44	1.11	19.74	5.37	23.68			8.10	0.36	30.12
Average all		3.39	0.31	1.20	17.66	5.30	15.98			7.58	0.56	24.62

[a] Time per run in AMD Athlon (2000 MHz) minutes, Java program.
[b] Time per run in Sun Sparc 2 workstation minutes.
[c] Time per run in Pentium IV (2000 MHz) minutes.

4 Conclusion

By combining self-organization in evolution, we presented an approach that extends and improves neural networks applied to the VRP. Operators have a similar structure based on closest point findings and simple moves performed in the plane. They can be interpreted as performing parallel and massive insertions, simulating the behavior of spatially distributed agents which interact continuously, having localized and limited abilities. The evolutionary framework adds another level of parallel computation and direct the search toward problem goals. Application within a dynamic and stochastic context is now a question to be addressed. Exploiting the natural parallelism of the approach for multi-processor implantations is also a key point to address.

References

[1] Arora, S.: Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. Journal of the ACM 45(5), 753–782 (1998)

[2] Bäck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: 4th Int. Conf. on Genetic Algorithms, La Jolla, CA (1991)

[3] Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., et al. (eds.) Combinatorial Optimization, pp. 315–338. Wiley, Chichester (1979)

[4] Clarke, G., Wright, J.W.: Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research 12, 568–581 (1964)

[5] Cochrane, E.M., Beasley, J.E.: The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem. Neural Networks 16, 1499–1525 (2003)

[6] Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 52, 928–936 (2001)

[7] Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S.: New Heuristics for the Vehicle Routing Problem. In: Langevin, A., Riopel, D. (eds.) Logistics Systems: Design and Optimization, pp. 279–297. Springer, New York (2005)

[8] Créput, J.C., Koukam, A., Lissajoux, T., Caminada, A.: Automatic Mesh Generation for Mobile Network Dimensioning using Evolutionary Approach. IEEE Transactions on Evolutionary Computation 9(1), 18–30 (2005)

[9] Créput, J.C., Koukam, A.: Local search study of honeycomb clustering problem for cellular planning. International Journal of Mobile Network Design and Innovation 1(2), 153–160 (2006)

[10] Créput, J.C., Koukam, A.: Interactive Meshing for the Design and Optimization of Bus Transportation Networks. Journal of Transportation Engineering 133(9), 529–538 (2007)

[11] Créput, J.C., Koukam, A.: Transport Clustering and Routing as a Visual Meshing Process. Journal of Information and Optimization Sciences, Taru Publications (2007) (in press)

[12] Créput, J.C., Koukam, A., Hajjam, A.: Self-Organizing Maps in Evolutionary Approach for the Vehicle Routing Problem with Time Windows. International Journal of Computer Science and Network Security 7(1), 103–110 (2007)

[13] Ergun, Ö., Orlin, J.B., Steele-Feldman, A.: Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem. MIT Sloan Working Paper No. 4393-02, USA (2003)

[14] Gendreau, M., Laporte, G., Potvin, J.-Y.: Metaheuristics for the capacitated VRP. In: Toth, P., Vigo, D. (eds.) The vehicle routing problem, pp. 129–154. SIAM, Philadelphia (2002)

[15] Ghaziri, H.: Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem. In: Osman, I.H., Kelly, J.P. (eds.) Meta-Heuristics: Theory & Applications, pp. 651–660. Kluwer, Boston (1996)

[16] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1994)

[17] Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I.M.: Metaheuristics in vehicle routing. In: Crainic, T.G., Laporte, G. (eds.) Fleet Management and Logistics, pp. 33–56. Kluwer, Boston (1998)

[18] Gomes, L.C.T., Von Zuben, F.J.A.: Vehicle Routing Based on Self-Organization with and without Fuzzy Inference. Proc. of the IEEE International Conference on Fuzzy Systems 2, 1310–1315 (2002)

[19] Kohonen, T.: Self-Organization Maps and associative memory, 3rd edn. Springer, Berlin (2001)

[20] Laporte, G., Gendreau, M., Potvin, J.Y., Semet, F.: Classical and Modern Heuristics for the vehicle routing problem. International Transaction in Operational Research 7, 285–300 (2000)

[21] Matsuyama, Y.: Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In: Proc. of the International Joint Conference on Neural Networks, Seatle, WA, pp. 385–390 (1991)

[22] Mester, D., Bräysy, O.: Active Guided Evolution Strategies for Large Scale Vehicle Routing Problems with Time Windows. Computers & Operations Research 32, 1593–1614 (2005)

[23] Modares, A., Somhom, S., Enkawa, T.: A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research 6, 591–606 (1999)

[24] Moscato, P.: Memetic Algorithms: A Short Introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, McGraw Hill, New York (1999)

[25] Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31, 1985–2002 (2004)

[26] Reimann, M., Doerner, K., Hartl, R.F.: D-ants: savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research 31, 563–591 (2004)

[27] Reinelt, G.: TSPLIB-A traveling salesman problem library. ORSA Journal on Computing 3, 376–384 (1991)

[28] Rochat, Y., Taillard, E.D.: Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167 (1995)

[29] Schumann, M., Retzko, R.: Self-organizing maps for vehicle routing problems minimizing an explicit cost function. In: Proc. of the International Conference on Artificial Neural Networks, Paris, pp. 401–406 (1995)

[30] Schwardt, M., Dethloff, J.: Solving a continuous location-routing problem by use of a self-organizing map. Int J Physical Distribution & Logistics Management 35(6), 390–408 (2005)

[31] Taillard, E.D., Gambardella, M.L., Gendreau, M., Potvin, J.Y.: Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research 135, 1–16 (2001)

[32] Toth, P., Vigo, D.: The granular tabu search and its application to the vehicle routing problem. INFORMS Journal on Computing 15, 333–348 (2003)

[33] Vakhutinsky, A.I., Golden, B.L.: Solving vehicle routing problems using elastic net. In: IEEE International Conference on Neural Network, pp. 4535–4540 (1994)

An Evolutionary Algorithm for the Block Stacking Problem

Tim Hohm, Matthias Egli, Samuel Gaehwiler, Stefan Bleuler, Jonathan Feller, Damian Frick, Richard Huber, Mathias Karlsson, Reto Lingenhag, Thomas Ruetimann, Tom Sasse, Thomas Steiner, Janine Stocker, and Eckart Zitzler*

Computer Engineering and Networks Laboratory (TIK), ETH Zurich
{hohm,bleuler,zitzler}@tik.ee.ethz.ch,
http://www.tik.ee.ethz.ch/sop/

Abstract. How has a stack of n blocks to be arranged in order to maximize its overhang over a table edge while being stable? This question can be seen as an example application for applied statics and at the same time leads to a challenging optimization problem that was discussed recently in two theoretical studies.

Here, we address this problem by designing an evolutionary algorithm; the proposed method is applied to two instances of the block stacking problem, maximizing the overhang for 20 and 50 block stacks. The study demonstrates that the stacking problem is worthwhile to be investigated in the context of randomized search algorithms: it represents an abstract, but still demanding instance of many real-world applications. Furthermore, the proposed algorithm may become useful in empirically testing the tightness of theoretical upper bounds proposed for this problem.

1 Introduction

What is the largest overhang beyond the edge of a table that can be reached with a stack of n blocks (see Fig. 1)? A question that can be reformulated into the following optimization problem: Find a stable stack consisting of n blocks with a maximal overhang beyond the edge of a table. This is an example system to demonstrate the principles of static equilibrium that was discussed in two recent theoretical studies by Hall [3] and Paterson and Zwick [6].

Although this problem is mentioned in several engineering mechanics textbooks throughout the years [7,4] as well as in mathematics [1,5], until now optimal solutions are only known for strongly restricted variants. Nevertheless construction schemes and upper bounds for some scenarios are available [3,6]. Up to now, there exists no algorithm generating stacks where each block is explicitly represented; both studies, Hall's and Paterson and Zwick's, present algorithms involving implicit representations for parts of the stacks by introducing additional weights.

* This study arose from a student project within the Bachelor's program in Information Technology and Electrical Engineering at ETH Zurich that was supervised by Stefan Bleuler, Tim Hohm and Eckart Zitzler.

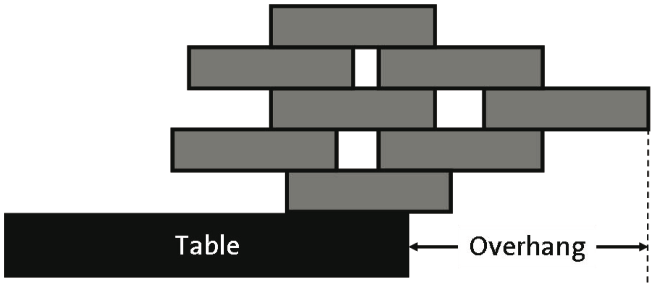


Fig. 1. An example Stack and its overhang over a table edge

Therefore developing a technique generating fully represented stack provides an interesting complement to the theoretical studies, empirically substantiating the proposed bounds. In addition, following Hall, the described problem *“... could pose a worthy test for general optimization algorithms.”* (page 1115 in [3]). This problem is an interesting addition to the set of test problems for stochastic optimization techniques like evolutionary algorithms (EAs) and expands the set of representatives of stacking problems often considered in EA studies [2].

Therefore, taking up Hall's suggestion in the following, we describe an EA designed to work on the described problem, addressing the questions of (i) how to represent the problem and how to design appropriate variation operators such that an EA can explore the space of stack configurations effectively, and (ii) what overhang can be reached by an EA - regarding a 20 block and a 50 block setup.

2 Related Work

The block stacking problem has a long history and was mentioned already in the 19th century [8] recurring from then on regularly in textbooks, providing only limited information on how a general optimal solution could look like. Recently, two theoretical studies examined the problem in more detail. In the first study, Hall [3] investigated the influence of different restrictions of the problem on the achievable overhang. He showed that the widely believed to be optimal overhang D_n for a stack of n blocks of

$$D_n = \frac{1}{2}\sum_{i=1}^{n}\frac{1}{i} \tag{1}$$

only holds for the most restricted variant: a scenario considering only vertical forces on the blocks and further demanding that all blocks are lying one-on-one. Non-intuitively, since Eq. 1 diverges for $n \rightarrow \infty$, the overhang for stacks following these restrictions can already reach an infinite overhang. Additionally he tested two less restricted variants, one where as well only vertical forces are considered but more than one block is allowed to rest on top of another and one extending the former scenario by considering friction as vertical contribution. For both less restricted settings better overhangs could be achieved.

Taking up the work by Hall and focusing on stacks of where more than one block is allowed per level and only vertical forces are considered, in a succession study Paterson and Zwick [6] further formalized the problem and introduced upper bounds, whereas their tightest bounds remain unproven. Further on they propose construction schemes for a slightly varied problem where they only consider a set of blocks producing the actual overhang and the remaining blocks are only implicitly represented by point forces applied from above. Using this variated problem (loaded stacks), empirical results for different numbers of blocks are presented.

Thereby these two studies provide formalizations and theoretical background for the considered problem, paving the way to consider the described problem as a demanding test problem for general optimization techniques. In addition, the tightest proposed bounds for optimal overhangs are given for loaded stacks only therefore posing the questions what an algorithm designing stacks with optimal overhang looks like and if the bounds can be reached.

3 The Block Stacking Problem

The problem considered here is to find a stack of n blocks producing a maximal overhang over the edge of a table while being stable, see Fig. 1. In this study the two dimensional block stacking problem is investigated for which it is assumed that:

- all blocks are rectangular,
- all blocks are of same size,
- all blocks are rigid,
- all blocks are perfectly smooth (no friction between blocks),
- all blocks are of equal and same density,
- all blocks have to lie on their long edge.

Further on, it is assumed that the table on which the stacks are built covers the third quadrant $(x, y \leq 0)$ of a Cartesian coordinate system in which the origin marks the upper right corner of the table.

In a stack built from such blocks, contacting blocks exert forces onto each other. These forces can be summed up to a single resulting force acting at one point in the contact interval. Resulting from the assumption that there is no friction between blocks, no horizontal forces but only vertical forces are present and since their is no drag between blocks, all forces F exerted have to be non-negative ($F \geq 0$). Following Newton's third law of motion, for each force F_A exerted by a block A on a block B, there has to be a counterforce $F_B = -F_A$ exerted from block B onto block A.

Now, for a stack to be stable it is necessary that all blocks in this stack are in equilibrium—a condition met if all forces exerted by blocks lying on top of a considered block C (plus the weight force F_W of C) and the moments imposed by these forces are evened out by counter forces and moments exerted by blocks lying below C. For the example stack shown in Fig. 2, for the middle light gray

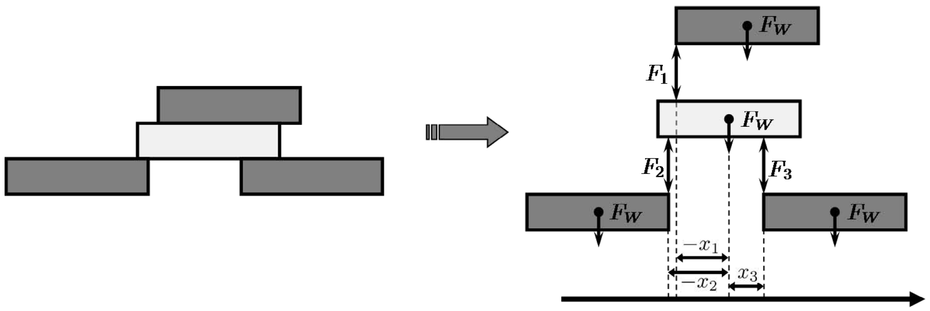


Fig. 2. To determine the forces exerted on or by a block A, a stack like the one shown on the left is decomposed into single blocks (shown on the right). Adhering gravity, each of these blocks has a downwards directed weight force F_W. In addition, there are forces exerted between the blocks, summed up to resulting forces that act on a certain point of the contact surface and for each of these forces exists an exact counterforce (F_1, F_2, F_3). Finally, momenta act on each block. They are defined by the forces acting on the block times the horizontal parts of the vectors between the forces' points of application and the block's centroid.

block this results in the following equations, Eq. 2 and Eq. 3, that need to be satisfied:

$$F_1 + F_W = F_2 + F_3 \tag{2}$$

$$x_1 F_1 + x_3 F_3 = x_2 F_2 \ , \tag{3}$$

where x_i denotes the horizontal part of the distance of the point where F_i is exerted to the centroid of the block. If and only if there exists a set of non-negative forces for which these equations for all blocks in the stack are satisfied, the stack is stable.

The model described here is the same that was already used in the studies of Hall and Paterson and Zwick, representing an abstraction of the full three dimensional block stacking problem that, i.e., can be tested by using wooden blocks like in the game *J*enga and allows for fast fitness function evaluations due to its simplicity. Additionally, it can be easily extended by, i.e., incorporating friction between blocks or using non-identical block shapes. Still, the principles underlying the optimization will stay the same, only the question if a stack is stable or not will become harder to answer.

4 An EA for the Block Stacking Problem

In the following, an EA for the block stacking problem is proposed, in particular (i) the choice of a suitable representation, (ii) evolutionary operators for generating offspring, and (iii) the considered fitness function are described. Here, choosing a suitable stack representation poses the most important but most difficult task since straight forward representations often suffer from resulting in

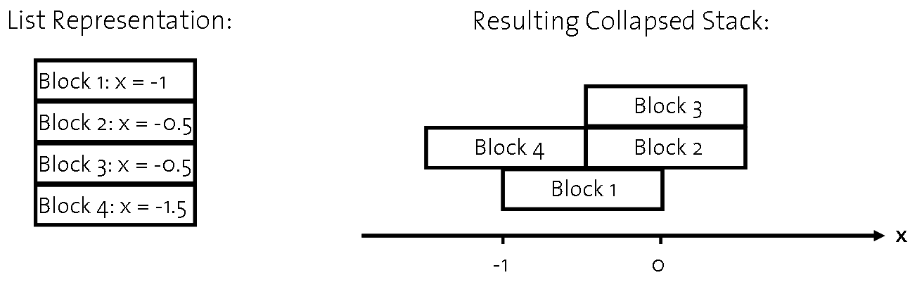


Fig. 3. Schematic overview on the transition process from representation to a stack

physically invalid stacks due to overlapping blocks and thereby make it hard to design functioning evolutionary operators.

4.1 Representation and Candidate Stack Initialization

How to represent a stack in a way that no two blocks overlap is less obvious then it appears. For example, when representing the stack by storing the x, y coordinates of a certain point of each block, without using a repair mechanism overlapping blocks can occur. To avoid this problem we have decided to use a representation which by default produces feasible stacks: A stack s is represented by an ordered list $s = [b_1, \ldots, b_n]$ of blocks where for each block b_i the x coordinate of its lower left corner is stored. To generate the stack represented by this list, the blocks are dropped from above onto the table one-by-one according to their order given in the list. Their y coordinate is then determined by the level down to which they fell before hitting either the table or another block (cf. Fig. 3). Further on, since only stacks with just a single block lying directly on the table can have an optimal overhang, stacks have to fulfill this criterion to be valid. If invalid stacks are occurring during optimization or initialization, the process leading to the invalid stack is repeated until a valid stack is generated. In addition to this validity condition, during initialization of the first individuals further criteria are posed on the stacks that need to be fulfilled. For initialization, x coordinates are randomly drawn from a normal distribution $\mathcal{N}(-1, \sigma)$[1], where σ is iteratively increased as the stack grows from bottom to top: starting from $\sigma = 0$ with each new x coordinate drawn for a stack (one block is added), σ is increased by a constant s. For a stack to become valid, it is then required that each of the newly placed blocks has a minimum contact surface to blocks on top of which they are placed. For the first block the minimal overlap in percent o is defined by an offset constant o_{off}, iteratively increased with each block so that for the last block a predefined overlap value o_{max} is reached. The iterative

[1] Since each block is represented by the x coordinate of its lower left corner and each block has a breadth of 1, the normal distribution the x coordinates are drawn from has a mean of -1: this allows for generation of stacks where the first block in expectation comes to rest with its centroid above the table.

increase for o is calculated by linearly distributing $o_{max} - o_{off}$ on the n blocks of the stack. Using this method, a hundred times the number of required stacks are generated and those stacks with the minimal stack height are chosen.

4.2 Operators

The proposed algorithm involves two selection steps, mating selection and environmental selection as well as mutation and crossover. For mating selection a tournament selection with tournament size two is used. Taking thereby selected pairs of individuals, they are first recombined and one of the resulting offspring individuals is afterwards mutated. Recombination takes place according to a predefined crossover probability p_{cross} and in case no recombination is applied, simply the first parent is handed over to mutation. After mutation, a plus-selection scheme on parent population and offsprings is used to determine the new population by taking the best individuals from this set.

Crossover and mutation are working in detail as follows: For recombination, one point crossover is used, choosing the crossover point P_{cross} uniformly distributed $P_{cross} \in \{1, 2, \ldots, n\}$, where n is the number of blocks in the stack. Afterwards the resulting offspring stack is collapsed to check if it adheres to the validity constraint (only one block lies on the table, none is falling in the void beyond the table). If the resulting stack is invalid, crossover is repeated up to 99 times and if within these 100 tries no valid stack is produced, crossover is omitted.

For mutation, one of the blocks in the stack is chosen uniformly random. This block and all the blocks on top of it in the collapsed representation are then moved according to a random movement drawn from a normal distribution $\mathcal{N}(0, \sigma)$, where σ is a predefined mutation strength. If this movement results in an invalid stack, mutation is repeated up to 99 times and if within these 100 tries no valid stack is created, the parent remains unchanged and is added to the set of offsprings. By using this mutation, stacks that are loosing height are only scarcely created although in Sec. 2 it was shown that it is necessary to have stacks with a height smaller than their number of blocks to reach optimal overhang. Therefore, in one percent of all mutation cases a different type of mutation is used: only the x coordinate of the chosen block is changed while all the others stay the same. This mutation variant is more likely to produce stacks loosing height.

4.3 Fitness Evaluation

The aim of the optimization process is to identify stable stacks with a maximal overhang, therefore to evaluate a given stack two questions need to be answered: first, is the considered stack stable and second, what is its overhang.

For a stack to be stable, in Sec. 3 it was shown that all blocks need to be in equilibrium. Therefore, for each block the corresponding equations for forces and resulting moments under the constraint that forces between blocks are non-negative and that weight forces are strictly positive, have to be set up. If and only if there exists a feasible force distribution for this usually under-determined problem, the tested stack is stable. The question if there exists a feasible solution can be formulated as a linear programming problem that in turn can be solved using standard solvers like Matlab, an operation consuming about three milliseconds for a 20 block stack on one core of a Dual Core Double CPU AMD Opteron 2.6GHz 64 Bit machine with 8GB RAM.

The overhang on the other hand can be calculated by determining the position of the rightmost block boundary or the greatest x-extension of a block—the table was located in the third quadrant and therefore the table edge is located at the origin of the Cartesian coordinate system fitted into the two dimensional space.

Since we deemed it easier for the optimization process to improve the overhang starting from a stable stack than stabilizing an instable stack with good overhang, we decided to use as a fitness $f(x)$ of stack x its overhang $over(x)$ if the stack is stable and zero otherwise which is given by the following equation:

$$f(x) = \begin{cases} over(x) & \text{if x stable} \\ 0 & \text{else.} \end{cases} \tag{4}$$

Thereby stable stacks are always preferred compared to instable ones.

5 Simulation Results

The proposed approach is tested by applying it to two instances of the block stacking problem, one using 20 block stacks and the other one stacks made up from 50 blocks. While the latter case represents a problem size that is on the verge of becoming inaccessible to exhaustive search procedures and thereby provides a glimpse on the general applicability of the proposed method, the first is used mostly for parameter optimization for the latter case. In the following first the results of the parameter optimization are presented, followed by the simulation results on the 50 block stack optimization.

5.1 Parameter Testing

During the parameter optimization process we tested mutation- and crossover rates, population- and offspring set sizes and different settings for the initialization method.

Starting with the initialization technique, offset values $o_{off} \in \{10, 20, 30, 40, 45\}$ and maximal overlaps $o_{max} \in \{50, 60, 70, 80, 90, 99\}$ were tested. Hereby, the aim was to identify a good trade-off between stack stability and stacks with small height: initial simulations showed that the optimization process becomes difficult if only few stable stacks are in the initial population (data not shown) which is often the case if only small overlaps for the blocks are required. On the other hand, to reach a good overhang it is necessary to build stacks of small height, allowing for counterweight stacks to emerge (cf. Sec. 2). Therefore we tested 10000 stacks for each combination of o_{off} and o_{max}, counting the number of stable stacks (cf. Tab. 1) with the variant $o_{off} = 45$ and $o_{max} = 99$ building the

Table 1. Trade-off between overlaps between blocks and stability measured for 10000 randomly chosen individuals, for 20 and 50 block scenarios

o_{off}	o_{max}	Fraction of stable stacks for 20 block stacks	Fraction of stable stacks for 50 block stacks
10	50	0.0003	0.0000
10	60	0.0016	0.0000
10	70	0.0052	0.0000
10	80	0.0135	0.0000
10	90	0.0275	0.0003
10	99	0.0415	0.0003
20	50	0.0017	0.0000
20	60	0.0073	0.0000
20	70	0.0192	0.0002
20	80	0.0340	0.0003
20	90	0.0618	0.0007
20	99	0.0874	0.0013
30	50	0.0070	0.0001
30	60	0.0218	0.0000
30	70	0.0434	0.0003
30	80	0.0719	0.0021
30	90	0.1071	0.0049
30	99	0.1471	0.0067
40	50	0.0205	0.0001
40	60	0.0364	0.0005
40	70	0.0746	0.0018
40	80	0.1185	0.0053
40	90	0.1793	0.0089
40	99	0.2378	0.0143
45	50	0.0251	0.0001
45	60	0.0474	0.0006
45	70	0.0912	0.0017
45	80	0.1493	0.0021
45	90	0.2285	0.0083
45	99	0.3107	0.0185

best compromise between stability and stack height. In turn, when testing the initialization method for 50 block stacks, the fraction of stable stacks dropped considerably to a value of 0.0185 still feasible for optimization but indicating that for even larger stacks the initialization technique might fail. To address this problem there are at least two possibilities, (1) couple the initialization technique with a local search technique trying to modify the stacks in such a way that they become stable or (2) initialize the stacks by using a set of stack designs that are stable by default.

Further on the influence of mutation strength and crossover rate were investigated. For mutation, different percentage levels $\sigma_{mut} \in \{0, 12.5, 25, 27.5, 50\}$ with respect to the unit block breadth have been tested while for recombination, probabilities $p_{cross} \in \{0.0, 0.25, 0.5, 0.75, 1\}$ have been tested. To measure

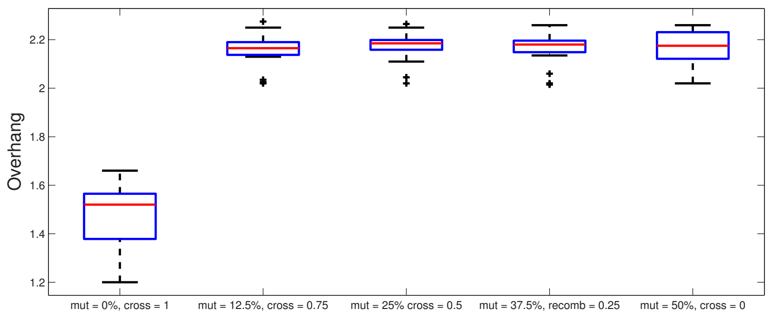


Fig. 4. Factorial design results for different mutation strengths (mut) and recombination probabilities (cross)

the overall performance, 21 optimization runs with a population size $\mu = 100$ and a offspring set size $\lambda = 200$ and 100000 objective function evaluations have been conducted. The results are shown in Fig. 4, indicating that mutation is necessary for the optimization process but as soon as mutation takes place (even if only slight movements of blocks are made) the overall performance for different mutation strengths is comparable. Only the setting where no recombination takes place shows a discernible loss of performance. For the further simulations we chose a mutation strength of $\sigma_{mut} = 25\%$ and a crossover probability of $p_{cross} = 0.25$. Using the determined parameters for initialization, mutation strength and recombination rates, in a last step we tested different population sizes μ and offspring set sizes λ. The tested settings have been $(\mu, \lambda) \in \{(10, 20), (100, 200), (200, 400)\}$. Again for each setup 21 runs, running for 100000 objective function evaluations, where conducted. The simulations showed that the mean performance for all three settings is comparable (cf. Fig. 5), whereas the variance for $(\mu, \lambda) = (10, 20)$ is much higher than for the latter two settings. In turn there is no reduction in variance when moving from $(\mu, \lambda) = (100, 200)$ to $(\mu, \lambda) = (200, 400)$ while the number of evaluations needed to reach a good overhang was smaller for $(\mu, \lambda) = (100, 200)$ than for $(\mu, \lambda) = (200, 400)$ (cf. Fig. 5.1), indicating that the population size of $(\mu, \lambda) = (200, 400)$ already could be a bit to large. In effect the best set of parameters identified during parameter optimization comprises the following settings: $o_{off} = 45$ and $o_{max} = 99$ for initialization, $\sigma = 25\%$ as mutation strength, $p_{cross} = 0.25$ as recombination probability and $(\mu, \lambda) = (100, 200)$ as population size or offspring set size respectively.

5.2 Application to a 50 Block Problem

The stacks evolved during the parameter optimization process already showed good overhangs—the best 20 block stack found had an overhang of 2.275 while for this stack size an overhang of 2.32014 is optimal and for a 19 block stack the optimal overhang is 2.27713 respectively. Both optima have been determined in [6] by exhaustive search. A simple exhaustive search results in a runtime which is

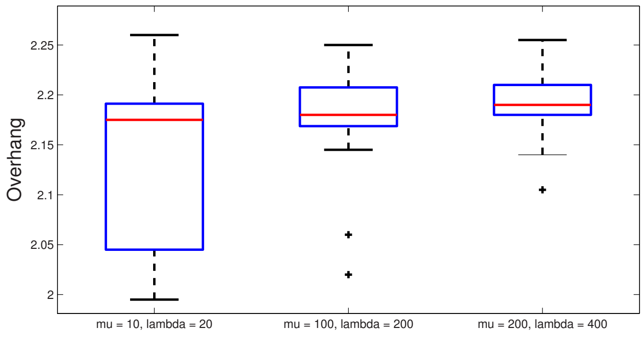


Fig. 5. Overview on the distributions of the best individuals found during 21 runs for different population sizes μ and offspring sizes λ

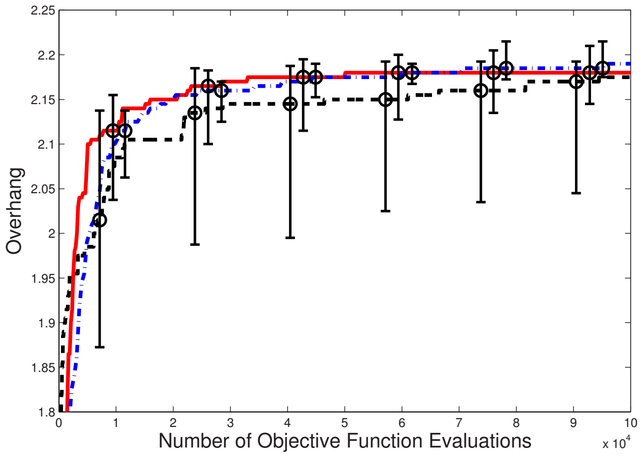


Fig. 6. Evolution of the best fitness per generation averaged over 21 runs for different population sizes. The error bars indicate the first and third quartile over the 21 runs. The dashed line represents a (μ, λ) combination of $(10, 20)$ (left error bar), the solid line of $(100, 200)$ (middle error bar) and the dash-dotted line of $(200, 400)$ (right error bar).

exponential in the number of used blocks. However, the problem complexity itself has not been determined yet. In addition to the good overhangs generated during parameter testing, the stack construction shown in Fig. 7 closely resembles the construction of the optimal ones given by Paterson and Zwick (cf. [6], Fig. 3, page 233). Taking this as a promising prospect, we decided to further test our approach on a more demanding instance of the problem: stacks containing 50 blocks. Conducting 50 runs, 100000 fitness function evaluations each, a stack with overhang 2.97 was found (cf. Fig. 7). According to Paterson and Zwick, a relatively tight putative upper bound for this problem instance is an overhang of 3.28136 and for 40 block stacks of 3.02248 [6]. Therefore the best stack found

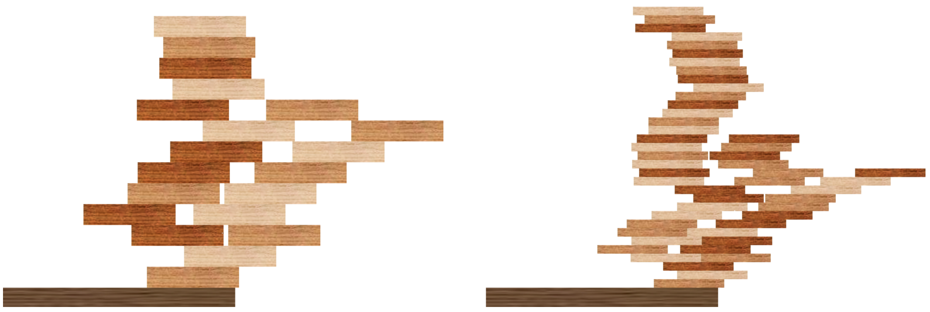


Fig. 7. Best stacks identified, for 20 blocks on the left (overhang = 2.275) and for 50 blocks on the right (overhang = 2.97)

during optimization represents a solution with some distance to the optimum, still it is a good solution serving as a proof of principle that the proposed EA is in general capable to optimize the given stack overhang problem. Nevertheless the algorithm can be improved, first by introducing a method trying to repair instable stacks by systematically introducing small changes looking for stable stacks in the neighborhood of a given solution and second by using a type of local search trying to find the stack with the best overhang in direct vicinity to the given stack.

6 Conclusions

We have presented a preliminary study, originating from a student project, concerned with the optimization of stack overhangs over the edge of a table, an example problem for static equilibria as well as an interesting test problem for combinatorial optimization methods. The proposed algorithm was applied to two instances of this problem, the optimization on 20 block and 50 block stacks.

Whilst the 20 block instance falls well within the range of exhaustively testable problems, the latter already ranges amongst those which elude themselves from exhaustive testing. Therefore, prior to tackling the 50 block problem, a parameter optimization on the 20 block instance was conducted. During the following optimization runs, for both instances good stacks where found. For the 20 block problem the found stacks were close to optimal while those for the 50 block problem have been good but only comparable to the putative tight upper bound for 40 block stacks.

The problems during optimization of the latter instance mainly stem from the fact that too many instable stacks are generated both during initialization and optimization—a problem that can be addressed in the future by coupling the proposed method with a local search procedure that is concerned with identifying stable stacks in the vicinity of a given, instable solution. Therefore, although the results not yet have been optimal, a proof of concept for the usefulness of the proposed method in tackling the given problem has been provided.

Acknowledgments

Tim Hohm has been supported by the European Commission under the Marie Curie Research Training Network SY-STEM, Project 5336.

References

1. Coffin, J.G.: Problem 3009. Amer Math Monthly 30(2), 76 (1923)
2. Dubrovsky, O., Levitin, G., Penn, M.: A Genetic Algorithm with a Compact Solution Encoding for the Container Ship Stowage Problem. Journal of Heuristics 8(6), 585–599 (2002)
3. Hall, J.F.: Fun with stacking blocks. Am J Phys 73(12), 1107–1116 (2005)
4. Housner, G.W., Hudson, D.E.: Applied Mechanics: Statics, 2nd edn., Van Nostrand, D., New York, NY, USA (1961)
5. Johnson, P.B.: Leaning Tower of Lire. Am J Phys 23, 240 (1955)
6. Paterson, M., Zwick, U.: Overhang. In: SODA 2006: Proceedings of the seventh annual ACM-SIAM symposium on Discrete Algorithms, pp. 231–240. ACM Press, New York (2006)
7. Plummer, H.C.K.: The Principles of Mechanics: An Elementary Course, Bell, G., London, UK (1929)
8. Walton, W.: A collection of Problems in Illustration of the principles of Theoretical Mechanics, 2nd edn., Deighton, Bell, Cambridge, UK (1855)

A Study of Evaluation Functions for the Graph K-Coloring Problem

Daniel Cosmin Porumbel[1], Jin-Kao Hao[1], and Pascale Kuntz[2]

[1] LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers Cedex 01, France
[2] LINA, PolytechNantes, 44306 Nantes, France

Abstract. The evaluation or fitness function is a key component of any heuristic search algorithm. This paper introduces a new evaluation function for the well-known graph K-coloring problem. This function takes into account not only the number of conflicting vertices, but also inherent information related to the structure of the graph. To assess the effectiveness of this new evaluation function, we carry out a number of experiments using a set of DIMACS benchmark graphs. Based on statistic data obtained with a parameter free steepest descent, we show an improvement of the new evaluation function over the classical one.

1 Introduction

Heuristic algorithms are known to be a very powerful tool for solving hard and large combinational optimization problems. It is now well recognized that the performance of a heuristic algorithm is strongly conditioned by the design of a number of key components. For instance, for local search algorithms, the neighborhood relation constitutes an element that must be carefully studied. Similarly, for evolutionary algorithms, it is important to seek problem specific operators such as crossover and mutation in order to obtain a better search performance. For both local and genetic search paradigms, another indispensable key component is the evaluation or fitness function. Indeed, it is this function that guides the search process to explore an arbitrarily large search space.

There are different approaches to design an informative evaluation function [14]. First, the static or dynamic penalty approach is a well established technique for constrained problems. Here relaxed constraints are integrated into the evaluation function with special penalty terms, which can be fixed statically or tuned dynamically. Second, in a hierarchical approach, the evaluation function is decomposed into several ordered components; the evaluation is realized according to that order. The third approach, less studied in the literature, consists in designing specific evaluation function specially adapted to the problem at hand. Contrary to the penalty or hierarchical approaches which are general techniques and thus applicable to different problems, the problem-specific approach requires a fine analysis of the target problem in order to identify particular properties that are useful for the design of the evaluation function. Such a problem-specific approach has demonstrated its effectiveness for several NP-hard problems [4,8,13].

In this paper, we consider the well-known graph K-coloring (K-COL) problem and try to devise a new and more informative evaluation function for a heuristic solving of this problem. Informally, for a given graph, K-COL requires to find a conflict-free vertex coloring using only K different colors such that two adjacent vertices receive two different colors. As one of the three problems chosen for the DIMACS second implementation challenge [9], K-COL is certainly one of the most studied NP-complete problems with a large number of solution algorithms.

The long-term goal of our study is the development of high performance algorithms able to produce competitive results across a large range of benchmark instances. For this purpose, we here focus our study on the discovery of a new evaluation function that will provide a better guidance than the classical penalty based evaluation function for local or genetic search algorithms. Indeed, the conventional evaluation function (called f in the paper) simply counts, for a given K-coloring, the number of conflicting vertices and consequently cannot distinguish two K-colorings of equal number of conflicts having different potential for further improvements.

2 Heuristic Search for Graph Coloring

2.1 Graph K-Coloring and Graph Coloring

Definition 1. *(K-COL) Given a graph $G = (V, E)$ (V and E are respectively the vertex and edge set) and a positive integer K such that $K < |V|$, the graph K-coloring problem is to determine whether there exists a conflict-free vertex coloring using K colors or less, i.e. a function $c : V \rightarrow \{1, 2, \cdots, K\}$ such that $\forall \{i, j\} \in E$, $c(i) \neq c(j)$. If such a coloring exists, G is said to be $K-colorable$. In the following, we denote a coloring by $C = (c(1), c(2), \cdots, c(|V|))$.*

Definition 2. *(COL) Given a graph G, the graph coloring problem is to determine the smallest K such that G is $K-colorable$. The smallest K is the chromatic number of G.*

From a theoretical viewpoint, K-COL is a very important NP-complete problem as it is one of the 21 NP-complete problems listed [10]. Coloring problems are also at the heart of numerous applications including for instance, scheduling, register allocation in compilers and frequency assignment in mobile networks.

The graph coloring problem COL is an *optimization* problem (to minimize K), while K-COL is its corresponding decision problem (to determine whether there exists a K-coloring or not). Notice that if one can solve K-COL, one can also solve COL by the following iterative approach: find a $K-$coloring of G for a fixed K, $K < |V|$ (solve K-COL), then decrease K ($K = K - 1$) until no conflict-free K-coloring can be found.

2.2 Local Search for K-Coloring

To solve K-COL by local search, we consider K-COL from an optimization point of view. For a given K-COL instance, i.e., a graph $G = (V, E)$ and an integer K, we define the following optimization problem (S, f) where:

– S is the search space composed of all the $|V|^K$ possible K-colorings, i.e. $S = \{C | C : V \rightarrow \{1, 2, \cdots, K\}\}$;
– f is an "artificial" objective counting the number of conflicting edges, i.e. $\forall C \in S, C = (c(1), c(2), \cdots, c(|V|))$,

$$f(C) = \sum_{\{i,j\} \in E} p_{ij}, \text{ where } p_{ij} = \begin{cases} 1 & \text{if } c(i) = c(j) \\ 0 & \text{if } c(i) \neq c(j) \end{cases}$$

Accordingly, any $C^* \in S$ such that $f(C^*) = 0$ corresponds to a conflict-free K-coloring and thus represent a solution to the given K-COL instance. To solve this optimization problem by local search, three main components need to be defined: an evaluation function, a neighboring relationship and an exploration strategy of the neighborhood.

– Evaluation function: One convenient *evaluation function* is the above objective function f which counts the number of conflicting edges of a given K-coloring. Indeed, this evaluation function is largely used by many well-known coloring algorithms [7,5,2,6,1]. We will show in this paper that this evaluation function is not discriminating enough and can be improved.
– Neighborhood: Given a configuration (K-coloring) $C = (c(1), c(2), \cdots, c(|V|))$, a neighboring configuration C' of C is a K-coloring $C' = (c'(1), c'(2), \cdots, c'(|V|))$ with *exactly* one conflicting color $c(i)$ being changed to $c'(i)$.
– Exploration strategy: The exploration strategy used in this paper is a steepest descent detailed in section 4.1.

3 A New Evaluation Function

For the graph K-coloring problem, many previous algorithms use f as their evaluation function, although other functions are also proposed (see for instance [12,3]). However, the function f is not sufficiently discriminating since it cannot distinguish configurations having the same number of conflicts while these colorings may have different possibilities for further improvement. To overcome this difficulty, we are trying to define another evaluation function able to identify the configurations which, despite of having the same conflict number, are more promising for further exploration.

3.1 The New Evaluation Function

Let us consider two configurations (3-colorings) C_1 (figure 1.a)) and C_2 (figure 1.b)) of the same graph for which one needs to obtain a 3-coloring without conflicts. We denote by E_{confl} the set of edges in conflict, by $\delta(i)$ the degree of vertex i and by $confl(i)$ the number of conflicts for vertex i.

The E_{confl} set has just one element for both examples: $\{i, j\}$ for C_1 and $\{i, k\}$ for C_2. Since $\delta(j) < K < \delta(k)$, we can assign to j a color not used by its $\delta(j)$ neighbors (i.e black or white) to solve the $\{i, j\}$ conflict; it requires just

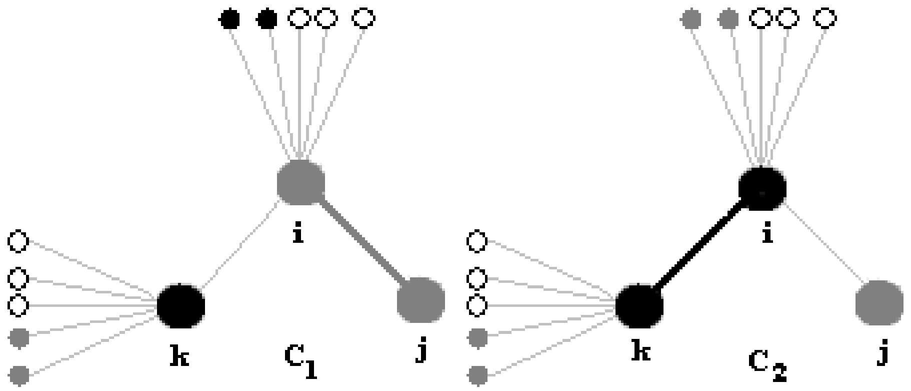


Fig. 1. Two 3-colorings with one conflict. The conflicting edge is marked in larger thickness; it is easier to solve the gray one (C_1, left) than the black one (C_2, right) even if both configurations have just a single conflict.

one more step. Since this is not necessarily the case for the conflict $\{i, k\}$ in C_2, C_1 is here preferable to C_2. Furthermore, it is natural to consider that C_1 is preferable to C_2 only because $\delta(j) < \delta(k)$ (without considering the value of K), since the more neighbors a vertex has, the more difficult it is to change its color without perturbing the rest of the configuration. In order to consider all conflicting vertices in a single formula, we propose the following evaluation function:

$$\hat{f}(C) = f(C) - \sum_{i \in V} confl(i)\frac{1}{\delta(i)}. \tag{1}$$

In all practical cases, we have $f(C) - 1 < \hat{f}(C) < f(C)$ since, for a non-trivial problem the final number of conflicts (the number of terms in the sum) is considerably smaller than the average degree of conflicting vertices (average denominator $\overline{\delta(i)}$). The second part of the function allows us to discriminate the configurations *having the same number of conflicts*; note that $\hat{f}$ preserves the f ordering: $\hat{f}(C) < \hat{f}(C')$ whenever $f(C) < f(C')$. In other words, we have two components each one with a different goal: (a) the first counts the number of conflicts (f more precisely), (b) the second is a quantity of the form $\sum_{i \in V_{conflict}} \frac{1}{\delta(i)}$ (which is less than 1) that better discriminates colorings unable to be distinguished by f. All reported values of $\hat{f}$ in this paper will be rounded to the nearest greater integer since all encountered values of $\hat{f}(C)$ satisfy $\left\lceil \hat{f}(C) \right\rceil = f(C)$.

3.2 Computational Complexity

The computational efforts required by f and $\hat{f}$ are equivalent. To see this, we re-write the formula of $\hat{f}$ in a computationally convenient way. Let us remark that:

$$\sum_{i \in V} confl(i) \frac{1}{\delta(i)} = \sum_{\{i,j\} \in E_{confl}} (\frac{1}{\delta(i)} + \frac{1}{\delta(j)}).$$

Consequently, we have:

$$\hat{f}(C) = f(C) - \sum_{\{i,j\} \in E_{confl}} (\frac{1}{\delta(i)} + \frac{1}{\delta(j)}) = \sum_{\{i,j\} \in E_{confl}} (1 - \frac{1}{\delta(i)} - \frac{1}{\delta(j)})$$

In our implementation, both functions are constructed by adding constant coefficients ($E[i,j] = 1 - \frac{1}{\delta(i)} - \frac{1}{\delta(j)}$) for each conflicting edge $\{i,j\}$. All values from E are computed before starting the main algorithm and they have a time complexity of ($O(|V|^2)$). In our numerical experiments, the computing time for E is always less than 1 second on a Pentium 4 with a CPU at 2.8GHz. The only non-negligible difference is the data types we are manipulating: instead of integer values we use double values to store the table E and to perform all operations.

4 Experimental Comparisons of the Two Evaluation Functions

In this section, we perform an extensive experimental analysis of the effect of $\hat{f}$ on the steepest descent (SD) algorithm. We start by detailing the algorithm, then we present the test instances, the comparison criteria and we analyze the results.

4.1 Steepest Descent

The steepest descent (SD) algorithm starts from a random initial configuration and iteratively chooses, from the whole neighborhood, the *best* neighbor according to the evaluation function. When there exists several equally best neighbors, one of these neighbors is chosen at random. The algorithm stops when there exists no improving neighbor - i.e. when the current coloring is a local optimum with respect to the given neighborhood relation and evaluation function. For all the experimental results reported in this paper, this simple SD algorithm is considered with the two evaluation functions f and $\hat{f}$.

The choice of the SD algorithm for our experimentations is here justified by the fact that it is one of the few algorithms to ensure a complete neutrality in the final results. In any algorithm which closely depends on a given parameter (e.g. temperature in simulated annealing, tabu list length in Tabu Search, etc), the tuning of this parameter might significantly skew the results favoring one method or another.

We also started performing experiments with other more advanced algorithms (especially Tabu Search), but however, the aim of this paper is not to compete with the best graph coloring algorithms. The goal of our experiments is to study the influence of evaluation functions for the graph coloring algorithms from a completely neutral point of view.

4.2 The Experimental Conditions

Instances. The following graphs from the well-known second DIMACS challenge benchmarks are used:

– Five uniform random graphs generated by Johnson *et. al* in their state-of-the-art papers about simulated annealing [8] and used extensively afterwords in testing graph coloring algorithms: $dsjc250.5$, $dsjc500.5$, $dscj1000.1$, $dsjc1000.5$ and $dscj1000.9$. They have 250, 500 and 1000 vertices respectively and the density p is denoted by the last digit (i.e. .5 for the first graph).
– Three Leighton graphs: $le450.15a$, $le450.25a$ and $le450.25c$, these graphs have each 450 vertices and a known chromatic number (15 for the first, 25 for the others) [11]. The last two graphs are generated in the same manner, but with different random seeds.

Comparison Criteria. The main indicator of *solution quality* is the number of conflicts of the configuration obtained at the end of the search (Other criteria are explained in Section 5. For each graph, we set K to be the smallest number of colors for which a coloring has ever been found or the chromatic number when it is known (i.e. for the Leighton graphs). Consequently, all these instances are difficult to solve. For the first five graphs (dsjc*.*) we use the least K found either by a hybrid algorithm [6] or by a population based local search algorithm combining two specific neighborhoods and using the strategy of successive building of color classes [12].

Experimental Protocol. The experimental evaluation was carried out by considering 1000 independent runs, with different random seeds, for both functions and by statistically analyzing the solutions obtained. We examine the extremal conflict numbers found with the two evaluation functions and precisely analyze the distributions of the values obtained on the run set.

4.3 Results

For each instance, we show in table 1 the minimum, the mean and the maximum solution quality (columns 2,3,4 and 6,7,8 respectively) computed separately for both functions. We also compute the standard deviation (columns 5 and 9 respectively), as it is an indicator of the algorithm's precision and robustness.

The first observation is that the distributions of f and $\hat{f}$ are disjoint in 75% of cases: the quality of the solutions obtained with $\hat{f}$ is always better (with smaller numbers of conflicts) for absolutely all runs. And moreover, even for the rest of 25% cases, the general tendency remains the same.

More surprisingly, let us remark that $\hat{f}$ leads one time to a proper coloring (no conflicts) for the $le450.25a$ graph with K=25 (its chromatic number). In fact, even some state-of-the-art algorithms like HGA ([6]) fails to find a conflict free coloring with 25 colors for graphs in the $le450.25a$ family.

Furthermore, we collected for each graph the final values obtained by our SD algorithm with the two evaluation functions f and $\hat{f}$ into a single sample

Table 1. The results of 1000 runs of the SD algorithm on all the tested graphs. $\hat{f}$ allows SD to obtain a better local optimum with a smaller number of conflicts for each graph and even to find an optimal coloring for $le450.25a$. The numbers between the parenthesis in Column 1 correspond to the smallest K reported in the literature.

	Classic Function(f))				New Function($\hat{f}$)			
Graph (colors)	Min	Max	Mean	Std. Dev.	Min	Max	Mean	Std. Dev.
$dsjc250.5(28)$	60	106	83.0	7.4	36	71	54.1	5.6
$dsjc500.5(49)$	140	209	173.1	10.7	89	136	112.2	8.2
$dsjc1000.1(20)$	260	355	307.2	15.2	152	231	191.8	11.7
$dsjc1000.5(83)$	364	478	424.9	16.6	249	333	290.0	13.3
$dsjc1000.9(224)$	301	402	347.5	14.4	183	253	218.9	9.5
$le450.15c(15)$	270	345	310.3	10.6	216	284	250.3	9.9
$le450.25a(25)$	11	28	18.2	2.8	0	10	4.6	1.6
$le450.25c(25)$	87	128	107.7	6.1	51	78	64.1	4.8

and depicted in Figure 2 the distribution of the results according to two axes: quality (number of conflicts) and the frequency (number of colorings having a given quality). The distribution confirms once again the superiority of $\hat{f}$ in the search process. Indeed, the distribution with $\hat{f}$ is more on the left than the distribution with f, meaning that the solutions with $\hat{f}$ have a smaller number of conflicts.

Additionally, it is important to remark that all 1000 configurations found for each method and each graph are pairwise different, i.e. the algorithm never comes to two identical solutions. We consider two configurations $(c(1), c(2), \cdots c(|V|))$ and $(c'(1), c'(2), \cdots, c'(|V|))$ to be identical if and only if there exists a permutation σ of the set $\{1, 2, 3, \cdots, K\}$ such that the first configuration is mapped into the second by σ (i.e $\sigma(c(i)) = c'(i), \forall i \in \{1..|V|\}$).

5 Why the New Evaluation Function Works?

In this section, we try to understand why the new evaluation function $\hat{f}$ works better than the classical one f. For this purpose, we analyze the dynamics of the steepest descent with f and $\hat{f}$ and consider three indicators: a) the convergence of the SD algorithm with f and $\hat{f}$, b) the number of quality-improving neighbors induced by each evaluation function and c) the cardinality of equivalence classes of configurations.

5.1 Convergence

Table 2 indicates the total number of iterations performed by typical search processes using both functions. Figure 3 depicts the evolution of solution quality with both functions along the same scale. These results show that the SD with

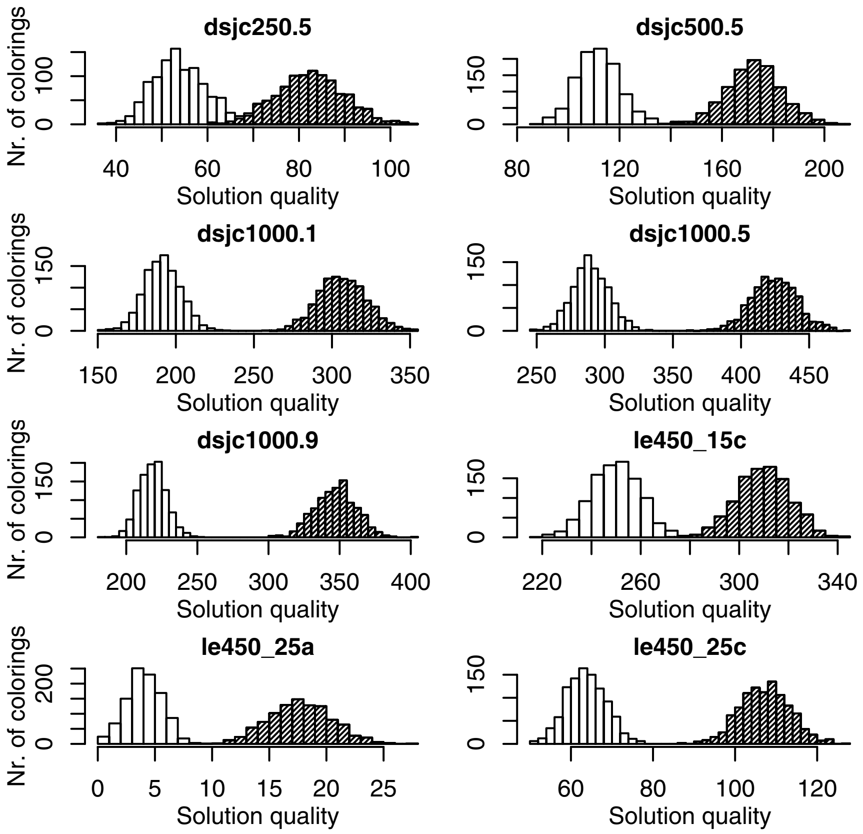


Fig. 2. The solution quality (evaluation function value) distribution for all graphs considering 1000 random steepest descents with the new function $\hat{f}$ (denoted by simple bars) and the classic one f (denoted in shading lines)

Table 2. The number of iterations performed by the steepest descent using f and $\hat{f}$. The descent process with $\hat{f}$ lasts always longer than the descent with f.

Graph (colors)	New Function($\hat{f}$)	Classic Function(f)
$dsjc250.5(28)$	245	158
$dsjc500.5(49)$	465	353
$dsjc1000.1(20)$	941	613
$dsjc1000.5(83)$	1003	703
$dsjc1000.9(224)$	921	635
$le450.25a(25)$	191	154

f is trapped earlier in a local optimum than with $\hat{f}$. For instance, for the graph ($dsjc250.5$), while the descent with f stops at 159^{th} iteration, the search continues with $\hat{f}$. This is possible because $\hat{f}$ offers improving neighbors for a longer time.

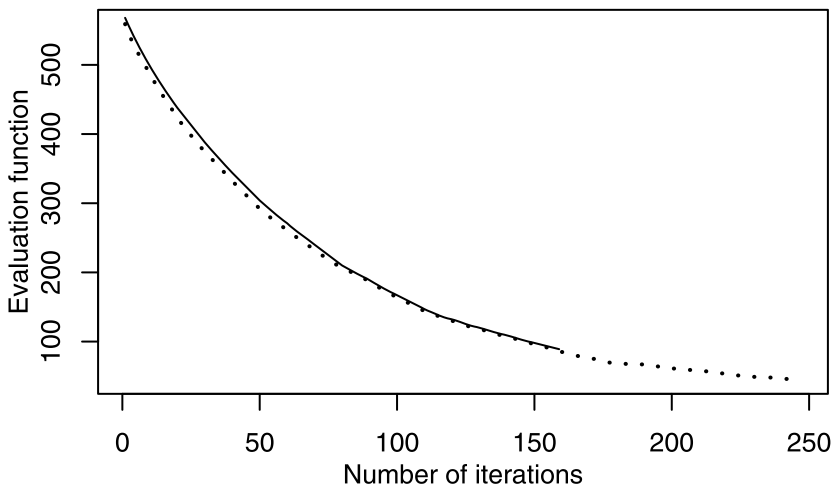


Fig. 3. The solution quality (number of conflicts) evolution for a classic steepest descent run ($G = dsjc250.5, K = 28$); $\hat{f}$ is depicted in dotted line, f in continuous line. The descent process with f stops at 159^{th} iteration while the search continues with $\hat{f}$.

5.2 Neighborhood Analysis

An important indicator about the dynamics of a search process is the evolution of the number of improving neighbors during the search. Intuitively, if this number decreases rapidly, the search process might easily get blocked in a local optima. This is particularly true for a descent algorithm. Indeed, in a SD algorithm, the number of improving neighbors tends to monotonically decrease until this number drops to zero thus triggering the stop condition.

Let Δ denote the improvement added by a neighbor vertex V_{next} to the current vertex $V_{current}$ according to f or $\hat{f}$: $\Delta f = |f(V_{next}) - f(V_{current})|$ or $\Delta \hat{f} = |\hat{f}(V_{next}) - \hat{f}(V_{current})|$. For each coloring, we are interested to study: 1) how many improving neighbors are there at each step and 2) what is the actual improvement these neighbors can add (what values Δ can take).

Figure 4 depicts the first indicator on a typical run by drawing the curves for the number of neighbors satisfying $\Delta > 0$ (thin lines) and $\Delta = 0$ (thick lines). These curves confirm that $\hat{f}$ is more discriminating than f: the thick curves present significantly lower values for $\hat{f}$ than for f and, at each step, there are numerous equivalent neighbors for f especially.

Figure 5 depicts the distribution of Δ values, showing the degree of the improvements according to the two evaluation functions. An intriguing observation is that the improvement at each iteration is rather small; the algorithm performs many small steps rather than few large steps. In figure 5, one can see that the number of neighbors improving the current solution by more than 3 is usually very low. In most cases, the actual possible improvement of both f and $\hat{f}$ is 1 or 2. Furthermore, note that at the end of the search using $\hat{f}$ there are some improvements of less than 1 (in fact of almost 0). The existence of these

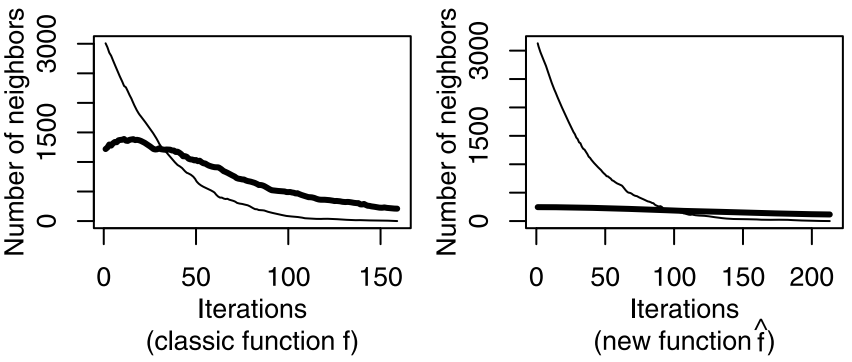


Fig. 4. A typical evolution of the number of quality-improving neighbors ($\Delta > 0$, in thin line) and of quality-stagnation neighbors ($\Delta = 0$, in thick line) for f (left) and $\hat{f}$(right)

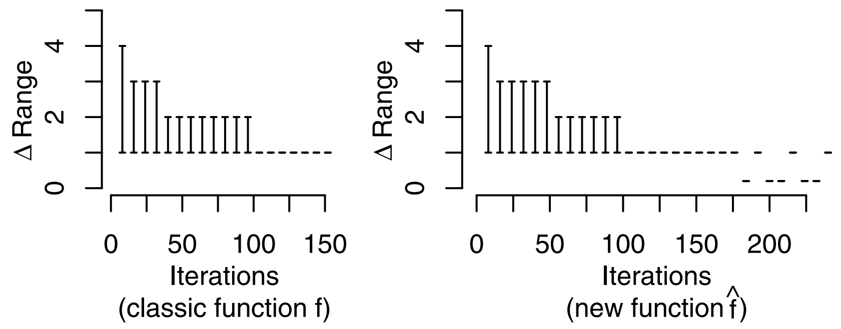


Fig. 5. A typical evolution of the possible improvement (values of Δ) in the neighborhood according to f (left) and $\hat{f}$ (right)

improvements is justified only by the second part of the function $\hat{f}$ and not by the conflict number; and this explains why $\hat{f}$ can well distinguish between these colorings having the same number of conflicts.

5.3 Classes of Configurations in the Landscape

An analysis of equivalence classes of configurations may be also useful for a better understanding of the dynamics of the search process. In our case, an equivalence class is a set of configurations that are evaluated at the same value by f or $\hat{f}$. Thus, two colorings are in the same f-class if they have the same conflict number and in the same $\hat{f}$-class if they have, in addition, the same degree distribution of their conflicting vertices. Consequently, the cardinal of a $\hat{f}$-class is considerably smaller and this is another indicator why the $\hat{f}-$search is more discriminating.

The cardinal of an $\hat{f}$-class is strongly influenced by the degree distribution of the considered graph. In regular graphs (all vertices have the same degree), any f-class is also an $\hat{f}$-class (so there is no practical difference between the two functions) since all edges $\{i,j\}$ generate the same values $E[i,j] = 1 - \frac{1}{\delta(i)} - \frac{1}{\delta(j)}$. At the opposite, in a heterogeneous graph with few vertices having the same degree, there are statistically very few edges generating equal values in the table E.

6 Conclusions and Further Work

In this paper we have introduced a new evaluation function $\hat{f}$ for the graph K-coloring problem. This evaluation function is based not only on the conflicts induced by a K-coloring, but also on information related to the structure of a graph. The experimental results with a steepest descent algorithm show that this function outperforms the classic evaluation function which is based only on the conflict number. To explain the good performance of the new evaluation function, some empirical justifications were proposed, based on the distribution of improving neighbors induced by each evaluation function and an analysis of the equivalent classes of configurations.

To further assess the practical usefulness of the proposed evaluation function, we are experimenting this function within a Tabu Search (TS) algorithm. The preliminary results show that the new evaluation function boosts the Tabu Search algorithm. Indeed, for a large number of the DIMACS graphs, the TS algorithm using $\hat{f}$ (as well as some other simple improvements) finds the best known colorings. Moreover, it is even able to improve on half of the results obtained by previous Tabu Search algorithms.

More generally, we believe that the evaluation function introduced in this paper may be useful for other heuristic coloring algorithm and shed light on the design of other informative evaluation functions for the graph coloring problem.

Acknowledgments. This work is partially supported by the CPER project "Pôle Informatique Régional" (2000-2006) and the Régional Project MILES (2007-2009). We would like to thanks our referees for their useful comments.

References

1. Avanthay, C., Hertz, A., Zufferey, N.: A variable neighborhood search for graph coloring. European Journal of Operational Research 151(2), 379–388 (2003)
2. Dorne, R., Hao, J.K.: Tabu search for graph coloring, T-colorings and set T-colorings. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, 77–92 (1998)
3. Eiben, A.E., van der Hauw, J.K.: Adaptive penalties for evolutionary graph coloring. In: Rao, A., Singh, M.P., Wooldridge, M.J. (eds.) ATAL 1997. LNCS, vol. 1365, pp. 95–106. Springer, Heidelberg (1998)
4. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics 2, 5–30 (1996)
5. Fleurent, C., Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63, 437–461 (1996)
6. Galinier, P., Hao, J.K.: Hybrid Evolutionary Algorithms for Graph Coloring. Journal of Combinatorial Optimization 3(4), 379–397 (1999)

7. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
8. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. Operations Research 39(3), 378–406 (1991)
9. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, AMS, USA (1996)
10. Karp, R.M.: Reducibility among combinatorial problems. Complexity of Computer Computations 43, 85–103 (1972)
11. Leighton, F.T.: A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 84(6), 489–503 (1979)
12. Morgenstern, C.: Distributed Coloration Neighborhood Search. In: [9], pp. 335–357 (1996)
13. Rodriguez-Tello, E., Hao, J.K., Torres-Jimenez, J.: An improved evaluation function for the bandwidth minimization problem. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 650–659. Springer, Heidelberg (2004)
14. Rodriguez-Tello, E., Hao, J.K.: On the role of evaluation functions for heuristic search (working paper, 2007)

Genetic Branch-and-Bound or Exact Genetic Algorithm?

C. Pessan[1,2], J.-L. Bouquard[1], and E. Néron[1]

[1] LI, Université François Rabelais Tours, 64 av. Jean Portalis, 37200 Tours, France
`cedric.pessan@univ-tours.fr`
[2] SKF France SA, Industrial division, MDGBB Factory, 204, boulevard Charles de Gaulle 37542 Saint-Cyr-sur-Loire CEDEX, France

Abstract. Production resettings is a vital element of production flexibility and optimizing the setup tasks scheduling within a production channel is required to improve production rate. This paper deals with a NP-Hard production resetting optimization problem based on an industrial case. In this paper we present how to hybrid a Branch-and-Bound method for this problem with a genetic algorithm. The idea is to use the genetic algorithm to improve the upper bound and thus speeding up the Branch-and-Bound while the genetic algorithm uses the content of the Branch-and-Bound stack to reduce its search space. Both methods are running in parallel and are therefore collaborating together.

1 Introduction

Improving production flexibility is one of the main problems encountered in the industry as it is closely linked with customer service improvement and the length of delays between customers orders and delivery of products. It is thus important to reduce resetting times between batches. A production resetting consists in operations made on each machine of a production channel made by operators. These operations are required to setup the machines for the new batch. One way to improve resetting times is to work on the global organization of the different setup tasks according to industrial constraints, e.g., the skills of the operators, their availability periods. The study is based on a real industrial case found in SKF MDGBB (Medium Deep Groove Ball Bearings) factories.

This problem can be identified as an unrelated parallel machine scheduling problem, for which we have developed a Branch-and-Bound method in Pessan et. al. [2006b]. The main drawback of this method is the upper bound that is so far away of the optimal solution in our experiments that the method can not prune any node at the beginning of the resolution. On the other hand, we have also developed in Pessan et. al. [2006a] a heuristic on a more general problem (serial-parallel production channel) based on a genetic algorithm hybridized with a local search that proved to work well for this problem. The idea of this paper is to hybrid the Branch-and-Bound with a genetic algorithm in order to get the best of both methods: fast convergence to good solutions and exact resolution. The genetic algorithm is not only run at the root node but in parallel with the Branch-and-Bound. Moreover, the encoding method is based on the content of the Branch-and-Bound stack: it means that while the Branch-and-Bound progresses, it reduces the search space of the genetic algorithm, and when the genetic algorithm improves the best known solution, it helps pruning more nodes. So, both methods are really collaborating together during the whole execution.

It is natural to use genetic algorithms to find a good upper bound of the optimal solution quickly either on the root node or regularly during the execution of the Branch-and-Bound. Such attempts have been made in several papers. Portman et. al. ([1998]) use a genetic algorithm on the root node of a Branch-and-Bound in order to provide a good initial upper bound to the Branch-and-Bound. Jouglet et. al. ([2005]) propose a similar approach but they use the genetic algorithm to provide an initial upper bound to a constraint programming method. In Basseur et. al. ([2005]) a biobjective unrelated parallel machine problem is tackled with a genetic algorithm that provides an initial pareto front to a 2 phases Branch-and-Bound. Branch-and-Bound are also commonly hybridized with other meta-heuristics like in Rocha et. al. ([2004]): the GRASP meta-heuristic is used to provide an upper bound to a Branch-and-Bound method. Cotta et. al. ([1995]) show some preliminary results on various combination of Branch-and-Bound and genetic algorithms: they have tried using Branch-and-Bound like methods as local search operator of a genetic algorithm leading to a heuristic hybrid method. On the other hand, they propose an hybrid method that run in parallel a Branch-and-Bound and a genetic algorithm but they mention some difficulties in handling diversification of the genetic algorithm population and the slow convergence of genetic algorithms for the traveling salesman problem they are working on. French et. al. ([2001]) present also such a hybrid algorithm, they use the Branch-and-Bound to find promising nodes and thus generate the initial genetic algorithm population and then use the genetic algorithm results to give hints to the Branch-and-Bound on where there can potentially be interesting solutions. They switch back and forth between the two methods. Their results seem promising. Puchinger ([2005]) in its survey on these hybrid methods distinguishes between integrative combinations that tend to use one method as an operator of the other and collaborative methods. This second category is also categorized between sequential execution and parallel execution. It is also mentioned that parallel execution algorithms have not been extensively tried but with the emergence of mainstream multi core processors that can execute in parallel several algorithms with fast access to a common memory area that ease data sharing between the algorithms, it may be time to give importance to such algorithms.

In this paper we present in section 2 the model of the problem we are studying and the existing Branch-and-Bound method. Then, in the section 3 we describe the hybrid method. Finally, experimental results are presented in section 4.

2 Existing Methods

2.1 Problem Description

Let n be the number of machines of a production channel, it is also the number of tasks to schedule. For each machine (or task) $M_i, i \in \{1, \dots, n\}$, we know its release date r_i. It is the minimum duration needed for the last ball bearing of the previous batch to go from M_1 to M_i. We also know its tail q_i, the minimum duration needed for the first ball bearing of the next batch to go from M_i to M_n. When a machine M_i is restarted, it can not have any effect on production rate that is measured at the end of the production line before q_i time unit. t_i and C_i denote respectively the beginning and the completion time of setup task on machine M_i.

In the production unit, there are λ operators. Each operator $O_h, h \in \{1, \dots, \lambda\}$, depending on his own experience, needs a different time to set up a machine: this time is denoted $p_{i,h}$. If an operator O_h does not have the skill for a machine M_i, we set, without loss of generality, $p_{i,h} = +\infty$. Moreover each operator is only available during a time interval $[R_h, D_h]$.

In this paper, we consider serial channels: it means that the production can only restart when all machines have been setup. Therefore, we have to optimize the maximum completion time of the setup tasks, also known as $C_{max} = max_{i=1,\dots,n} C_i + q_i$ in standard scheduling notations.

According to classical scheduling problem classification, this problem can be identified as an unrelated multipurpose parallel machines problem with release dates and tails. In our problem, resources are the operators, operations are the setup tasks of each machine. This problem is denoted $R, MPM|r_i, q_i|C_{max}$. Figure 1 presents an instance made up of 4 machines and 2 operators.

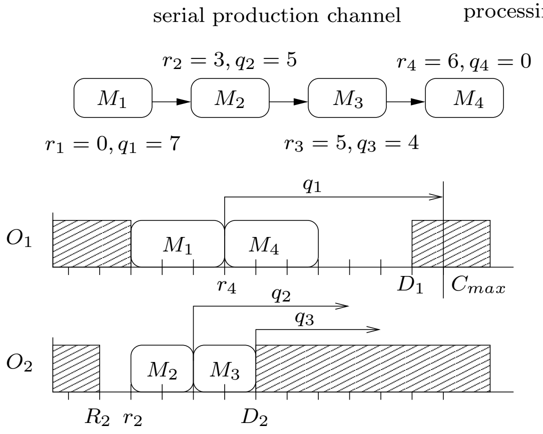
Moreover as explained previously, r_i and q_i can be seen as distances in time from machine M_i to respectively the beginning and the end of the production channel. It means that for a machine M_i, the farther it is from the beginning of the channel, the closer it is to the end of the channel. Then the non decreasing r_i order is the same as the non increasing q_i order. This proposition (prop. 1) is illustrated on the figure 2.

Proposition 1. *In a serial channel, r_i and q_i are such that:* $\forall i \in \{1, \dots, n\}, r_i \leq r_{i+1}$ *and* $q_i \geq q_{i+1}$.

Corollary 1. *On each operator, scheduling tasks in non decreasing release date order is optimal*

Using proposition 1, it is easy to deduce the corollary 1 as shown in Pessan et. al. ([2006b]). Therefore, the problem can be seen as an assignment problem: once tasks are assigned to operators, their order and then their starting time are deduced using corollary 1.

The $R, MPM|r_i, q_i|C_{max}$ problem is $\mathcal{NP}-Hard$ even in our case of serial channel since the particular case $P, MPM||C_{max}$ is known to be $\mathcal{NP}-Hard$ as shown by Garey and Johnson ([1978]).



processing times related to skills of operators

$p_{i,h}$	O_1	O_2
M_1	3	2
M_2	2	2
M_3	∞	2
M_4	3	∞

	O_1	O_2
R_h	3	7
D_h	2	12

Fig. 1. A 2 operators 4 machines instance

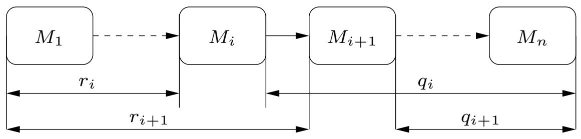


Fig. 2. Property on r_i and q_i

Unlike the $P|r_i, q_i|C_{max}$ problem studied in Carlier ([1987]) and Gharbi and Haouari ([2002]) and unlike the multipurpose parallel machines problem studied in Jurisch ([**?**]), the $R, MPM|r_i, q_i|C_{max}$ problem has not been extensively studied in the literature. We can mention for instance Gharbi and Haouari ([2005]) but the corollary 1 on our specific problem allows us to implement more efficient algorithms (cf. section 2.2).

2.2 Branch-and-Bound Method

In this section we describe briefly the Branch-and-Bound method presented in Pessan et al. ([2006b]). This Branch-and-Bound is used in the hybrid method.

Generalities: A Branch-and-Bound method is a classical way of implicitly enumerating all solutions of a search space to find the optimal one. In a Branch-and-Bound method the search space is assimilated with a tree stored using a data structure, e.g. stack, containing not yet explored nodes. Each node represents a partial solution and a sub-domain of the search space. Moreover, going from one level to the next one means making a decision and reducing the domain of at least one variable. On a leaf node, all variables are fixed. The figure 3 shows the relationships between the search space representation, the tree and the stack. The tree is a representation of the search space and the stack contains the frontier of the unexplored parts of the search space.

At each iteration, a node (N) is extracted from the stack and a lower bound $lb(N)$ of all the solutions in its corresponding sub-space is computed. Then, this

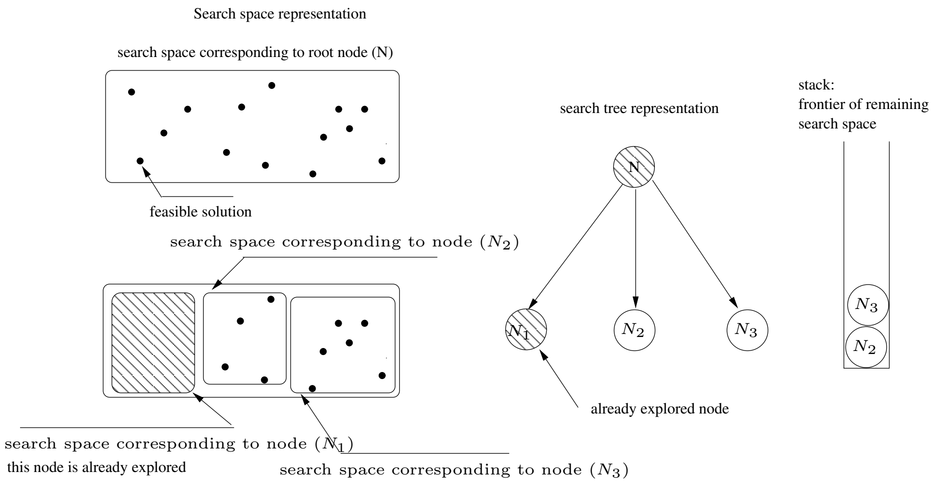

Search space representation
search space corresponding to root node (N)
feasible solution
search space corresponding to node (N_2)
search space corresponding to node (N_1)
this node is already explored
search space corresponding to node (N_3)
search tree representation
N
N_1
N_2
N_3
already explored node
stack:
frontier of remaining
search space
N_3
N_2

Fig. 3. Relationship between search space, stack and tree

lower bound is compared to the best known solution ub also known as the upper bound of the optimal solution. If (N) is a leaf node and if its criterion is better than the upper bound, the upper bound is updated. Otherwise, if the lower bound is greater that the upper bound, i.e. $lb(N) > ub$, the node is discarded (we also say that the node is pruned) and if it is lower, child nodes are generated and pushed on the stack. A cutting rule can be seen as an extension of the use of a lower bound to prune the search tree. A cutting rule is a procedure that takes one parameter D and returns a boolean. Answer 'no', corresponds to the fact that no promising solution, i.e., solution having a makespan lower than or equal to D, could be found in the subspace corresponding to the current node. If the cutting condition answers 'no' with D= $ub - 1$ then the node can be pruned.

Notice that generally lower bounds are computed once node is extracted from the stack, but in the case that the exploration strategy is based on the lower bounds, e.g, best-first strategy (see section 3.2), the lower bound is computed before new nodes are pushed on the stack.

Branching Scheme: According to corollary 1 the only decision we have to make is the assignment of the tasks to operators. Moreover without loss of generality, we assume that the tasks are already sorted so that $\forall i \in \{1, \ldots, n-1\}, r_i \leq r_{i+1}$. At each level (i) of the search tree, the branching scheme tries to assign task M_i to each operator that is able to perform the task. It means that there are n levels in the tree and a maximum of λ branches per node depending on the number of operators who master the skill corresponding to the task.

Upper Bound: The upper bound is computed using a simple greedy algorithm that uses the Earliest Completion Time (ECT) priority rule. It is assigning tasks to the operators who can complete it first. In our experimental results (Pessan et. al. [2006a]), this upper bound was the main problem of the method as it was too far away from the optimal to prune nodes. Moreover, solutions built using ECT may not be feasible, regarding to $[R_h, D_h]$.

Cutting Rule: The cutting rule is based on relaxation of non-preemption constraint : it means a task can be interrupted and restarted later either on the same operator or on another one. We define deadlines for each task as $\tilde{d}_i = ub - 1 - q_i$. The lower bound checks if it is possible to achieve all the tasks within the allowed time-windows (within $[r_i, \tilde{d}_i]$). If it is not possible, then the node can be pruned. Checking this can be done polynomially using a linear program as shown by Lawler and Labetoulle ([1978]).

3 Hybrid Method

3.1 Generalities

We have seen that a method such as Branch-and-Bound is enumerating implicitly all feasible solutions. The search space can be reduced whenever a node (N) can be pruned, that is when $lb(N) \geq ub$ or when the cutting condition return 'no', thus, the upper bound is an important part: the upper bound is improved as the method is discovering better solutions but as long as the upper bound is not close enough of the optimal, it is usually hard to prune nodes as the condition $lb(N) \geq ub$ is rarely satisfied. So it is important to find a good upper bound as fast as possible. Moreover, if the search is stopped before reaching the optimal solution, it is interesting to have a good solution that can be given to the decision maker.

Here we describe how a Branch-and-Bound can be improved using an efficient method to compute an upper bound, namely, genetic algorithm. The genetic algorithm can be used at the root node or it can be used at any time to search a better upper bound. The idea, is that the genetic algorithm process should focus on improving the best known solution by searching within the remaining search space, i.e, whose frontier is still in the stack of the Branch-and-Bound, while the Branch-and-Bound should eliminate as large parts of the search space as possible. Basically there are two ways to modify the search space. First, each time child nodes are created in the Branch-and-Bound, the search space of the parent node is subdivided into disjoint sub-spaces of the newly created nodes. Second, when a node is pruned in the Branch-and-Bound, a part of the search space is eliminated. So the genetic algorithm should be implemented in a way such that it only searches within the unexplored areas and a convenient way to do this is to use the nodes contained in the stack.

3.2 Hybrid Exact Genetic Algorithm

Genetic algorithm is a meta-heuristic inspired by biological evolution that has been introduced by Holland ([1975]). The method uses a population of solutions that are encoded using a carefully chosen encoding function: these encoded solutions are called individuals or chromosoms. During each iteration, the algorithm follows these steps:

- **selection:** pairs of individuals are selected
- **crossover:** a crossover operator is applied to the selected individuals pairs.

- **mutation:** each generated individuals may be mutated by slightly changing them. The probability of mutating an individual should be low
- **replacement:** individuals that will survive through next generation should be selected in order to keep a constant population size

The idea is that good individuals should propagate their characteristics. This kind of algorithms has proved to work well on a more general case of the problem (serial-parallel production channel, see Pessan et. al. ([2006a])), that is why we have tried to use a genetic algorithm to improve the Branch-and-Bound.

Encoding Function: As mentioned above, the corollary 1 means that our problem can be reduced to the assignment problem of tasks to operators. So, the encoding function of the genetic algorithm should only contain assignments of operators to the tasks. The decoding function of the genetic algorithm only has to order tasks in non decreasing r_i order, and then computing starting time of tasks.

As shown on figure 4, the chosen encoding function generates individuals compound of three parts: a node, the node level and an assignment array. A node structure contains a partial solution and its level in the search tree: a node at level k containt k assignments. So, the node of the individual defines the first assignments. The remaining assignments should be encoded in the third part of the individual.

node	node level	assignments of remaining tasks
⑦	2	1/3/4/4

Fig. 4. An example of an encoded solution for a problem with $n = 6$ and $m = 4$

The individual of the example presented in figure 4 is using the node 7 that is at level 2 of the search tree: the 2 first assignments are taken from the node. The node contains only a partial solution with 2 assignments, so the rest of the individual requires $n - node\_level = n - 2 = 4$ additional assignments needed to build a solution.

The advantage of this function is that the hybrid method is using the stack to generate individuals and keeping the node in the encoded individual eases the synchronization of the population with the stack content as explained below.

Crossover Operator: The crossover operator is a classical one point crossover that generates two individuals from two parents. It selects randomly a number p between 0 and n. To generate the first child, the operator copies, p assignments from the first parent and $n - p$ assignments from the second parent. The node of the child is the node of first parent. The second child is created by exchanging the roles of parents. An advantage of this operator is that child always contains existing nodes of the stack and valid assignments and thus belong to the remaining search space whose frontier are the nodes of the Branch-and-Bound stack.

On the example of the figure 5, p is set to 3. So, 3 assignments are copied from the first individual to the first child: the 2 assignments of the node and

individual 1:	⑦	2	1/3/4/4
individual 2:	㉑	4	2/1
cross point : 3			
child 1:	⑦	2	1/X/2/1
	X should be extracted from partial solution of node ㉑		
child 2:	㉑	4	4/4

Fig. 5. Crossover example

one additional assignment. Then, other assignments are taken from the second individual: one that comes from the node 21 and the rest from the third field of the individual.

Mutation Operator: There are two mutation operators in this method. The first one is randomly changing a gene within the third field of an individual, that is, within the assignments that complete the partial solution of the node.

The second mutation operator is randomly switching to a new node present in the stack. If the new node is at a lower, it extracts missing assignements from the old node. On the example of figure 6, node 7 that is at level 2 is replaced by node 21 that is at level 4: the 4 first assignements of the mutated individual are extracted from node 21 and the two remaining assignments come from the original individual.

before mutation:	⑦	2	1/3/4/4
after mutation:	㉑	4	4/4

Fig. 6. Second mutation operator example

The probability $pm1$ of mutating an individual should be set to a high value mainly because of the chosen encoding function and crossover operator: as long as there are no mutations, no new assignments are introduced in the population. But it is also required because of the need to explore quickly a subspace in this hybrid method. If there is a mutation, the probability $pm2$ of using the second operator should be low because this operator change many assignments in an individual and using it too much would lead to too much randomness.

Synchronization Operator: it is an operator that is specific to our hybrid method. It is there to check that the genetic algorithm is only searching within the unexplored area: the part of the search space whose frontier is in the stack. This operator requires non negligible amount of time to be executed and should not be executed at each iteration. Moreover if it is called too many times, it may be hard for the genetic algorithm to evolve correctly as it would eventually invalidate solutions as soon as they are created. So we have chosen to call it every $maxIt$ ($maxIt = 2000$) iterations of the genetic algorithm.

The operator is simply checking that all individuals of the genetic algorithm has a node that is still in the stack. If a node that is not anymore in the stack is discovered the second mutation operator (that changes the node of an individual) is called.

Moreover, if there is no improvement of the best solution using the genetic algorithm for quite a while and if we are running the method on a mono-processor computer, the genetic algorithm can be paused a few seconds in order to give the Branch-and-Bound more cpu time: this is done because in many cases it can takes time to prove that we have found the optimal solution and then trying to improve the best known solution is not relevant.

Finally, the synchronization operator gives the list of solutions with criterion equals to *ub* to the Branch-and-Bound. The Branch-and-Bound can then quickly explore the corresponding nodes. This has two advantages:

– it removes these solutions of the stack and forces the genetic algorithm to search in other areas of the search space
– creating child nodes of all the nodes that are in the path that lead to these solutions introduce in the stack some nodes in the neighborhood of the best genetic algorithm solutions that can potentially lead either the Branch-and-Bound or the genetic algorithm to better solutions.

Exploration Strategy: We have tried to use the depth first search. In classical Branch-and-Bound, this strategy has the advantage of finding feasible solutions quickly and to keep a reasonable stack size. But for our method, it may not be the best strategy because all the nodes of the stack are within the same area of the search space and it does not help the genetic algorithm much.

So, we have implemented another strategy: the best first strategy that takes the node that has smallest lower bound and within the nodes that have equals lower bound, it takes the deepest one. This has the advantage of keeping a large

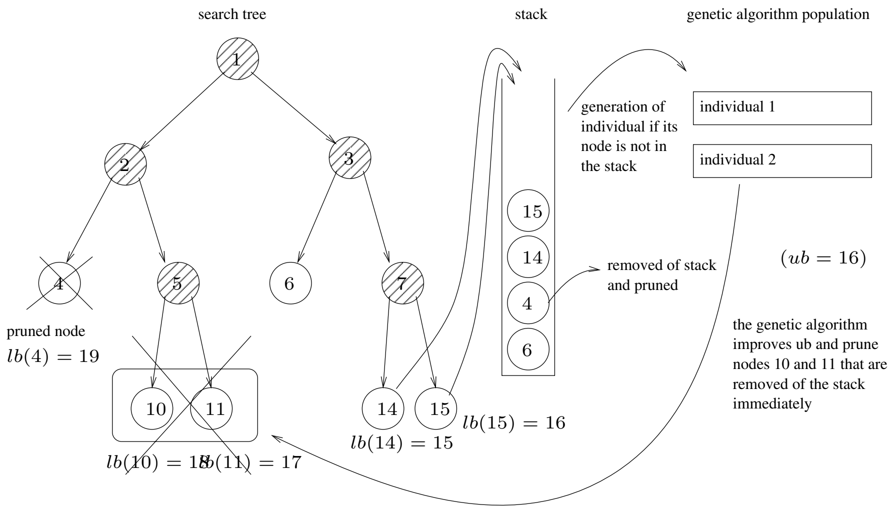

search tree
stack
genetic algorithm population
1
2
3
4
5
6
7
10
11
14
15
pruned node
$lb(4) = 19$
$lb(10) = 18$
$lb(11) = 17$
$lb(14) = 15$
$lb(15) = 16$
15
14
4
6
generation of individual if its node is not in the stack
individual 1
individual 2
removed of stack and pruned
$(ub = 16)$
the genetic algorithm improves ub and prune nodes 10 and 11 that are removed of the stack immediately

Fig. 7. Synchronization of the stack with the genetic algorithm and effect of the genetic algorithm on the Branch-and-Bound

enough stack that contains nodes that can be within all areas of the search space: it gives a large choice for the genetic algorithm that can be more efficient. Another advantage of this strategy is that it can improve the lower bound of the overall problem and it is an information that can be given to the decision maker to let him decide if it is worth continuing to search the optimal for very long to solve problem. The idea behind this strategy is to use the Branch-and-Bound to cut as many nodes as possible and to finally prove that we have found the optimal while the task of searching good solutions is done by the genetic algorithm.

Notice that we do not have explicit lower bound but a lower bound can be computed using cutting condition: let us consider D^* the smallest value of D for which the cutting condition answers 'no', then $D^* + 1$ is a valid lower bound. Then to compute a valid lower bound at node N different values of D are tested starting from the lower bound of the parent nodes of N. At the root node lb is computed using binary search on D.

Moreover when the synchronization operator is called, it sends to the method that handles nodes priority all the individuals that have a criterion equals to the best solution found until now. The Branch-and-Bound will then immediately explore these nodes. This way, they are removed from the stack and the genetic algorithm will be forced in the next synchronization to search other solutions than these best ones.

4 Experimental Results

The method is coded using Java language and was testing on a monoprocessor machine equipped with a 1.7GHz centrino and 1Gb of RAM. The two algorithms run in their own thread meaning that the program would immediately benefit of a second processor on a multicore or multiprocessor machine.

Preliminary tests have shown that the following parameters give nice results: a population size of 500 individuals, $pm = 30\%$, $pm2 = 5\%$ and $maxIt = 2000$. The method is limited to 10 minutes. The tests have been done on 2000 generated instances with a number of tasks between 10 and 45 and a number of operators between 2 and 10. These have been generated such that they have a similar skill repartition and similar values than what is found in industrial instances. In our industrial case, instances usually have between 30 and 40 tasks. In the results table, we have used the following abbreviation: **bb** means Branch-and-Bound only, **df** means hybrid method with depth-first search and **bf** means hybrid method with best-first search. When tested alone, the Branch-and-Bound is using a depth first search because using a best first search leads to stack size explosion due to the poor upper bound.

We can see on the table 1 that the hybrid method with depth first search is nearly always as good or better than the Branch-and-Bound alone despite the fact that in the hybrid method less nodes can be explored as the processor is shared by both method. Between the two hybrid methods, it looks like the best first search is better for large instances with higher differences starting from

Table 1. Percentage of instances solved to optimality

	n=10			n=15			n=20			n=25			n=30			n=35			n=40			n=45		
m	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf
2	100	100	**100**	100	100	**100**	100	100	**100**	100	100	**100**	100	100	97	97	97	**97**	97	97	**97**	83	83	**83**
3	100	100	**100**	100	100	**100**	100	100	**100**	100	100	97	97	100	97	80	87	**93**	70	70	**77**	50	47	**77**
4	100	100	**100**	100	100	**100**	100	100	97	80	87	**90**	73	73	**80**	50	50	**57**	43	43	**47**	27	33	**50**
5	100	100	**100**	100	100	**100**	83	83	**83**	77	80	77	47	47	**63**	40	43	**57**	23	20	**27**	3	7	**37**
6	100	100	**100**	100	100	**100**	90	93	**93**	60	63	**67**	23	23	**37**	20	23	**27**	3	3	**7**	6	7	**7**
7	100	100	**100**	97	100	97	60	70	60	60	60	57	13	13	**20**	3	13	**20**	3	7	3	0	7	**7**
8	100	100	**100**	93	93	83	47	50	**60**	33	33	30	17	20	**23**	13	13	**13**	3	7	3	6	7	**7**
9	100	100	**100**	90	93	90	53	53	37	23	27	**30**	12	3	0	3	7	**7**	0	3	**7**	3	3	**3**
10	100	100	**100**	90	93	90	13	17	13	10	13	**13**	7	7	3	3	3	0	0	3	**3**	3	3	**3**

Table 2. Average gap between lower and upper bound

	n=10			n=15			n=20			n=25			n=30			n=35			n=40			n=45		
m	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf	bb	df	bf
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.03	.01	.01	.01	.05	.05	.05	.3	.3	.3
3	0	0	0	0	0	0	0	0	0	0	0	.01	.2	.1	.02	.5	.5	.09	.7	.7	.4	1.7	1.7	.2
4	0	0	0	0	0	0	0	0	.02	.6	.6	.1	.9	.9	.2	1.9	1.9	.6	2.7	2.7	.5	2.9	2.9	.6
5	0	0	0	0	0	0	.9	.9	.5	.9	.8	.3	2.7	2.7	.4	3.1	3.1	.6	3	3	1	5.5	5.5	1.6
6	0	0	0	0	0	0	.4	.4	.1	2	2	.7	3.7	3.7	1.1	4.7	4.7	1.4	7.7	7.7	2.3	6.3	6.3	2.1
7	0	0	0	3	3	.03	.6	5.6	1.6	3	3	2	6	6	2.9	6.4	6.4	2.1	7	7	3.4	8	8	3.7
8	0	0	0	5	5	2	11	11	6.3	5.1	5.1	2.5	10	10	3.5	7.9	7.9	3.3	8	8	4.2	9.1	9.2	4.2
9	0	0	0	4	3	1.5	13	13	8.2	15	15	11	9.4	9.4	4.8	8.8	8.8	5.2	9.6	9.6	4.5	10.5	10.5	5.5
10	0	0	0	4	3	1.5	25	25	13	28	28	14	14	14	5.9	9	9	5.5	11	11	5.3	10.4	10.4	6.1

n=30. This corresponds to the size of the instances found in our industrial case.

On the other hand, the table 2 shows the gap in percentage between the lower bound of all the nodes remaining in the stack and the upper bound. It shows that the hybrid method with depth first search rarely improves the gap but best first search improves it significantly. It is mainly due to two factors. The first factor is that the diversity of the stack content gives more chance to the genetic algorithm to improve its best solution than the local search done with depth first search. The second factor is that with the best first strategy, the lower bound of the remaining nodes is naturally improved over time.

5 Conclusion

We have presented how a genetic algorithm can be combined with a Branch-and-Bound method in order to improve both methods. From the point of view of the genetic algorithm, the Branch-and-Bound is there to reduce its search space. From the point of view of the Branch-and-Bound, the genetic algorithm helps to improve the upper bound and thus to prune more node earlier. The results are promising for this method especially when used with a best first search. Moreover, these tests have been done on a monoprocessor machine and meaning it is very promising in the case of multicore processors.

References

[2005] Basseur, M., Lemesre, J., Dhaenens, C., Talbi, E.G.: Cooperation between Branch and Bound and Evolutionary Approaches to solve a BiObjective Flow Shop Problem. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 72–86. Springer, Heidelberg (2005)

[1987] Carlier, J.: Scheduling jobs with release dates and tails on identical parallel machines to minimize the makespan. European Journal of Operational Research 127, 298–306 (1987)

[1995] Cotta, C., Aldana, J.F., Nebro, A.J., Troya, J.M.: Hybridizing genetic algorithms with Branch and Bound techniques for the resolution of the TSP. In: Pearson, D.W., Steele, N.C., Albrect, R.F. (eds.), Artificial Neural Nets and Genetic Algorithms. Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, Ales, France, pp. 277–280 (1995)

[2001] French, A.P., Robinson, A.C., Wilson, J.M.: Using a Hybrid Genetic-Algorithm/Branch and Bound Approach to Solve Feasibility and Optimization Integer Programming Problems. Journal of Heuristics 7, 551–564 (2001)

[1978] Garey, M.R., Johnson, D.S.: Strong NP-Completeness results: motivations, examples, and implications. Journal of the ACM 5(3), 499–508 (1978)

[2002] Gharbi, A., Haouari, M.: Minimizing makespan on parallel machines subject to release dates and delivery times. Journal of Scheduling 5, 329–355 (2002)

[2005] Gharbi, A., Haouari, M.: Optimal parallel machines scheduling with availability constraints. Discrete Applied Mathematics 148, 63–87 (2005)

[1975] Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press, Michigan (1975)

[2005] Jouglet, A., Sevaux, M., Oguz, C.: Flowshop hybride: de nouvelles perspectives en mêlant algorithme génétique et propagation de contraintes. In: ROADEF 2005, Congrés de la Société Française en Recherche Opérationnelle, Tours, France (2005)

[1978] Lawler, E.L., Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming. Journal of the ACM 25(4), 612–619 (1978)

[2006a] Pessan, C., Néron, E., Bellenguez-Morineau, O.: Modélisation et planification des opérations de réglage de machines lors de changements de série. In: Gourgand, M., Riane, F. (eds.) MOSIM 2006 conference, vol. 2, pp. 1545–1554 (2006)

[2006b] Pessan, C., Bouquard, J.-L., Néron, E.: An unrelated parallel machines model for an industrial production resetting problem. European J. Industrial Engineering 2(2), 153–171 (2008)

[1998] Portman, M., Vignier, A., Dardilhac, D., Dezalay, D.: Branch and Bound crossed with GA to solve hybrid flowshops. Eur J Oper Res 107, 389–400 (1998)

[2005] Puchinger, J., Raidl, G.R.: Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, vol. 3562, pp. 41–53 (2005)

[2004] Rocha, P.L., Ravetti, M.G., Mateus, G.R.: The meta-heuristic grasp as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times. In: EU/ME Workshop, Nottingham, UK (2004)

Aerodynamic Topology Optimisation Using an Implicit Representation and a Multiobjective Genetic Algorithm

Windo Hutabarat, Geoffrey T. Parks, Jerome P. Jarrett, William N. Dawes, and P. John Clarkson

Engineering Design Centre, Department of Engineering
University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
{wh226,gtp,jpj1001,wnd,pjc10}@eng.cam.ac.uk

Abstract. Given the focus on incremental change in existing empirical aerodynamic design methods, radical, unintuitive, new optimal solutions in previously unexplored regions of design space are very unlikely to be found using them. We present a framework based on an implicit shape representation and a multiobjective evolutionary algorithm that aims to produce a variety of optimal flow topologies for a given requirement, providing designers with insights into possibly radical solutions. A revolutionary integrated flow simulation system developed specifically for design work is used to evaluate candidate designs.

1 Introduction

Fluid dynamics is a complex and nonlinear discipline. Predicting the behaviour of aerodynamic objects is not easy. Hence, aerodynamic design processes are rarely started from first principles. Initial decisions in aerodynamic design are usually based on empirical knowledge [1]. However, it can be argued that, in some areas, aerodynamic designers have approached the point where using an overly derivative approach may lead to some basic designs being used outside of their optimal region. We shall illustrate this point with an example.

The Grid Fin Example. The wing is the solution most commonly used whenever the generation of lift is required. By continuous incremental improvement to its basic shape and working principle, there are now many derivatives of wings which are widely applicable. However, today's increasingly demanding design requirements may uncover situations in which none of these variations is optimal; indeed, the situation may be such that a radically different shape is necessary.

Figure 1 shows an advanced air-to-air missile equipped with a novel type of lifting surface called *the grid fin*. This trellis-like contraption at the tail of the missile has characteristics that happen to be very well suited to the demanding requirements of supersonic dogfighting [2]. It is clear that using grid fins instead of wing derivatives contributes to the fact that this particular missile (the Vympel R-77) is currently widely held to be the premier supersonic dogfighting missile.

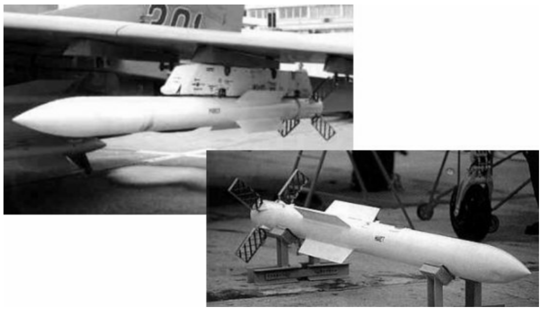


Fig. 1. Grid fins on the Vympel R-77 missile [**?**]

1.1 Topology as a Design Variable

The difference between the wing and the grid fin is greater than can be captured by the concept of *shape*; they are said to be different in *topology*. Due to the high cost of change, it is important to select the "right" topology early in the design process. Ideally, therefore, topology should be one of the key variables determined during the conceptual design phase. However, the authors are not aware of any method or tool that designers can use early in the design process to explore alternative aerodynamic topologies. The topology of a design is therefore usually predetermined before the design process even begins.

One possible solution is suggested by the use of simulation-based multiobjective optimisation to produce optimal designs and parameter trade-offs [4,5]. If such a system can be used to explore the topology trade-off, its output may help designers gain insights into radical and previously unconsidered options.

1.2 Proposed Framework

We aim to demonstrate that, for a given set of requirements, a framework consisting of a *stochastic, multiobjective optimiser* using a *topologically unconstrained shape representation* and coupled with a *robust CFD evaluator* is able to consistently identify a variety of solution flow topologies, which will hopefully provide designers with insight into the available topological trade-offs.

A stochastic optimiser is widely recognised to have the following characteristics:

1. The ability to incorporate practically any sort of design objectives;
2. The ability to treat evaluation codes as black-boxes;
3. The ability to escape local optima;
4. Easy parallelisation.

We consider these characteristics to be more suitable in the information-starved, early stages of a design process than alternative approaches, despite the usual perceived disadvantage of this choice: that of demanding a large number of objective function evaluations.

We have demonstrated the viability of a basic version of this proposed framework in an earlier study [6]. Our current work extends this concept to introduce multiobjective optimisation, continuous surfaces, and a proper CFD code into the framework.

The availability of a suitable CFD flow solver is absolutely crucial. We are grateful to Cambridge Flow Solutions for providing us with BoXeR, a newly-developed CFD package with exactly the sort of capabilities that we need [7].

2 Related Work

Shape optimisation in fluids has been the subject of intense research [8], and there are various ways in which CFD can be used for this purpose [9]; combining CFD with stochastic optimisation algorithms has proven to be quite a successful approach [5].

Nevertheless, research in topology optimisation in fluids has only started very recently with the pioneering work of Borrvall and Petersson [10]. This work is based on the material distribution approach, a well-known method of structural topology optimisation [11]. The state of the art of this approach is summarised in Gersborg-Hansen's thesis [12]. As yet, however, this approach has not been demonstrated using a finite-volume Navier-Stokes discretisation, which is the best established approach in CFD.

Another approach [13] can be viewed as an extension of the Evolutionary Structural Optimization method [14]. This approach adds or removes Boolean cells according to a set of flow-based optimality criteria.

Our approach is built on previous work in Genetic Algorithm (GA)-based structural topology optimisation. Our previous work [6] can be seen as an extension of voxel-based GA structural topology optimisation work [15], while the present work can be viewed as an extension of Kita and Tanie's work [16].

One topologically unconstrained way of representing shapes is by using an implicit representation. Examples of shape optimisation using implicit shape representation are provided by the level-set community, such as [17].

The Radial Basis Function (RBF) has been recognised as an efficient way of storing implicit representations. It has been shown to be compatible with topological optimisation [18]. One of the chief objections to implicit representations is that they cannot represent complex details without running foul of the curse of dimensionality. However, methods exist that enable an RBF representation to represent shapes of nearly arbitrary complexity [19].

The development of BoXeR was first reported in [20]. Since it is not yet commercially deployed, further information may be found on the Cambridge Flow Solutions website [7].

3 Framework Implementation

3.1 Topologically Unconstrained Shape Representation

The literature on topology optimisation suggests three main topologically unconstrained representation methods:

1. Binary occupancy: [15] in structures and [13] in fluids
2. Material distribution: [11] in structures and [10] in fluids
3. Implicit representation: [17] in structures and the present work in fluids

In our previous work [6] we used a simple binary representation scheme, which unfortunately is too expensive to use to model continuous surfaces. The second method, material distribution, requires very extensive modification of the flow simulation, which is not a realistic option.

In the present work we use an implicit representation with information stored in an RBF equation. In an implicit representation, an object Ω with boundary ω is defined as the set of points $\mathbf{x}$

$$\{\mathbf{x} : s(\mathbf{x}) = p\}, \ \mathbf{x} \in \omega \tag{1}$$

where p can be set to any scalar value; usually $p = 0$.

RBFs have been used by several research groups as representation methods of solids and surface interpolators [21]. An RBF interpolation process constructs an implicit function $s(\mathbf{x})$ by using a set of control point coordinates $\mathbf{x}_i$ and function value s_i at $\mathbf{x}_i$. We shall briefly describe our RBF implementation.

RBF Construction. An RBF is basically a weighted sum of a given basis function that is evaluated over the distances of all pairs of a set of control points. More elaborately, an implicit function $s(\mathbf{x})$ is expressed as

$$s(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(|\mathbf{x} - \mathbf{x}_i|), \ \mathbf{x} \in \mathbf{R}^d \tag{2}$$

where $\mathbf{x}_i$ are the locations of the control points, N is the number of control points, λ_i is the weight for control point $\mathbf{x}_i$, $\phi(r_i)$ is the basis function, and $||$ is the Euclidean norm in $\mathbf{R}^d$. The basis function $\phi(r_i)$ is usually chosen from a family of well-known functions, such as the thin-plate, Gaussian, multiquadric, etc. The multiquadric basis function

$$\phi(r_i) = \sqrt{\epsilon \cdot r_i^2 + 1} \tag{3}$$

is quite widely used. Here ϵ is an adjustable constant.

The RBF $s(\mathbf{x})$ is completed by calculating the weights λ_i. If we define

$$\mathbf{\Phi} = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix}, \text{ where } \phi_{ij} = \phi(|\mathbf{x}_i - \mathbf{x}_j|) \tag{4}$$

$$\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)^T \tag{5}$$

$$\mathbf{S} = (s_1, s_2, \ldots, s_N)^T \tag{6}$$

then the approximation process is performed by solving the linear system of equations:

$$\mathbf{\Phi} \cdot \mathbf{\Lambda} = \mathbf{S} \tag{7}$$

The function $s(\mathbf{x})$ can then be used as an implicit shape representation.

Initial solutions can be generated by generating control points – either randomly or uniformly – and then giving each control point a random scalar value. The embedded shape can be manipulated either by changing the values of the control point scalars or by addition/removal of control points. Should we choose to change only the scalar values and fix the number and locations of the control points, we can then use standard real-valued genetic search operators. Prior to shape evaluation, the embedded shape can be extracted from the implicit function by a variety of methods, such as the exhaustive Marching Cubes method.

In the present work, we chose to limit ourselves to 2D implicit surfaces, with the control points spread in a regular 2D lattice. The embedded shapes are edited by changing the nodal scalar values. The shape generation method then resembles the generation of an elevation map, and shape manipulation changing the elevation of some nodes. The steps required to create a shape, shown schematically in Figure 2, are as follows:

(a) A regular $I \times J$ lattice of N control points is created.
(b) Each control point is associated with a scalar value s, where $s_{\min} < s < s_{\max}$. In Figure 2(b) the scalar value is plotted as a height map.
(c) A RBF implicit function is created according to the control points.
(d) The embedded shape can be extracted as an isoline of a specified value $s = s_\omega$.

In most cases, the generated isolines will not form closed shapes, intersecting the shape generation boundary instead. We can choose whether to leave these

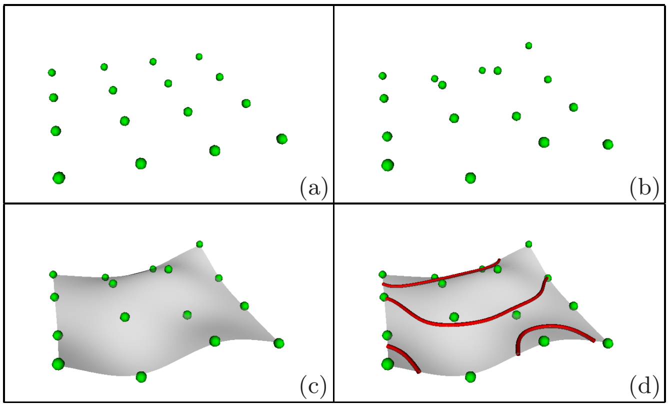


Fig. 2. Implicit representation of a 2D object. The end result of this process is the embedded shape, shown as the red line in (d). Prior to evaluation, the red lines will be extruded into surfaces as shown in Figure 4.

shapes open or to close them with some repair mechanism. In the present implementation we opt for allowing non-closed shapes; this has some important consequences that will be encountered later.

3.2 Optimisation Algorithm

For the optimisation algorithm we chose the well-known multiobjective GA NSGA-II [22] due to its demonstrated capability in flow geometry optimisation [23] and in unveiling innovative design principles [4].

Design Variables. The design vector **s** is the vector of scalar values stored in each interpolation node. These are standard real-valued design variables; hence the special crossover and mutation operators that we originally proposed [6] are currently not necessary. We limit the scalar values of the control points to the range $-1 < s < 1$, and we extract the shape boundary at $s = 0$.

Genetic Operators. Since no special operators are necessary, the crossover and mutation operators can be chosen from the wide range of real-valued GA operators available in the literature. With ease of implementation as our main criterion, we chose Parent-centric Normal Crossover (PNX) [24] and the standard Gaussian or "normal" mutation operator [25] in preference to the simulated binary crossover (SBX) and polynomial mutation operators suggested in the original NSGA-II paper [22]. We have used these operators with our initial test cases and found their performance to be satisfactory.

3.3 Evaluation

BoXeR represents a new approach to the use of flow simulation in the design process. Recognising the bottlenecks of CFD usage, Cambridge Flow Solutions wraps revolutionary features around their industry-standard "NEWT" code, a finite-volume Navier-Stokes flow solver, producing an integrated and parallel geometry kernel, mesh generator, Cartesian flow solver, and post-processor [7]. Among the revolutionary features of BoXeR, our work benefited most from its automated mesh generation capability.

The success of discretised flow simulation is highly dependent on the quality of its mesh. CFD meshes are usually built starting from existing CAD models, which are rarely designed with CFD use in mind [26]. This makes the process laborious and error-prone, quite unsuited to an automated optimisation system. BoXeR's integrated CAD-importing and mesh generation tools do away with this problem. The BoXeR user can simply define the domain, the geometry, and the flow conditions, then step back and watch as the system automatically meshes, refines, and iterates towards convergence in real-time.

4 Test Case

In this test case we set up a simple optimisation problem and see if our framework achieves what we seek to demonstrate. We first put the shape generation box

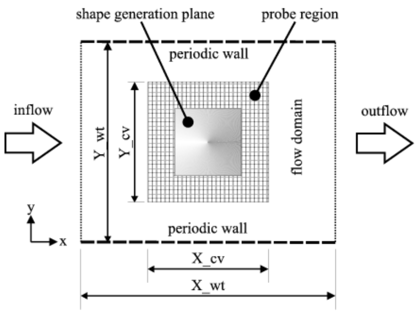


Fig. 3. Virtual wind tunnel dimensions

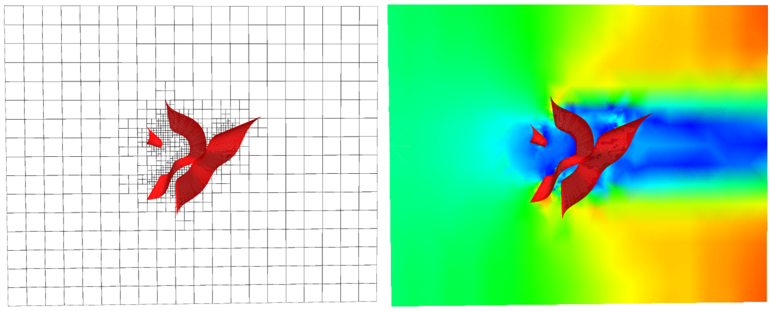


Fig. 4. BoXeR octree domain discretisation, and a Mach number colour-mapping visualisation of the simulated flow. The curvy surfaces in the middle is the extruded shape.

into a BoXeR virtual wind tunnel, as shown in Figure 3. BoXeR is then run for a given number of iterations, and the resulting flow field is processed to extract the objective function values. We wish to see whether:

1. The framework is consistently able to produce Pareto fronts.
2. The resulting Pareto front is composed of a variety of flow topologies.
3. The resulting shapes are "aesthetically pleasing". We shall use this catch-all clause to direct our next efforts.

Tables 1 and 2 show the parameters used in the GA and the virtual wind tunnel, respectively. The GA parameters in Table 1 are those suggested in [22], which

Table 1. Optimisation parameters

Number of control points	$I = J = 8$	Basis function	Multiquadric
Control point scalar bounds	$s = [-1.0, 1.0]$	Shape boundary value	$s_\omega = 0.0$
Number of individuals	100	Crossover rate	0.85
Mutation rate	0.033	Number of generations	100

Table 2. BoXeR wind tunnel parameters

Wind tunnel dimensions	$X_{wt} = 10\,m, Y_{wt} = 8\,m, Z_{wt} = 1\,m$
Control volume dimensions	$X_{cv} = 5\,m, Y_{cv} = 4\,m, Z_{cv} = 0.5\,m$
Test generation plane dimensions	$X_g = Y_g = 1\,m$

Specific heat capacity	$c_p = 1005\,J\,kg^{-1}K^{-1}$	Heat capacity ratio	$\gamma = 1.4$
Kinematic viscosity	$\mu = 1 \times 10^{-5} m^2 s^{-1}$	Inlet total pressure	1×10^5 Pa
Inlet total temperature	288 K	Inlet static pressure	7.56×10^4 Pa

were found to be satisfactory after a small number of test runs. In Table 2, the dimensions of the virtual wind tunnel are chosen to model a comfortably large wind tunnel with a 2 m by 2 m test section. The last six parameters in Table 2 define the flow to be under atmospheric, subsonic conditions, a regime of great interest but free of the complications found in transonic or supersonic flow.

4.1 Objective Functions

The design objective is simple: given a uniform, horizontal incoming fluid flow, what kind of topology will produce maximum upward momentum with the least loss in horizontal momentum? This crudely approximates the lift-drag trade-off of a downforce generator on a racing car.

Translating this to the optimisation framework, the objective evaluator is then a post-processor that integrates the momentum fluxes within a control volume $\mathbf{p}_{cv}$. We define the control volume as a 3D hexahedron centrally surrounding the shape generation plane, having dimensions exactly half the corresponding dimensions of the simulation domain. The momentum integration is

$$\mathbf{p}_{cv} = m_{cv} \cdot \mathbf{u}_{cv} = \sum_{i=1}^{N_{cv}} \rho_i \cdot V_i \cdot \mathbf{u}_i \tag{8}$$

where N_{cv} is the number of cells within the control volume, V_i is the volume of cell i, and ρ_i and $\mathbf{u}_i$ are the density and velocity of the fluid in that cell.

A large vertical momentum means that the structure performs well in deflecting flow upwards, while a large horizontal momentum means that the design creates minimal drag. Hence, if a given design variable vector $\mathbf{v}$ produces $\mathbf{p}_{cv} = (p_x, p_y)^T_{cv}$, then the two objectives to be *minimised* are

$$f_{objective_1}(\mathbf{v}) = -p_x \tag{9}$$

$$f_{objective_2}(\mathbf{v}) = -p_y \tag{10}$$

4.2 Selection Operator

To maintain population diversity, NSGA-II applications typically use a crowded tournament selection operator working in objective function space. We use a design-space-based crowding operator instead, since our initial experiments suggest that, while this operator causes the system converge slower, the final population usually has greater topological variety.

5 Results and Discussion

Figure 5 shows the entire population of the final generation of a typical test case run. This entails 10,000 separate runs of the CFD code with 2000 iterations towards convergence each. Each test case run takes approximately 24 hours to run on a cluster of twelve Opteron nodes.

Figure 6 shows a selection of shapes that result from one optimisation run. It is important to remember that in aerodynamic shape optimisation the objective function is extracted not from the shape of the material but from the shape of the void, or, more accurately, from the flow that is affected by the shape of the void regions.

We see here a collection of shapes that importantly create flows with different topological characteristics. The results range from the singular shape that performs minimal flow manipulation, to a strongly curved shape that has more effect on the flow, to multiple curved shapes that multiply the effect of the singular curved shape. In short, we see that the optimisation system has created varying topologies generating flow patterns that minimise the two conflicting objectives to varying degrees.

The shapes still have quirky undulations which may be smoothed if we continue the optimisation further, but we feel that, at this stage, it is more important to prove that the system can find "embryonic" solution topologies.

Apparently, our allowing non-closed shapes has resulted in a final population consisting of aesthetically unpleasing plate-like shapes. Apart from the obvious (but not modeled) structural problems associated with such structures, the optimiser misses out on the opportunity of finer flow control useful in the latter stages of the optimisation, since it has no separate control of the two sides of material in contact with the flow. This, however, suggests a hybrid optimisation approach, where these results are then used as the initial points for a more conventional shape optimisation.

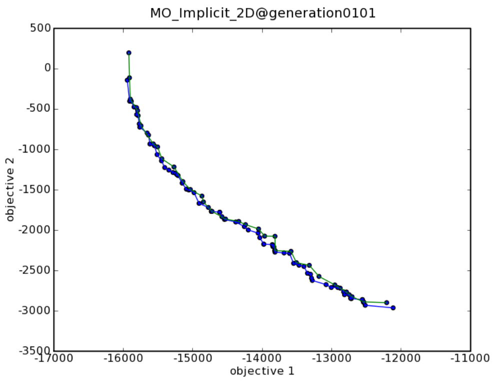


Fig. 5. A typical Pareto front from the test case The lines connects solutions belonging to the same Pareto rank as defined by NSGA-II, this particular one has two ranks

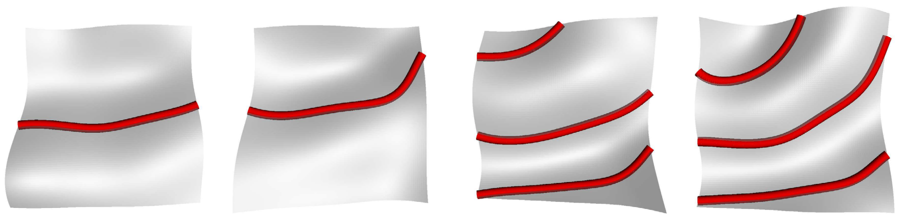


Fig. 6. Some samples from the Pareto front. The fluid flows from left to right, and is deflected upward. The carpet-like surfaces directly facing the viewer are the visualisation of the RBF interpolation surface, the thick lines represents s_ω isolines. Extruded surface is omitted for clarity. The objective function values are, respectively: $(-15769, -685)$, $(-14610, -1779)$, $(-13963, -2075)$, and $(-12556, -2862)$.

Comparing the results with our list of objectives given in Section 4, we found that in the limited number of tests we have done, the framework is consistently able to produce the Pareto front for the given test case. Our first objective is thus achieved.

As can be seen from Figure 6, the Pareto front consists of shapes with variable topologies, albeit non-closed. This demonstrates the ability of our framework to produce a trade-off across flow topologies. This means that the second objective is also achieved.

However, with respect to the third objective, we found the resulting non-closed shapes aesthetically unpleasing. This suggests that the optimisation problem can be better defined to produce aesthetically more pleasing closed shapes. One way to tackle this is by placing a constraint on non-closed shapes. We can do this by exploiting the fact that non-closed shapes will have a sign change on the interpolated scalar field boundaries. The extent of these scalar sign changes can be used as a measure of the extent of constraint violation.

6 Conclusions and Future Work

Our previous work has shown that GAs can be used to perform fluidic topology optimisation. Our present work seeks to demonstrate that a multiobjective GA, driving a topologically unconstrained representation, coupled with a proper CFD code, is able to come up with a solution topology trade-off. Most of our objectives have been achieved, with the remaining objective providing the impetus for further improvement.

In the future, we aim to do the following:

– Implement an objective function that extracts the non-dimensional aerodynamic characteristics of a shape. This will provide a comparative figure of merit that is better and more familiar to aerodynamic designers.
– Study the effects of the dimensions of the virtual wind tunnel.
– Further exploration of the robustness of the GA operator parameters.

Acknowledgments

The support of Cambridge Flow Solutions and of the UK Engineering and Physical Sciences Research Council (EPSRC) under grants GR/R64100/01 and EP/E001777/1 is gratefully acknowledged.

References

1. Giles, M.B.: Aerospace Design: a Complex Task. Technical Report 97/07, Oxford University Computing Laboratory (1997)
2. Fluent Inc.: First Viscous Analysis of Grid Fins Gives Better Prediction of Missile Trajectory. JA136 Journal Articles by Fluent Software Users, Fluent Inc. (2001)
3. Scott, J.: Missile Grid Fin (2006) (Last checked on September 15th, 2007), `http://www.aerospaceweb.org`
4. Deb, K.: Unveiling Innovative Design Principles by Means of Multiple Conflicting Objectives. Engineering Optimization 35(5), 445–470 (2003)
5. Sasaki, D., Obayashi, S., Nakahashi, K.: Navier-Stokes Optimization of Supersonic Wings with Four Objectives Using Evolutionary Algorithm. Journal of Aircraft 39 (2002)
6. Hutabarat, W., Parks, G.T., Jarrett, J.P., Clarkson, P.J.: A New Approach to Aerodynamic Topology Optimization. In: Parmee, I. (ed.) Adaptive Computing in Design and Manufacture, vol. 2006 (2006)
7. Cambridge Flow Solutions: BoXeR (2007) (Last checked on September 15th, 2007), `http://www.cambridgeflowsolutions.com`
8. Mohammadi, B., Pironneau, O.: Applied Shape Optimization for Fluids. Oxford University Press, Oxford (2001)
9. Jameson, A., Vassberg, J.C.: Computational Fluid Dynamics for Aerodynamic Design: Its Current and Future Impact. In: 39th AIAA Aerospace Sciences Meeting and Exhibit, AIAA (2001)
10. Borrvall, T., Petersson, J.: Topology Optimization of Fluids in Stokes Flow. International Journal for Numerical Methods in Fluids 41, 77–107 (2003)
11. Bendsoe, M., Kikuchi, N.: Generating Optimal Topologies in Structural Design Using a Homogenization Method. Computer Methods in Applied Mechanics and Engineering 71(2), 197–224 (1988)
12. Gersborg-Hansen, A.: Topology Optimization of Flow Problems. PhD thesis, Technical University of Denmark (2007)
13. Häussler, P., Nitsopoulos, I., Sauter, J., Stephan, M.: Topology and Shape Optimization Methods for CFD Problems. In: 24th CADFEM Users' Meeting 2006, International Congress on FEM Technology (2006)
14. Xie, Y., Steven, G.: Evolutionary Structural Optimization. Springer, Heidelberg (1997)
15. Kane, C., Schoenauer, M.: Topological Optimum Design Using Genetic Algorithms. Control and Cybernetics 25(5), 1059–1088 (1996)
16. Kita, E., Tamaki, T., Tanie, H.: Topology and Shape Optimization of Continuum Structures by Genetic Algorithm and BEM. Computer Assisted Mechanics and Engineering Sciences 11, 63–75 (2004)
17. Allaire, G., Gournay, F.D., Jouve, F., Toader, A.M.: Structural Optimization Using Topological and Shape Sensitivity Via a Level Set Method. Technical report, Ecole Polytechnique (2004)

18. Wang, S., Wang, M.: Radial Basis Functions and Level Set Method for Structural Topology Optimization. International Journal for Numerical Methods in Engineering 65(12), 2060–2090 (2005)
19. Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., McCallum, B., Evans, T.: Reconstruction and Representation of 3D Objects with Radial Basis Functions. In: Conference on Computer Graphics and Interactive Techniques, pp. 67–76 (2001)
20. Dawes, W.N.: Building Blocks Towards VR-Based Flow Sculpting. In: 43rd AIAA Aerospace Sciences Meeting & Exhibit, AIAA (January 2005)
21. Turk, G., O'Brien, J.: Modelling with Implicit Surfaces that Interpolate. ACM Transations on Graphics 21, 855–873 (2002)
22. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. Evolutionary Computation 6(2), 182–197 (2002)
23. Hirschen, K., Schäfer, M.: A Study on Evolutionary Multi-objective Optimization for Flow Geometry Design. Computational Mechanics 37(2), 131–141 (2006)
24. Ballester, P.J., Carter, J.N.: An Effective Real-Parameter Genetic Algorithm with Parent Centric Normal Crossover for Multimodal Optimisation. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 901–913. Springer, Heidelberg (2004)
25. Fogel, D.B.: Mutation Operators. In: Evolutionary Computation I: Basic Algorithms and Operators, pp. 237–255. Institute of Physics Publishing, Bristol, UK (2000)
26. Kellar, W.P.: Geometry Modeling in Computational Fluid Dynamics and Design Optimisation. PhD thesis, University of Cambridge (January 2003)

Direct and Indirect Representations for Evolutionary Design of Objects

Juraj Plavcan[1] and Pavel Petrovic[2]

[1] Faculty of Mathematics, Physics, and Informatics, Comenius University, 84248 Bratislava, Slovakia,
[2] Department of Computer and Information Science NTNU, 7491 Trondheim, Norway
ppetrovic@acm.org

Abstract. We follow up on the work of Ebner[1] in studying representations for evolutionary design of objects. We adopt both the method and the simulation framework, and perform more thorough experiments. We design new representations, both direct and indirect, and compare their performance to the original work. We design and make use of a specialised system for distributed computing that integrates smoothly with the EO library[5]. First, we confirm the results of Ebner with VRML scene graphs representation. Next, we demonstrate how both of the new representations based on triangular mesh perform significantly better. Finally, we study and improve the performance of the distributed system that we utilised to run our experiments on tens of computing nodes.

1 Introduction

Evolutionary Design is a promising application area of the Evolutionary Computation (EC) with a large and yet to be discovered potential. Steadily more intricate and specialised designs are needed in the various technological fields, in the scale ranging from the space applications down to the nano level. In order to utilise EC for the design of parts and objects, effective ways of encoding the 3D shapes into genotype representations must be provided and evaluated. Straight-forward direct mapping of unit cells of a 2D or 3D grid into bits of a genotype suffers from huge search spaces, very localised search with operators that cannot span across local extremes, and inflexibility of the search in focusing on the most relevant locations while covering the large even areas with only a few data items. For instance, consider an application of evolving a can opener. Close attention must be paid to the shape of the sharp blade, while the handle can consist of one large cylinder. The genotype representation should be able to dedicate many genes/alleles to the blade and cover the handle with only a few. This distribution need not to be known in advance and should be discovered by the evolutionary algorithm.

Previously, EC has been applied to design various shapes. For example, Robinson et.al. [8] evolve structures for satellite boom with passive vibration control, i.e. shapes capable of cancelling unwanted vibration by the means of object topology. In [3], Hamda et.al. use indirect representations based on Voronoi diagrams, dipoles, and bars to represent 2D shapes for Topological Optimum Design. Hornby [4] is advocating for reuse of the design components (modules) represented by parts of genotype in particular by applying Lindenmayer systems, an instance of what they define as *generative representations*, where elements of genotypes are reused in their translations to phenotypes. In addition to reusing modules within the same single design, authors reuse modules across individual designs within design families. Ebner [1] addressed the challenge of indirect representations by adopting scene graphs as genotype representation of the shape of objects. Our work repeats Ebner's experiments and moves ahead by comparing them against two new representations. His representations are based on geometric 3D objects as building blocks that do not allow effective use of material and possibility to describe shapes accurately enough. We instead encode objects using a triangular mesh in two different ways: directly, by encoding the mesh node coordinates, and by generating the coordinates through a series of spacial transformations in a manner similar to Ebner's VRML representation. The second of our representations is generative.

Evolutionary experimental runs, demand extensive CPU resources. The runs would take very long to complete on a single computer. We therefore chose to develop a distributed version of the algorithm. The evolutionary engine runs on a master node, which is utilising the available slave nodes to evaluate the individuals in the population (the specific shapes). The evolutionary objective function measures the performance of shapes by simulation in a popular open-source body physics simulator ODE.

In the following sections, we describe the details of the ODE simulator, explain the scene graphs as used by Ebner, introduce the representations based on triangular mesh, unveil the details of our evolutionary algorithm, discuss the distributed system for fitness evaluations, present the results of the experiments and summarise the paper in conclusions.

2 Simulation

Physical simulation usually works with models of a continuous nature, where the state of the objects in the simulated world changes continuously. On the contrary, simulation of bank transactions typically has a discrete nature. In computer simulation, the continuous model is often approximated using a discrete model with small time steps: either as *fixed-increment time advance*, or as *next-event time advance*. We use the former, where the state of the system changes only in the instantaneous moment between short time intervals of constant duration.

2.1 Open Dynamics Engine

Several simulation engines have been developed (for instance Newton Dynamics™ and Havok Physics™). As Ebner, we adopted the open-source project Open Dynamics Engine [10] as the simulating platform. ODE is used for interactive simulation in real time in computer games and robotic simulators (SimRobot [6],

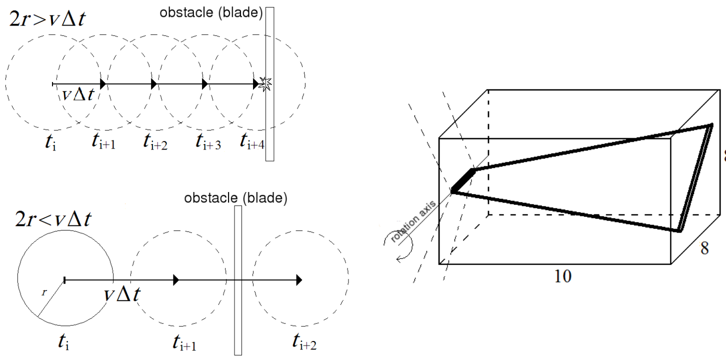


Fig. 1. Relation between the maximum feasible step size and wind velocity (left), reference plane and bounding prism for the rotor blades (right)

UCHILSIM [12], Ubersim [2], Webots [7]) and it prioritises the speed and stability over accuracy and correctness, however, it provides a realistic environment, where the performance of representations can be studied and evaluated. We use ODE for simulation of stream of air particles passing through and colliding with the blades of a wind turbine and causing its rotation in result.

ODE simulation consists of two main components, *Collision Detection Subsystem* and *Physics Engine Subsystem*. The latter works with rigid bodies, objects defined by their position, orientation, speed (linear and rotational), weight, and centre of gravity. The bodies can be isolated or connected with each other using joints to form complex objects. The former is responsible for maintaining the mutual position of the bodies, and for avoiding their overlapping. This requires further attributes, such as volume, shape, 'softness', and 'flexibility'. During a contact, a special type of joint (contact joint) is created for the time period of one simulation step. All joint types together determine the interactions of the simulated bodies and their outcome. The geoms can have simple shapes – box, sphere, capped cylinder, or more complex shapes – defined by a set of triangles that together form a triangular mesh. The joints between the bodies are of different types (ball, hinge, slider, fixed) and have corresponding various degrees of freedom.

Due to the discrete time intervals used, the bodies are "teleported" in time steps. This influences the size of the air particles, which must collide with the blades of the rotor, see Figure 1, and as a consequence the speed of the simulation.

2.2 Simulation Model

The rotor consists of three blades (each rotated by 120°), attached as one compound object to horizontal axis of rotation (represented by a capped cylinder) using a frictionless hinge joint. We enforced a constraint for the size of the blades, which ought to fit into a prism with dimensions 10x8x8 (figure 1). Given a specified wind strength (particle speed, weight and density), we observed the maximum angular velocity ω_{MAX}, the blade weight and its distribution (i.e. the distance of the blade's centre of gravity from the axis). Using these variables, we estimated the maximum reached kinetic rotational energy as:

$$E_{MAX} = \frac{1}{2} I \omega_{MAX}^2 = \frac{1}{2} 3mr^2 \omega_{MAX}^2.$$

The air particles were flying inside of a wind corridor with a circular profile. This cylinder had its bases parallel to the rotational plane. The particles of radius of 0.1 units were generated at the start of the wind corridor with starting velocity $v_0 = 10$. In each step, two forces were applied on the particles: the wind force $F_{wind} = 0.3$ with a vector parallel to the direction of the wind. The resistant force F_d with a vector parallel to the particle's speed vector, opposite orientation, and size linearly proportional to the particle speed:

$$\mathbf{F} = \mathbf{F_{wind}} + \mathbf{F_d} = \mathbf{F_{wind}} - \alpha \times \mathbf{v},$$

where $\alpha = 0.02$ is a friction coefficient and $\mathbf{v}$ is the speed vector. Thus the particles with starting speed $v_0 = 10$ will continue to move with constant speed and direction until they collide with a blade of the rotor, or reach the end of the corridor. At the collision, the particle gives up part of its kinetic energy in favour of the rotational energy of the rotor, and leaves the point of contact according to the law of reflection. Consequently, the wind force will act on the particle to slowly change its speed vector and direct it in the direction of the wind, accelerating to its maximum speed until it leaves the wind corridor or collides again with a blade of the rotor.

The parameters of the model include the size of the wind corridor and the number of air particles. Marc Ebner used a constant-size wind corridor with 100 particles, which were moved to the start of the corridor automatically when reaching the end of the wind corridor. Given the limited total number of steps in the simulation, and the complexity of the blades, we believe this number is too low.

We attempted a more accurate model of the wind. We used smaller time step (0.01 as compared to Ebner's 0.1) and more particles. We optimised the model by variable-length corridor (with the base radius 10.5). The length depending on the "depth" of the rotor. The starting base was placed 0.1 units ahead of the foremost point of the rotor, and the ending base was touching the very rear point of the rotor. The number of particles in cube unit of the corridor was constant (0.7), which in practice meant 500 - 2000 particles in total. The challenge was in generating wind with stable (regular) strength. We tried two approaches:

1. In each step of the simulation, all particles were active. Immediately after leaving the corridor, the particles were restarted. This approach achieves an almost constant total wind energy (sum of the kinetic energies of the particles) during the whole simulation. A disadvantage are large variations in the number of restarted particles in each step, which result in unwanted resonance and conflict with the criterium for terminating the simulation.

2. in each simulation step, a constant number of particles were started at the start of the corridor. The particles leaving the corridor were erased from the simulation. The number of starting particles in each step was determined empirically so that the number of particles in a cube unit was approximately 0.7. However, leaving this number constant throughout the simulation leads into a large variation in the total number of particles in the simulation and thus as well in the total wind energy. The number of particles depended extensively on the shape of the blades, and therefore was not comparable.

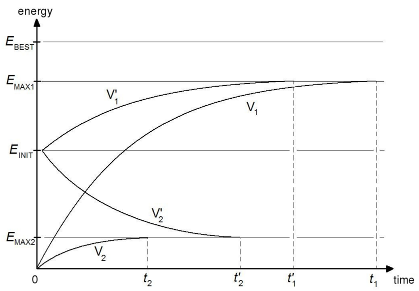
We chose the following compromise: we determined the total number of particles as in the first of the two approaches. However, we limited the maximum number of particles entering the corridor in each step ($restarts_{max}$). If too many particles started to leave the corridor, they had to wait in a backup stack until the number of particles arriving at the end of the corridor decreased under $restarts_{max}$ threshold. This limit was dynamically adjusted depending on the state of the stack: when the stack was growing above safe threshold (20), the limit increase was linearly proportional to the stack size. When it was shrinking (or empty), the limit was decreasing at constant rate. This implementation of "even" wind proved to be stable for different shapes of blades – the wind energy varied little and the wind resonance was balanced.

For a fixed set of wind parameters and specific rotor, there is a maximum angular velocity v_{MAX} it can reach. The goal of the simulation is to determine this speed, which together with the mass and its distribution implies the energy that the turbine can acquire from the wind. Accordingly, we derived the simulation termination criterium: the simulation executes until the angular rotor velocity converges, however always at least 500 and at most 2000 time steps. We consider the velocity converged when the averages over four different history windows vary less than δ.

In order to reduce the time required for the angular speed convergence, the rotors were assigned a nonzero starting kinetic energy, estimated as the half of the maximum kinetic energy reached by any shape currently present in the population. The rotors with lower performance decelerated, those with a higher performance accelerated. See figure 2 left for the effects of this optimisation.

3 Scene Graphs

Marc Ebner focused in his work on scene graphs and compared two existing representations: Open Inventor used in a 3D graphics library and VRML used for virtual reality modelling. Both representations have a tree structure. Internal nodes represent transformations, while leaves represent simple objects: sphere, capped cylinders or boxes. The resulting transformation of the leaf object in VRML representation is defined by gradual (in that order) application of transformations on the path from the tree root, while in the Open Inventor representation, the transformation nodes can be in terminal nodes too, and the transformations are influenced by special nodes (separators). The resulting transformation in Open Inventor is the composition of the transformations in the



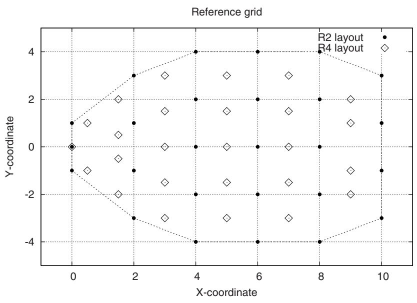


Fig. 2. Simulation time optimisation through initial rotor kinetic energy assignment (left). In most cases, the simulation time is decreased: original time t_1 has been decreased to t'_1. In some cases, the time increases: t_2 to t'_2. Layout for the blade for R_2 and R_4 (right).

order of searching the tree using depth-first search. The separators work as local accumulators and isolate the rest of the tree from the transformations within its subtree; the transformations in the subtree of a separator are applied only to the leaves inside of its subtree.

Ebner applied the two representations to represent the genotypes for the shape of the blade of the wind turbine and used the standard GP for evolving them in a population of 50 individuals, with the tournament selection of size 7, which is known in general to suggest a strong vulnerability to local optima (and we confirmed this also empirically in our early experiments). The parameters of transformations (translation vector, rotation axis and angle) and the leaves (the size of the placed objects) were initialised randomly and did not change during the evolution. This is in contrast with Ebner who used evolutionary strategy to evolve these values, while we preferred manual tuning so that the CPU cycles were used more efficiently during the runs. The standard tree crossover and mutation has been applied with the probability of 50%. Ten evolutionary runs of 200 generations were performed.

Ebner showed that VRML representation outperforms the Open Inventor representation significantly. He argued that the subtrees in VRML representation correspond to specific parts of the blade and thus the recombination and mutation of the root node in the subtree influences only these parts of the blade. The mutations in Open Inventor influence transformations of other subtrees and thus other parts of the blade.

4 Rotor Representations

4.1 Indirect VRML Representation with Objects R_1

We performed experiments with four different representations, R_1 - R_4. In the first, an individual is represented by a tree with branching factor 2–3. Maximal initial depth of tree is 4. Maximal size of tree is 100 nodes (inner nodes and leaves). Tree nodes are chosen from the set F

$$F = \{R_0, T_0, R_2, T_2, R_3, T_3\}, where$$
$$R_i(x, y, z, \alpha) \in (0; 3) \times (0; 1) \times (0; 1) \times (0; 2\pi), i = 0, 2, 3$$
$$T_i(d_x, d_y, d_z) \in (0; 3) \times (0; 1) \times (0; 1), i = 0, 2, 3$$

R_i represents rotation along axis (x, y, z) of angle α. T_i represents translation along vector (dx, dy, dz). R_0, T_0 are leaves and represent final rotation, and translation of capsule (building block of blade) respectively. Node types R_2, T_2 have two child nodes, R_3, T_3 three. Internal node parameters are initialised randomly and later modified by the mutation operator. The rotor (phenotype) is constructed in ODE environment by depth-first traversing of the corresponding tree (genotype). For every tree leaf, there is one capsule placed. Its position is transformed according to information located on the path from root of the tree to this leave. Capsules, which are not linked to the rotation axis of the rotor directly or via other capsules, are ignored and do not participate in simulation. If the final rotor blade is exceeding the dimensions of the bounding prism (10x8x8), then the individual's fitness is set to zero. Otherwise the simulation is started and kinetic (rotational) energy of rotor after completion of simulation represents the individual's fitness.

4.2 Direct Representation R_2

$$(\alpha, (z_1, b_1), (z_2, b_2), \ldots, (z_{26}, b_{26})), \alpha \in R(0; 1), z_i \in R(-4; 4), b_i \in \{0, 1\}$$

The basis for our direct representation is a reference grid (see Figure 2 right) containing 26 vertices. Coordinates (2D) of the vertices are fixed for all individuals. We set the total number of vertices and their position in reference grid by an expert guess.

Genotype defines elevation of each vertex relative to the reference grid. The elevation is limited, maximum +4 and minimum -4 units. Genotype contains presence bit for each vertex – 0 means the vertex is ignored, 1 means the vertex is part of the blade's shape definition (active vertex). Thus shape of the blade

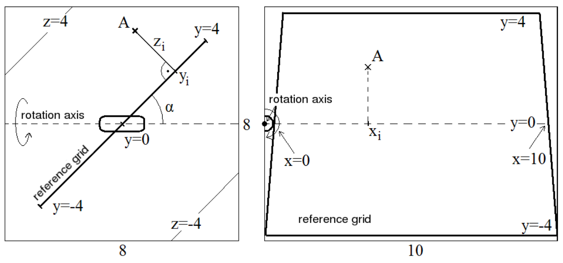


Fig. 3. Direct representation R_2

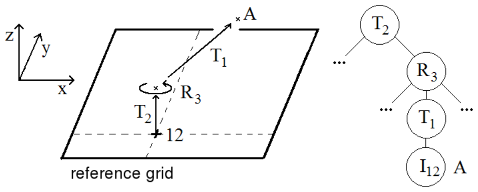


Fig. 4. Transformation in indirect representations R_3, R_4

is defined by triangular mesh over elevated active vertices of the reference grid. The last information present in the individual's genotype is an angle of skewness of the blade – the reference grid is rotated by this angle along x-axis.

During the initialisation and mutation of the individuals, dimensions of the corresponding blade are checked against the dimensions of the bounding prism (10x8x8). In the case the blade stretches out of this prism, the elevations of conflicting vertices are adjusted so that the whole blade fits inside the prism. We used Delaunay triangulation to determine a triangular mesh (set of triangles) from the set of active vertices. The triangular mesh defines the front face of the blade. For simplicity, the thickness of blades is constant (0.1 units).

4.3 Indirect Representation R_3, R_4

Our indirect representations of blade's shape are based on ideas of both Ebner's VRML representation, and our direct representation R_2 using triangular mesh. We use reference grid with 26 vertices (their position in grid is modified compared to R_2 to prevent the final blade to stretch out of the bounding prism too often, figure 2). All vertices are transformed in space using a transformation tree which represents the whole genotypical information. The tree has a constant number of leaves 26. Leaf contains an index of a vertex from the reference grid (all leaves together cover all vertices). Inner nodes of the tree are either rotation or translation nodes. Branching factor of the tree is 1–3. The transformations on the path from the root of the tree to a leaf were applied in this order on the corresponding vertex of the reference grid. Our first approach to indirect representations – R_3 used fixed triangular mesh – the set of triangles covering vertices of the reference grid was fixed. This resulted in minor complications when vertices moved relatively too far from their original locations. The second indirect representation R_4 solves this problem by projecting already transformed vertices into XY plane. Delaunay triangulation provides set of triangles covering these projected vertices. Moreover, in R_4, the final positions of vertices are restricted so that their projections could not lay outside the boundary displayed in figure 2 right. Another difference between R_3 and R_4 is in the implementation of mutation and crossover operators. R_3 did not check the dimensions of final blade, therefore individuals that represented blades, whose dimensions did not meet our criteria, were also present in population, but during their evaluation in ODE simulator they received fitness zero.

R_4 genetic operators check the dimensions of projection of transformed vertices and also 3D dimensions of actual blade. In case the dimensions exceed limits, the mutation or crossover are tried again (maximum 5 times). If all attempts fail, the original individual (parent) is copied into the next generation instead. The consequence of this modification (R_3 vs. R_4) is the absence of zero-fitness individuals (blades that do not meet size restrictions) in population with R_4. This made the comparison between direct R_2 and indirect R_4 more appropriate, as R_2 adjusts elevation of vertices of reference grid so that they never exceed bounding prism (population with R_2 thus never has zero-fitness individuals).

5 Evolutionary Algorithm

The initial population of 200 individuals was created using the RHH method as in the work of Ebner. We used tournament selection of size 2, weak elitism, and ran for 200 generations. Common parameters were tuned to $p_{cross} = 0.3$, $p_{mut} = 0.7$.

In the case of R_1, the standard subtree crossover, and four different mutations were used: *BranchMutation* replaces a subtree with random content, *PointMutation* replaces a tree node with another tree node of compatible arity, *CollapseSubtreeMutation* replaces a random subtree with a new leaf, and *ExpansionMutation* replaces a leaf with a randomly generated subtree. The mutation probabilities were 0.5, 0.3, 0.1, and 0.1, respectively in the same order as above. In the case of R_2, we used the standard one-point crossover and simple average crossover operating on skewness angle. Since the points are topologically ordered, the crossover corresponds to geometric crossover, cutting the blade along a straight line. Because only some of the points of the grid are present in the individual phenotypes, this slight bias towards straight lines is partially compensated. We believe it is important to exchange the large parts of the blade using the crossover while the mutation with a higher rate tunes the correct details of the shape. Mutation operators were of four types: *AngleMutation* adds Gaussian noise to the skewness angle, *GaussAllMutation* alters the heights z_i using Gaussian $N(0, \delta)$, *Uniform3Mutation* alters three randomly selected z_i by a uniform noise from interval (-4,4), *PresenceMutation* toggles the presence of three randomly selected vertices. The skewness angle always remains in the interval $(0{,}90^0)$, and all heights z_i in the interval (-4,4). The probabilities of the mutations were 0.1, 0.5, 0.3, 0.1 in the order as above. The operators are implemented so that the resulting individual fitted into the bounding prism. In the cases of R_3 and R_4, we use a kind of swap crossover operator, while paying attention to the leaf nodes, which must form a permutation of the 26 vertices (at most three transformation nodes are swapped). Four mutation operators were used: *ContentGaussMutation*, and *ContentUniformMutation* apply Gaussian and uniform noise to all parameters in a single selected node, *SwapSubtreesMutation* swaps two random subtrees, and *SwapLeavesMutation* swaps two random leaves. The ratios used were 0.4, 0.2, 0.3, and 0.1, respectively. The experiments were implemented with the universal open-source evolutionary library EO. We only made small modifications of the engine in the way the user's call back function is called in order to allow for the distributed evaluation of the objective function.

6 Distributed Evaluation

To perform the experiments, we utilised the idle CPU time of computers in the student laboratories and of several powerful server machines. The evolutionary algorithm was running on a master node, which communicated with the slave nodes at an application layer using MySQL database. In each generation, the master program sent a set of individuals to be evaluated into a database table, and waited until the whole population became evaluated.

The nodes retrieved one individual each, ran a simulation and stored the computed fitness back to the database table. In order to increase the utilisation of the computational nodes, and because of their varying performance, some individuals could be evaluated simultaneously by several nodes. This occurs when all individuals have either been evaluated or are already assigned to some node and there are still some nodes available. This cures the situation when a difficult individual is assigned to a node with low performance, which would otherwise block the progress of the whole evolution for tens of minutes, leaving all remaining nodes idle. Close to the end of each generation, the extra available computational nodes spread over the whole set of the remaining individuals evenly, prioritising those that are already being tried for the longest time.

7 Results

After many testing runs focused on tuning the parameters and understanding the underlying mechanisms, we performed 11 evolutionary runs for each of the four representations. Each representation required about 1 year of total single-CPU time. Figure 5 plots the best and average fitness for all runs against the number of generations. The actual number of evaluations was slightly lower in the case of direct representation due to different sequence of mutation application, but the trend and the result is the same as in the plot against generations. All three triangular mesh representations (both direct and indirect) clearly outperform the original representation based on VRML and capped cylinders. The best result is achieved with R_3, the first version of the indirect representation (VRML) with triangular mesh (significantly better than direct representation, t-test with $t = 4.463$), and the trend is more promising. However, this is likely due to the capability of R_3 of accumulating mass by creating "folds", and due to the possibility of exploiting additional areas close to the rotation axis, which are not part of the search space of the direct representation. This option was removed from otherwise identical R_4, which performed worse than R_2, the direct representation with triangular mesh, even though still better than the original R_1.

There are couple of observations, which can explain lower performance of the original VRML representation with capsules. The initial population contained smaller trees, which in turn generated small blades accumulating little kinetic

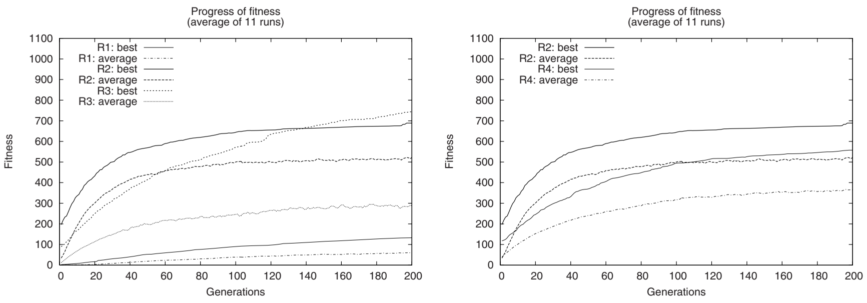


Fig. 5. Plot of the best fitness for R_1 - R_3 (left) and R_3, R_4 (right)

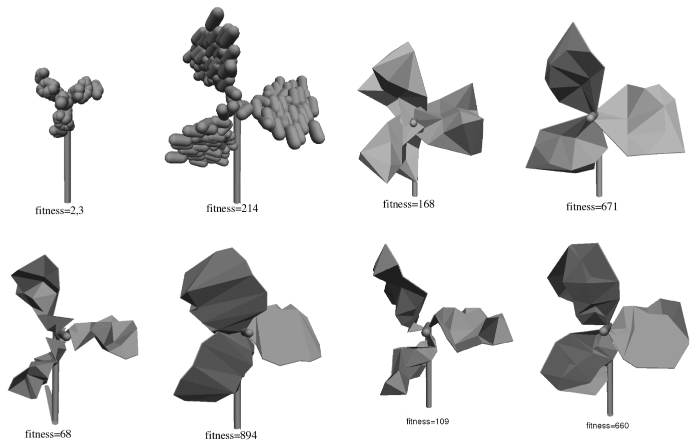


Fig. 6. Examples of evolved blades for the four representations R_1 – R_4, the best of the first and last generation

energy. This contributed to lower capacity of the evolution to explore the space with good solutions as it needed to spend more efforts on finding the proper region of the search space. In addition, an important aspect is the shape and size of the cutting edge of the blades. Using capsules implies thick blades with large friction against the air during the rotation. The figure 6 depicts the evolved shapes. Note their similarity to water turbines, rather than air turbines, which is due to our model that is not intended to be physically plausible. Note also how the evolution exploits the possibility to create folds to accumulate the weight.

Part of the genome in the direct representation was the angle of skewness of the reference plane. This variable converged to values around 10°, runs with indirect representation used a fixed 18° skewness.

8 Conclusions and Future Work

We follow up on the work of Marc Ebner in studying representations for evolutionary design of objects. We make a completely independent re-implementation and confirm his main results. We design two new representations, where the shape is defined as a set of triangles (triangular mesh). We demonstrate that the representation is more suitable for this problem and in general as it allows higher flexibility, better accuracy, and interesting genotype representations. We design both a direct and indirect (*generative*) representations, both of them performing significantly better than the original representation. We design a distributed system that utilises idle CPU time of computational nodes in student laboratories.

The future work will focus on studying new indirect representations that could outperform direct representations. A large number of possibilities exists, and remains only for a creative mind of the experimenter as we make the source-code publicly available [11], some more details of the experiments can be found in [9]. The same simulation framework can be evaluated with other problems of evolutionary design, and perhaps enabled with active controllers for the mutual co-evolution of robot morphologies and controllers.

References

1. Ebner, M.: Evolutionary Design of Objects Using Scene Graphs. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 47–58. Springer, Heidelberg (2003)
2. Go, J., Browning, B., Veloso, M.: Accurate and flexible simulation for dynamic,vision-centric robots. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) AAMAS 2004. LNCS (LNAI), vol. 3394, Springer, Heidelberg (2005)
3. Hamda, H., Jouve, F., Lutton, E., Schoenauer, M., Sebag, M.: Compact Unstructured Representations for Evolutionary Topological Optimum Design. Applied Intelligence 16, 139–155 (2002)
4. Hornby, G.S.: Generative Representations for Evolutionary Design Automation. PhD Thesis, Michtom School of Computer Science, Brandeis University, MA (2003)
5. Keijzer, M., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving Objects: a general purpose evolutionary computation library. Artificial Evolution, 231–244 (2001)
6. Laue, T., Spiess, K., RÃfer, T.: SimRobot – A General Physical Robot Simulator and Its Application in RoboCup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 173–183. Springer, Heidelberg (2006)
7. Michel, O.: Cyberbotics Ltd. Webots^{TM}: Professional Mobile Robot Simulation. Int. Journal of Advanced Robotic Systems 1(1), 39–42 (2004)
8. Robinson, G., El-Beltagy, M., Keane, A.: Optimisation in Mechanical Design. In: Bentley, P. (ed.) Evolutionary Design by Computers, Morgan Kaufmann, San Francisco (1999)
9. Plavcan, J., Petrovic, P.: Distributed Evolutionary Experiments with Representations for Evolutionary Design. IDI NTNU Technical Report (2007)
10. Smith, R.: Open Dynamics Engine Users Guide (2006)
11. RoboWiki: Evolving Shapes (2007), http://robotika.sk/projects/evodesign/
12. Zagal, J.C., del Solar, J.R.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 34–45. Springer, Heidelberg (2005)

Adaptive and Assortative Mating Scheme for Evolutionary Multi-Objective Algorithms

Khoi Le and Dario Landa-Silva

Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science, The University of Nottingham
{kxl,jds}@cs.nott.ac.uk

Abstract. We are interested in the role of restricted mating schemes in the context of evolutionary multi-objective algorithms. In this paper, we propose an adaptive assortative mating scheme that uses similarity in the decision space (genotypic assortative mating) and adapts the mating pressure as the search progresses. We show that this mechanism improves the performance of the simple evolutionary algorithm for multi-objective optimisation (SEAMO2) on the multiple knapsack problem.

1 Introduction

Selection plays an important role within evolutionary algorithms in selecting individuals for survival and selecting parents for recombination. Here, we are interested in *mating schemes*, i.e. the selection of parents for recombination within evolutionary multi-objective (EMO) algorithms. A number of mating schemes have been proposed in the literature including: fitness proportionate selection, tournament selection, rank-based selection, ancestry selection and assortative mating among others. In *fitness proportionate selection*, parents are chosen based on a probability proportional to their fitness compared to the rest of the population. In *tournament selection*, a group of individuals (usually two) is chosen (usually uniformly) from the population and the fittest individual from this group is selected as parent. In *rank-based selection*, individuals are first sorted according to some criteria (usually fitness) and a mapping function is used to assign a selection probability to each individual according to its rank in the ordering. In *ancestry selection* individuals are organised in clans and parents are usually selected from different clans. In *assortative mating* (inspired on natural genetics), individuals are selected based on their similarity (in the objective or the decision space) based on the assumption that recombining parents that 'look' alike produces better offspring. Some mating schemes incorporate some form of *restricted mating* (proposed by Goldberg [6]) where recombination is allowed only if parents meet certain criteria. For reviews on mating schemes and their performance of single-objective evolutionary algorithms see [1,7,8].

Despite the various restricted mating schemes that have been investigated for single-objective evolutionary algorithms, the emphasis within EMO algorithms has been mainly on mechanisms to select individuals for survival. In Pareto-based multi-objective optimisation the goal is to find a set of non-dominated solutions that is as close as possible to the Pareto optimal front and also well spread and distributed over the trade-off surface [2,4]. Therefore, most modern EMO algorithms incorporate selection mechanisms, like density-based selection and rank-based selection, in combination with elitism and archiving strategies to ensure the survival of good non-dominated solutions [2,4,14]. Also, most EMO algorithms use tournament or other basic selection mechanism for choosing parents and in most cases selection is based on fitness. Some restricted mating schemes have been investigated in the context of EMO algorithms but to a lesser extent than for single-objective evolutionary algorithms. In their book, Coello Coello et al. ([2], p. 201) express that restricted mating has not been fully investigated for EMO algorithms. They also note that there is no conclusive evidence to support whether restricted mating is beneficial or detrimental for the performance of these algorithms. Coello Coello et al. also suggest that experiments investigating the issue of restricted mating should benefit the literature on EMO algorithms. In this paper, we propose an *adaptive assortative mating scheme* [3] for evolutionary multi-objective optimisation. That is, parents are chosen based in their similarity in the decision space and the similarity threshold or mating pressure σ_{mating} is adapted during the search. In Section 2 we give a more detailed account of related work. In Section 3 we describe our proposed mating scheme and how this is incorporated into SEAMO2 (simple evolutionary algorithm for multi-objective optimisation) [15]. Section 3 also describes the experimental setting and our results. Final remarks are given in Section 4.

2 Mating Schemes for EMO Algorithms

We refer to the k-optimisation problem in which the aim is to optimise the function $f(x) = (f_1(x), f_2(x), ..., f_k(x))$ subject to $x \in X$, where x represents the decision vector, X represents the set of all feasible solutions, $f(x)$ represents the objective vector and each $f_i(x)$ represents the value of the *i-th* objective. Within a set $\mathbf{S}$ of solutions, solution x is said to be *non-dominated* if there is no solution in $\mathbf{S}$ that is better than x in each of the k objectives. Then, x is said to be Pareto-optimal if x is non-dominated with respect to the set X.

Now we briefly review restricted mating schemes that have been implemented into EMO algorithms. We focus on schemes proposed recently and that restrict mating based on similarity, i.e. assortative mating. For an overview of previous mating schemes within EMO algorithms refer to the book by Coello Coello et al. ([2], p. 201). Restricted mating has been usually implemented using the σ_{mating} parameter which defines a mating radius or similarity threshold, which can be perceived as the *mating pressure*. Individuals are not allowed to mate if the distance between them (objective or decision space) is larger than σ_{mating}. Kim et al. [13] incorporated neighbourhood crossover into SPEA2 to rank individuals according to how close they are in the objective space and used binary tournaments to select parents. Few years ago, Ishibuchi and Shibata started an investigation into the effect of restricted mating on the performance of the well-known NSGA2 and SPEA algorithms. In 2003 they proposed a restricted mating scheme based on the similarity between parents (assortative mating) [9]. Later, they modified their approach by incorporating a second layer to select parent A [10]. Their restricted mating scheme works as follows:

1. A set S_A of α candidates is chosen using iterative binary tournaments.
2. The center vector (average solution) $\bar{f}(x) = (\bar{f}_1(x), \bar{f}_2(x)...\bar{f}_k(x))$ in the objective space in S_A is calculated where $\bar{f}_i(x) = 1/\alpha \sum_{j=1}^{\alpha} f_i(x_j)$ for $i = 1, 2, ..., k$.
3. The solution in S_A that is most dissimilar (in the objective space) to $\bar{f}(x)$ is chosen as parent A.
4. A set S_B of β candidates is chosen using iterative binary tournaments.
5. The solution in S_B that is most similar to parent A (in the objective space) is chosen as parent B.

In [10] Ishibuchi and Shibata observed that their modified mechanism was capable of improving both convergence and diversity in SPEA and NSGA2. However, they also noted that the parameters α and β needed to be carefully adjusted to strike the balance between diversity and convergence speed. Note also that in [10] Ishibuchi and Shibata used similarity in the objective space only. In 2004, they reported further experiments to investigate the effect of the mating pressure parameters (α and β) and also the effect of similarity (in the objective space) when selecting parents A and B [11]. They tried their restricted mating mechanism in a number of operation modes resulting from combining different settings: $\alpha = \{1, 2, 3, \ldots\}$; $\beta = \{1, 2, 3, \ldots\}$; parent A being similar or dissimilar to $\bar{f}(x)$; parent B being similar or dissimilar to parent A. Once again, they observed that convergence speed and diversity were affected by the settings of α and β. They also expressed that there is a need to set α and β automatically in their mating scheme. More recently, Ishibuchi and Shibata reported yet more experiments in which they observed that recombining similar parents (which is controlled by varying β) had a positive impact on the performance of NSGA2, although they also observed that recombination seems to be less important than mutation on that particular algorithm [12]. In [12] they considered similarity in the objective and the decision space but only when selecting parent B and observed no significant difference in their results.

In summary, the investigations by Ishibuchi and Shibata have considered fitness-based binary tournaments and distance in the objective space to choose parent A. For selecting parent B, they employed fitness-based binary tournaments and distance both in the objective space and the decision space. The mating pressure is controlled by the number of tournaments (α and β) and by the target similarity to select parent A (with respect to the center vector $\bar{f}(x)$) and parent B (with respect to parent A). Their results have shown that although their mating scheme is able to improve the performance of SPEA and NSGA2, careful adjustment of the parameters is required to strike the balance between convergence and diversity according to the problem size.

3 The Adaptive Assortative Mating Scheme

3.1 The Experimental Setting

We propose an adaptive assortative mating scheme for selecting parents in EMO algorithms. The proposed mating scheme does not use tournaments, it uses similarity in the decision space and changes the mating pressure σ_{mating} as the search progresses. Therefore, this scheme differs from those proposed by Kim et al. [13] and Ishibuchi and Shibata [11]. In the proposed *dynamic assortative mating scheme* two individuals are considered for reproduction only if their dissimilarity (difference between their gene structures) is above a threshold σ_{mating}. For the experiments in this paper, we incorporate the proposed assortative mating scheme into the SEAMO2 algorithm [15] and carry out experiments on the multiple knapsack problem. We chose SEAMO2 because it is a simple evolutionary algorithm for multi-objective optimisation that relies mainly on its replacement strategy and it was shown to outperform more elaborate EMO algorithms like NSGA2 and SPEA2 on the multiple knapsack problem [15]. In this paper, with refer as SEAMO2(RM) to the SEAMO2 approach using the proposed scheme for restricted mating. Then, we focus our experiments in comparing the performance of SEAMO2(RM) against SEAMO2 [15], SPEA2 [16], NSGA2 [5] and SEAMO2(I) (the SEAMO2 algorithm using Ishibuchi and Shibata's mating strategy [10]) on the multiple knapsack problem. We use the instances with two, three and four knapsacks (with population size of 250, 300 and 350 respectively) and 750 items proposed in [17]. We carry out short, medium and long runs, 500, 960 and 1920 generations respectively, to investigate the performance of SEAMO2(RM). Results from 30 independent runs for each experiment are used for statistical analysis and discussion. We use two metrics, the *size of the space covered* $\mathcal{S}$ and the *coverage of two sets* $\mathcal{C}$ (see [17] for details on $\mathcal{S}$ and $\mathcal{C}$).

3.2 Similarity Measurement

In this paper, the dissimilarity or distance in the decision space between solutions to the multiple knapsack problem is measured as follows:

Individual $s = \{g_i | i : 0..(n-1)\}$
where n is the number of genes in the individual representation

$$g_i = \begin{cases} 0 & : \text{gene } i \text{ is not in the gene structure of individual } s \\ 1 & : \text{gene } i \text{ is in the gene structure of individual } s \end{cases}$$

the similarity and dissimilarity between individuals s_1 and s_2 are as follows:

$$simil(s_1, s_2) = \frac{|\{i | g_{1_i} = 1 \wedge g_{2_i} = 1\}|}{|\{i | g_{1_i} = 1 \vee g_{2_i} = 1\}|}$$

$$diff(s_1, s_2) = 1 - simil(s_1, s_2)$$

The recombination of s_1 and s_2 is allowed if and only if $diff(s_1, s_2) \geq \sigma_{mating}$ (where $0 \leq \sigma_{mating} \leq 1$). Setting the value of the mating pressure σ_{mating} is important and is discussed in the following sections.

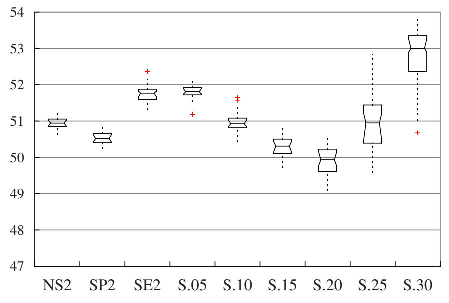
Note that the above definition of similarity is only valid for solutions to the multiple knapsack problem as encoded in this paper. If the similarity between two solutions of a problem is measured as a percentage, the proposed mating scheme can still be implemented as described later in this paper. Therefore, the generality of the proposed mating scheme is not affected by the encoding of solutions or the method used to measure similarity.

3.3 Static Setting of the Mating Pressure

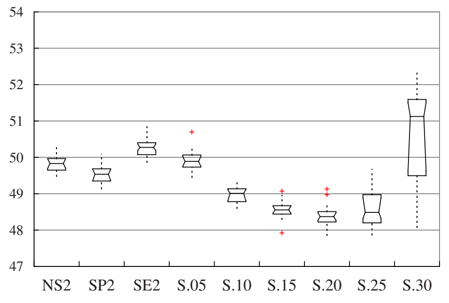
We first describe a simple strategy to preset σ_{mating} before starting the search and this value remains unchanged throughout the evolutionary process. We first calculate the value of $\mathit{diff}(s_m, s_n)$ for every pair of individuals s_m and s_n in the population. Then, we calculate the range using the minimum and maximum values, i.e. $\mathit{diff}(\mathit{range}) = (\max(\mathit{diff}(s_m, s_n)) - \min(\mathit{diff}(s_m, s_n)))$. We preset σ_{mating} to a value in this range. Otherwise, if σ_{mating} is set to a value smaller than $\min(\mathit{diff}(s_m, s_n))$ the selection of parents becomes uniform. Also, if σ_{mating} is set to a value greater to $\max(\mathit{diff}(s_m, s_n))$ no pair of individuals (s_m, s_n) would satisfy the selection condition for recombination.

In order to set σ_{mating} to an appropriate value within $\mathit{diff}(\mathit{range})$ we could let the population to evolve for a limited number of generations and observe the trend on the values of $\mathit{diff}(s_m, s_n)$ in the whole population. We carried out a simple experiment on the multiple knapsack problem and allowed the population to evolve for 100 generations recording $\mathit{diff}(\mathit{range})$ in every generation. We observed that $\mathit{diff}(\mathit{range})$ reduces significantly from 60%-70% in the first few generations to 0%-35% in later generations. Therefore, we set σ_{mating} to a value in the range of $(0.0, 0.3)$. Next, we carried out experiments using eleven different values of σ_{mating} : $0.050, 0.075, \ldots, 0.300$ in SEAMO2(RM). Results from 30 independent runs are reported in Figure 1. Note that we only show results for six values of σ_{mating} which are representative of all our experimental data. The box-plots in Figure 1 correspond to the percentage of *non-covered objective space*, i.e. smaller values indicate better algorithm performance. One box-plot is given for each algorithm: NSGA2, SPEA2, SEAMO2, and SEAMO2(RM) using different values of σ_{mating}.

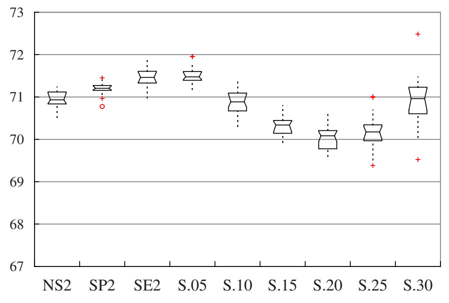
Figure 1 shows clearly that with respect to the *size of the space covered* $\mathcal{S}$ the proposed mating scheme has a positive effect on the performance of SEAMO2(RM). In general, we can see that the performance of SEAMO2(RM) using a preset value of σ_{mating} is consistent over the 30 independent runs (size of the boxplot). There is a significant improvement by applying a higher mating pressure (i.e. increasing the value of σ_{mating}). However, we can also observe that there is an upper limit for the mating pressure after which SEAMO2(RM) starts to perform worse. We can see in Figure 1 that this upper limit is about 25% for the 2-knapsack problem (Figure 1(a), 1(b)), between 25%-30% for the 3-knapsack problem (Figure 1(c), 1(d)), and slighly above 30% for the 4-knapsack problem (Figure 1(e), 1(f)). This is simply because when σ_{mating} goes above a given value, no parents can be found that satisfy the restricted mating condition. We omit full results for the $\mathcal{C}$ metric (all experimental results are available



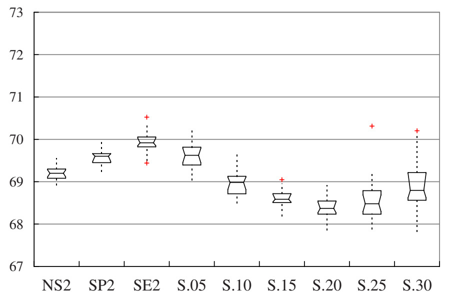
(a) ks2-500



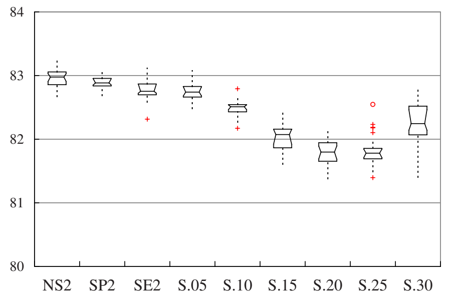
(b) ks2-1920



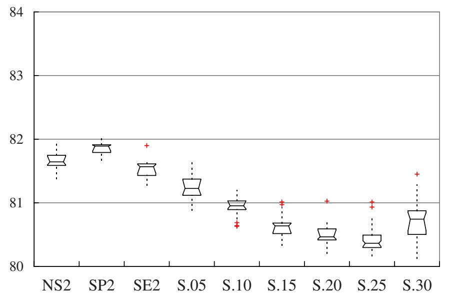
(c) ks3-500



(d) ks3-1920



(e) ks4-500



(f) ks4-1920

Fig. 1. Performance of algorithms on the multiple knapsack problem with respect to percentage of the *non-covered objective space*. NSGA2, SPEA2 and SEAMO2 are indicated by NS2, SP2 and SE2 respectively while S.xx indicates SEAMO2(RM) with a given value for σ_{mating}. Results are given for 2 (graphs a-b), 3 (graphs c-d) and 4 (graphs e-f) knapsacks with runs of 500 and 1920 generations.

on request) but we observed that increasing σ_{mating} seems to have a negative impact on convergence and a positive impact on diversity. To illustrate this, we show in Figure 2 the offline non-dominated fronts after 30 independent runs of SEAMO2(RM) on the 2-knapsack problem. For better visualisation, we show the non-dominated fronts in a lower density (only solutions separated by a distance

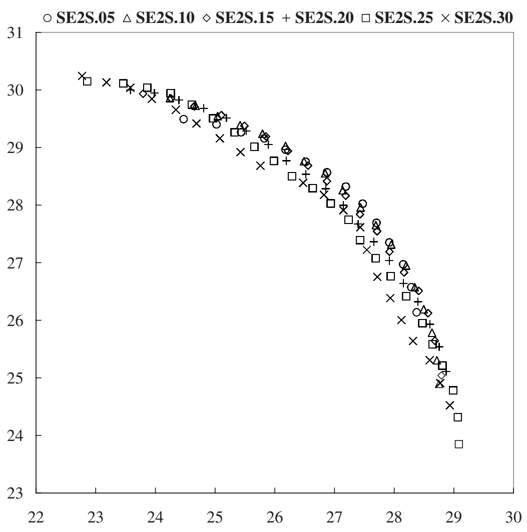


Fig. 2. Results of SEAMO2(RM) on the 2-knapsack and 750 items problem for six values of σ_{mating}. The horizontal axis represents profit in knapsack one and the vertical axis represents profit in knapsack two.

of at least 400 units in the objective space). We can see that higher σ_{mating} values reduce the convergence of SEAMO2(RM) but increase diversity (this is similar to the observations by Ishibuchi and Shibata [9]). Therefore, in the next subsection we propose to adapt the mating pressure as the search progresses.

3.4 Dynamic Setting of the Mating Pressure

Now we describe how σ_{mating} is adapted during the evolutionary search. This allows to improve both convergence and diversity of the population along with the evolutionary process. To dynamically change the value of σ_{mating}, we need first to establish the *diff*(*range*). As discussed in section 3.3, we select uniformly a value for σ_{mating} in every generation within the 5th and 95th percentile of *diff*(*range*). This prevents the restricted mating becoming uniform selection (if σ_{mating} is too low) or becoming a non-reproduction scheme (if σ_{mating} is too high). Note that the mating pressure σ_{mating} is set in an adaptive manner as *diff*(*range*) is adjusted after every generation to reflect the change of diversity (in the decision space) in the population. Then, the chosen value of σ_{mating} will adjust as the population diversity changes. For example, in the first few generations, the population is less 'stable' with many randomly generated solutions provoking a high value of σ_{mating}. However, once the population is more 'stable', changes in the value of σ_{mating} drive the population to evolve towards improving diversity (wider *diff*(*range*)) or improving convergence (smaller *diff*(*range*)).

As before, we carry out 30 independent runs of SEAMO2(RM) using the dynamic σ_{mating}. We also include the 'best results' obtained using Ishibuchi and

Shibata's restricted mating strategy [10] and using the static mating strategy of section 3.3. These 'best results' are based on the average of the $\mathcal{S}$ metric over 30 independent runs. We used 90 combinations of values $\alpha = \{1, 3, 4, \ldots, 9, 10\}$ and $\beta = \{1, 2, \ldots, 9, 10\}$ for Ishibuchi and Shibata's strategy and 11 different values of σ_{mating} in the static mating strategy. These 'best results' are indicated as SE2I and SE2S in Figure 3 while SE2D indicates SEAMO2(RM) using the dynamic σ_{mating}. Figure 3 compares NSGA2, SPEA2, SEAMO2, SEAMO2 with Ishibuchi and Shibata's mating strategy, SEAMO2 with the static σ_{mating} setting and SEAMO2 with the dynamic σ_{mating} setting, with respect to the $\mathcal{S}$ metric. Table 1 shows the comparison with respect to the $\mathcal{C}$ metric.

For each knapsack problem, Figure 3 shows the average *non-covered objective space* (smaller values indicate better algorithm performance) at generations 500, 960 and 1920 side by side to facilitate comparison. It is clear that the dynamic setting of σ_{mating} benefits SEAMO2 helping it to outperform NSGA2, SPEA2 and SEAMO2 as well as SEAMO2 with Ishibuchi and Shibata's mating strategy. Furthermore, both our static and dynamic mating strategies outperform Ishibuchi and Shibata's restricted mating strategy when incorporated into SEAMO2. In most cases, the dynamic strategy ourperforms the static one with the exception of the 2-knapsack problem with short and medium runs (graphs a-b in Figure 3). Table 1 shows the strong performance of SEAMO2D (the dynamic restricted mating incorporated in SEAMO2) particularly on problems with 3 and 4 knapsacks. From Figure 3 and Table 1 we can see that the dynamic mating strategy significantly improves diversity but it slightly worsens convergence in the higher dimension problem (4 knapsacks). We also notice an interesting result in that Ishibuchi and Shibata's strategy seems to worsen the performance of SEAMO2 (it was reported in [10] that Ishibuchi adn Shibata's strategy improves the performance of SPEA and NSGA2). This is more noticeable in the early stages of the evolutionary search (generations 500 and 960) in low dimension problems (2 and 3 knapsacks). We believe that Ishibuchi and Shibata's mating strategy conforms with the selection strategy in SPEA and NSGA2 where individuals are uniformly chosen using tournament selection. However, Ishibuchi and Shibata's mating strategy interferes with the selection strategy in SEAMO2 (Ishibuchi and Shibata's mating strategy chooses the first parent with binary tournaments while in SEAMO2 each individual acts as the first parent once). Figure 4 shows (in lower density as in Figure 2) the non-dominated fronts over 30 independent runs on the 2-knapsack problem. We can see that SEAMO2(RM) using the dynamic mating strategy outperforms SPEA2 and NSGA2 but its convergence is just slightly lower than for SEAMO2. Overall, results in Figure 3, Figure 4 and Table 1 give evidence that the dynamic setting of σ_{mating} is beneficial for SEAMO2 on the three multiple knapsack problems.

In Figure 5 we compare the proposed assortative mating scheme using the static setting and using the dynamic setting over 30 independent runs for the 2-knapsack problem with 750 items. The various static settings are indicated by SE2S.xx and the dynamic setting is indicated by SE2D. We can see that the *dynamic assortative mating scheme* can simultaneously maintain the convergence

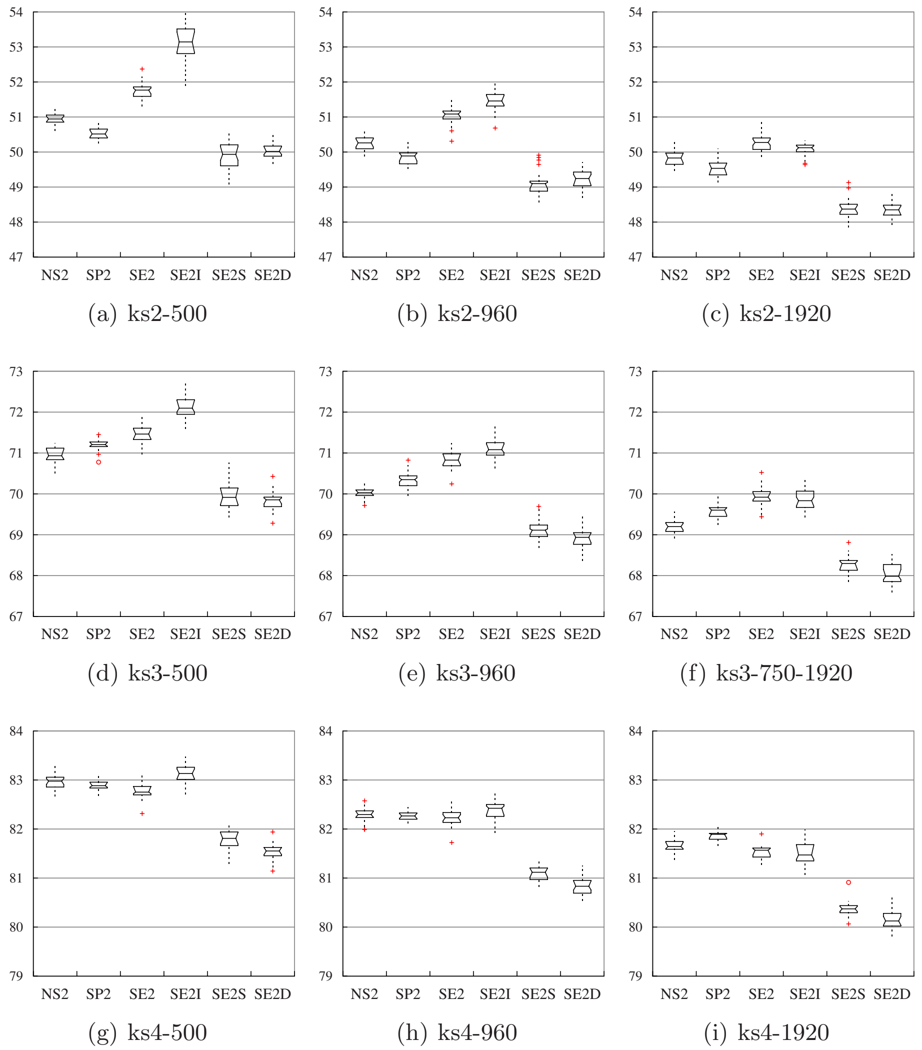


Fig. 3. Performance of algorithms on the multiple knapsack problem with respect to the percentage of the *non-covered objective space*. NSGA2, SPEA2 and SEAMO2 are indicated by NS2, SP2 and SE2 respectively. SE2I indicates SEAMO2 with Ishibuchi and Shibata's strategy, SE2S indicates SEAMO2 with the static setting and SE2D indicates SEAMO2 with the dynamic setting.

and the diversity of the population but the static setting can only give a positive effect on the convergence (using lower σ_{mating}) or on the diversity (using higher σ_{mating}) but not both at the same time. This shows that adapting the *diff*(*range*) (from where σ_{mating} is chosen) according to the population diversity during evolution, helps to strike a balance between convergence and diversity. Of course, more elaborate methods for adapting the mating pressure can be investigated, but the proposed one points us on the right direction.

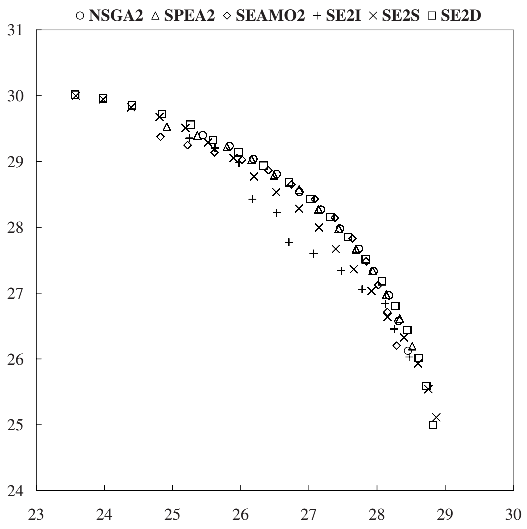


Fig. 4. Results comparing NSGA2, SPEA2, SEAMO2, SEAMO2 using Ishibuchi and Shibata's strategy (SE2I), SEAMO2 using the static mating strategy (SE2S) and SEAMO2(RM) using the dynamic mating strategy (SE2D) on the 2-knapsack problem with 750 items. The horizontal axis represents profit in knapsack one and the vertical axis represents profit in knapsack two.

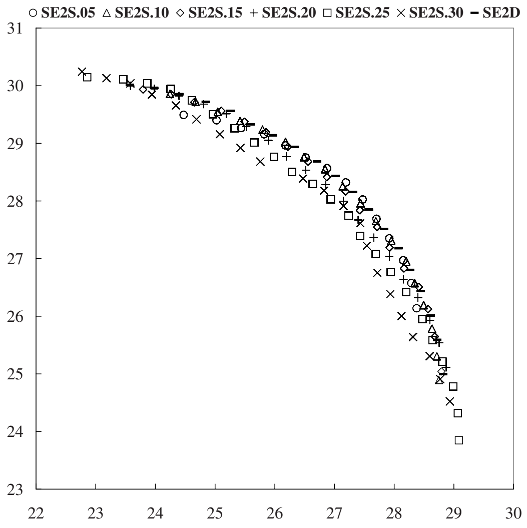


Fig. 5. Comparing the static and dynamic strategies for setting the mating pressure σ_{mating} in SEAMO2(RM). These results are for the 2-knapsack problem with 750 items. The horizontal axis represents profit in knapsack one and the vertical axis represents profit in knapsack two.

Table 1. Average values (standard deviation) of *coverage of two sets* $\mathcal{C}(A \succeq B)$

$\mathcal{C}(A \succeq B)$										
Algorithm		2 knapsacks			3 knapsacks			4 knapsacks		
A	B	500	960	1920	500	960	1920	500	960	1920
NSGA2	SEAMO2D	3(7)	12(16)	24(20)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
SPEA2		2(3)	8(7)	18(16)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
SEAMO2		11(17)	18(20)	26(19)	21(17)	24(16)	26(15)	26(22)	21(18)	19(14)
SEAMO2I		0(0)	0(0)	0(1)	0(0)	0(0)	2(5)	0(0)	0(1)	1(3)
SEAMO2S		2(3)	3(4)	5(5)	0(1)	1(1)	1(1)	0(1)	1(1)	1(1)
SEAMO2D	NSGA2	89(12)	69(24)	46(27)	92(7)	84(7)	77(8)	100(1)	99(3)	98(3)
	SPEA2	89(10)	74(15)	53(22)	88(8)	64(10)	45(9)	95(4)	84(7)	76(7)
	SEAMO2	76(34)	60(38)	47(34)	34(29)	24(25)	18(19)	16(20)	15(18)	12(12)
	SEAMO2I	100(0)	100(3)	92(15)	100(1)	98(2)	82(25)	91(16)	84(23)	65(32)
	SEAMO2S	80(8)	82(11)	83(10)	86(6)	83(6)	78(8)	79(6)	75(7)	67(7)

4 Final Remarks

This paper proposes a restricted mating scheme for evolutionary multi-objective (EMO) algorithms. This mating scheme is *assortative* because it selects parents based on their similarity in the decision space. Setting the mating pressure σ_{mating} to a constant value provokes either convergence or diversity to be negatively affected. Therefore, the proposed scheme is *adaptive* because it varies σ_{mating} taking into account the population diversity in the decision space. Our experiments show that the simple mechanism to adapt the mating pressure helps SEAMO2 (simple evolutionary algorithm for multi-objective optimisation) to improve its performance while striking a good balance between convergence and diversity. The proposed mating scheme can be incorporated into different EMO algorithms because it does not alter their original selection strategy. Future work contemplates comparison with other mating schemes proposed for EMO algorithms and on other problems such as nurse scheduling and job shop scheduling problems. We also intend to investigate other strategies to set the threshold σ_{mating} to further improve diversity and convergence of EMO algorithms.

References

1. Blickle, T., Thiele, L.: A Comparison of Selection Schemes Used in Evolutionary Algorithms. Evolutionary Computation 4(4), 361–394 (1997)
2. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
3. De, S., Pal, S.K., Ghosh, A.: Genotypic and Phenotypic Assortative Mating in Genetic Algorithm. Information Sciences 105, 209–226 (1998)
4. Deb.: Kalyanmoy: Multi-objective Optimization Using Evolutionary Algorithms, Wiley, Chichester (2001)

5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimisation and Machine Learning. Addison Wesley, Reading (1989)
7. Goldberg, D.E., Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In: Rawlins, G.J.E. (ed.) Foundations of Genetic Algorithms, Morgan Kaufmann, San Francisco (1991)
8. Huang, C.F.: An Analysis of Mate Selection in Genetic Algorithms. Technical Report, Center for the Study of Complex Systems, University of Michgan (2001), (also as poster paper in the GECCO 2001 conference p. 766 (2001)
9. Ishibuchi, H., Shibata, Y.: An Empirical Study on the Effect of Mating Restriction on the Search Ability of EMO Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 433–447. Springer, Heidelberg (2003)
10. Ishibuchi, H., Shibata, Y.: A Similarity-Based Mating Scheme for Evolutionary Multiobjective Optimization. In: The 2003 Genetic and Evolutionary Computation Conference, pp. 1065–1076 (2003)
11. Ishibuchi, H., Shibata, Y.: Mating Scheme for Controlling the Diversity-convergence Balance for Multi-objective Optimization. In: Deb, K., al., e. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1259–1271. Springer, Heidelberg (2004)
12. Ishibuchi, H., Shibata, Y.: Recombination of Similar Parents in EMO Algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 265–279. Springer, Heidelberg (2005)
13. Kim, M., Hiroyasu, T., Miki, M., Watanabe, S.: SPEA2+: Improving The Performance of The Strength Pareto Evolutionary Algorithm 2. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 742–751. Springer, Heidelberg (2004)
14. Laumanns, M., Zitzler, E., Thiele, L.: On The Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multiobjective Optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 181–196, Springer, Heidelberg (2001)
15. Mumford, C.L.: Simple Population Replacement Strategies for a Steady-State Multi-objective Evolutionary Algorithm. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1389–1400. Springer, Heidelberg (2004)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems, pp. 95–100 (2002)
17. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)

The Cooperative Royal Road: Avoiding *Hitchhiking*

Gabriela Ochoa[1], Evelyne Lutton[2], and Edmund Burke[1]

[1] Automated Scheduling, Optimisation and Planning Group, School of Computer Science & IT, University of Nottingham, Nottingham NG8 1BB, UK
[2] COMPLEX Team, INRIA Rocquencourt
Domaine de Voluceau BP 105, 78153, Le Chesnay Cedex, France

Abstract. We propose using the so called *Royal Road* functions as test functions for cooperative co-evolutionary algorithms (CCEAs). The Royal Road functions were created in the early 90's with the aim of demonstrating the superiority of genetic algorithms over local search methods. Unexpectedly, the opposite was found to be true. The research deepened our understanding of the phenomenon of *hitchhiking* where unfavorable alleles may become established in the population following an early association with an instance of a highly fit schema. Here, we take advantage of the modular and hierarchical structure of the Royal Road functions to adapt them to a co-evolutionary setting. Using a multiple population approach, we show that a CCEA easily outperforms a standard genetic algorithm on the Royal Road functions, by naturally overcoming the hitchhiking effect. Moreover, we found that the optimal number of sub-populations for the CCEA is not the same as the number of components that the function can be linearly separated into, and propose an explanation for this behavior. We argue that this class of functions may serve in foundational studies of cooperative co-evolution.

1 Introduction

Co-evolutionary Algorithms (CEAs) represent a natural extension to standard evolutionary algorithms for tackling complex problems; they can be generally defined as a class of evolutionary methods in which the fitness of an individual depends on its relationship to other members of the population. Several co-evolutionary approaches have been proposed in the literature; they vary widely, but the most fundamental classification relies on the distinction between cooperation and competition. In cooperative algorithms, individuals are rewarded when they work well with other individuals, and punished otherwise. Whereas in competitive algorithms, individuals are rewarded at the expense of those with which they interact. Most of the work on CEAs has been in competitive models; there has been, however, an increased interest in cooperation to tackle difficult optimization problems by means of problem decomposition [7,3,14,18,11,10]. The behavior of CEAs is very complicated and often counter-intuitive. Moreover, our knowledge about the dynamics and ways of improving standard evolutionary algorithms, is not directly transferable to co-evolution [15]. Thus, there is a need to conduct foundational research on co-evolutionary systems in order to improve their applicability as a problem solving methodology. With this in mind, we propose using the so called Royal Road functions [13,4] as test functions in cooperative co-evolution. The Royal Road functions were proposed with the aim of isolating some of the features of fitness landscapes thought to be most relevant to the performance of genetic algorithms. Surprisingly, it was found that a random-mutation hill climber significantly outperformed the genetic algorithm on these functions. However, this work led to a greater understanding of the phenomenon of hitchhiking in evolutionary search, whereby some deleterious alleles may become fixed in the population, after an early association with a highly fit schema.

Cooperative co-evolutionary approaches are generally applied to decomposable problems. Thus, we take advantage of the modular and hierarchical structure of the Royal Road test functions to adapt them to the co-evolutionary setting. We argue that these functions may serve theoretical studies of cooperative co-evolution, since the landscape can be varied in a number of ways, and the global optimum and all possible fitness values are known in advance. It would also be possible to study the dynamics of the search process by tracing the origins and history of individual building blocks. Moreover, these functions may be decomposed in several ways (including one or more blocks on each sub-component), which made them useful in studies testing the automated emergence of co-adapted components [18]. This study also makes a comparison between a standard and a cooperative evolutionary algorithm on several instances of the Royal Road functions. The cooperative algorithm explores all the alternative problem decompositions possible with the modular Royal Road functions. Our results show a clear advantage of the cooperative algorithm in this scenario, and we go further to analyze why this is the case. This analysis leads us to revisit the hitchhiking effect and the *building blocks* hypothesis in genetic algorithms.

Next section gives a brief overview on cooperative co-evolution, distinguishing between single and multiple population approaches, and describing some test problems used so far when studying cooperative co-evolution. Thereafter, section 3, introduces the Royal Road functions and describes the hitchhiking phenomenon. Section 4, describes the algorithms and parameter settings used, whilst section 5 presents and analyses our results. Finally, section 6 summarises our findings and suggests directions for future work.

2 Cooperative Co-evolution

Previous work on extending evolutionary algorithms to allow cooperative co-evolution can be divided into approaches that have a single population of interbreeding individuals, and those that maintain multiple interacting populations.

Single population approaches: The earliest single-population method that extended the basic evolutionary model to allow the emergence of co-adapted subcomponents was the classifier system [6]; which is a rule-based learning paradigm that evolves fixed length stimulus-response rules. An interesting generalization of this paradigm for solving complex problems was proposed in [1], where an aggregation of multiple individuals (in a single population) is considered for solving the inverse problem for Iterated Function Systems. In this approach, which has been termed *Parisian Evolution*, an additional fitness measure (a "local" fitness) is used to independently evaluate the subcomponents during the search process, while a "global" fitness is used at each generation to measure the progress of the aggregate solution. This scheme is well suited for incorporating additional or incomplete information about the searched solution. However, in order to avoid trivial and degenerate solutions, a special mechanism for maintaining the population diversity should be devised. Successful applications of the Parisian Approach can be found in the image analysis and signal processing literature [11,2]; and in data retrieval applications [10].

Multiple population approaches: The first authors to apply a multi-species cooperative co-evolutionary approach to tackle a difficult optimization problem were Husbands and Mill [8,7] who successfully co-evolved job-shop schedules, using a parallel distributed algorithm. A few years later, the work by Potter and De Jong [16] helped to popularise the idea of cooperative co-evolution as an optimisation tool. The authors devised a multiple population framework where a decomposition of the problem into subcomponents should be identified. Each component is, then, assigned to a subpopulation that evolves simultaneously but in isolation to the other sub-populations. The fitness of an individual in a given sub-population is calculated after selecting collaborators from the other sub-populations in order to form a complete solution. Notice that diversity in the ecosystem is, in this framework, naturally achieved through maintaining genetically isolated populations. This framework has been further analysed [22] by considering a relationship between cooperative co-evolution and evolutionary game theory, and thus studying it from a dynamical systems perspective. From the problem-solving point of view, multi-species cooperative co-evolution has been applied to neural networks and concept learning [14,17,18]; and to inventory control optimisation [3].

2.1 Abstract Test Functions

Most foundational empirical studies of cooperative co-evolution have used non linear function optimization problems as benchmark [17,23]. These problems are well suited for cooperative co-evolution, since a natural decomposition is straightforward: each subpopulation represents a particular variable of the function. In [14], much simpler functions (oneRidge and twoRidges) are studied. In his dissertation [15], Potter used several test functions including, a simple binary string covering task, continuous nonlinear functions, and Kauffman's coupled NK landscapes [9]. In a further, more theoretically oriented PhD dissertation, Wiegand [21] used cooperative versions of test functions such as the OneMax, LeadingOnes, and Trap functions.

3 Royal Road Functions and the Hitchhiking Effect

The building-block hypothesis [5] states that the genetic algorithm works well when short, low-order, highly-fit schemata (building blocks) recombine to form even more highly-fit, higher-order schemata. Thus, the genetic algorithm's search power has been attributed mainly to this ability to produce increasingly fitter partial solutions by combining hierarchies of building-blocks. Despite recent criticism, empirical evidence, and theoretical arguments against the building-blocks hypothesis [19], the study of schemata has been fundamental in our understanding of genetic algorithms. The first empirical counter-evidence against the building-block hypothesis was produced by Holland himself, in collaboration with Mitchell and Forrest [13,4]. They created the Royal Road functions, which highlight one feature of landscapes: hierarchy of schemata, in order to demonstrate the superiority of genetic algorithms (and hence the usefulness of recombination) over local search methods such as hill-climbing. Unexpectedly, their results demonstrated the opposite: a commonly used hill-climbing scheme (*random-mutation hill-climbing*) significantly outperformed the genetic algorithm on these functions. With this hill-climbing approach, a string is randomly generated and its fitness is evaluated. The string is then mutated (by a bit-flip) at a randomly chosen position, and the new fitness is evaluated. If the new string has an equal or higher fitness, it replaces the old string. This procedure is iterated until the optimum has been found or a maximum number of evaluations is reached. It is ideal for the Royal Road functions, since it traverses its "plateaus" and reaches the successive fitness levels. However, the algorithm (as with any other hill-climber) will have problems with any function with many local minima. The authors [13,4] also found that although crossover contributes to genetic algorithm performance on the Royal Road functions, there was a detrimental role of "stepping stones" - fit intermediate-order schemata obtained by recombining fit low-order schemata. The explanation suggested for these unexpected results lies in the phenomenon of hitchhiking (or spurious correlation), which they describe as follows [12]: "once an instance of a higher-order schema is discovered, its high fitness allows the schema to spread quickly in the population, with 0s in other positions in the string hitchhiking along with the 1s in the schema's defined positions. This slows down the discovery of the schema's defined positions. Hitchhiking can, in general, be a serious bottleneck for the GA."

3.1 Functions $R1$ and $R2$

To construct a Royal Road function [4], an optimum string is selected and broken up into a number of small building blocks. Then, values are assigned to each low-order schema and each possible intermediate combination of low-order schemata. Those values are, thereafter, used to compute the fitness of a bit string x in terms of the schemata of which it is an instance.

The function $R1$ (Figure 1) is computed as follows: a bit string x gets 8 points added to its fitness for each of the given order-8 schemata ($s_i, i = 1, 2, \ldots, 8$) of which it is an instance. For example, if x contains exactly two of the order-8 building blocks, then $R1(x) = 16$. Similarly, $R1(111\ldots1) = 64$. More generally,

s_1 = 11111111**; $c_1 = 8$
s_2 = ********11111111**; $c_2 = 8$
s_3 = ****************11111111**; $c_3 = 8$
s_4 = ************************11111111********************************; $c_4 = 8$
s_5 = ********************************11111111************************; $c_5 = 8$
s_6 = **11111111****************; $c_6 = 8$
s_7 = **11111111********; $c_7 = 8$
s_8 = **11111111; $c_8 = 8$
s_{opt} =1111111111111111111111111111111111111111111111111111111111111111

Fig. 1. The Royal Road function $R1$: An optimal string is broken into 8 building-blocks

s_9 = 1111111111111111**; $c_9 = 16$
s_{10} =****************1111111111111111********************************; $c_{10} = 16$
s_{11} =********************************1111111111111111****************; $c_{11} = 16$
s_{12} =**1111111111111111; $c_{12} = 16$
s_{13} =11111111111111111111111111111111********************************; $c_{13} = 32$
s_{14} =********************************11111111111111111111111111111111; $c_{14} = 32$
s_{opt}=1111111111111111111111111111111111111111111111111111111111111111

Fig. 2. The Royal Road Function $R2$: Some intermediate schemata are added to the those in $R1$. Namely, $s_9 \dots s_{14}$

$R1(x)$ is the sum of the coefficients c_s corresponding to each given schema of which x is an instance. Here c_s is equal to $order(s)$. The fitness contribution from an intermediate stepping stone (such as the combination of s_1 and s_3 in Figure 1) is thus a linear combination of the fitness contribution of the lower-level components.

In $R2$, the fitness contribution of some intermediate stepping stones is much higher (Figure 2). The Fitness in $R2$ is calculated as in $R1$: the sum of the coefficients corresponding to each schema (s_1 - s_{14}) of which a string is an instance. For example, $R2(1111111100011111111) = 16$, since the string is an instance of both s_1 and s_8, but $R2(111111111111111100\dots0) = 32$, because the string is an instance of s_1, s_2, and s_9. Thus, a string's fitness depends not only on the number of 8-bit schemata to which it belongs, but also on their positions in the string. The optimum string $11111111\dots1$ has fitness 192, because the string is an instance of each schema in the list.

In [13], the authors expected the genetic algorithm to perform better (i.e. find the optimum more quickly) on $R2$ than on $R1$, because in $R2$ there is a clear path via crossover from pairs of the initial order-8 schemata (s_1 - s_8) to the four order-16 schemata (s_9 - s_{12}), and from there to the two order-32 schemata (s_{13} and s_{14}), and finally to the optimum (s_{opt}). They believed that the presence of this stronger path would speed up the genetic algorithm's discovery of the optimum, but their experiments showed the opposite: the genetic algorithm performed significantly better on $R1$ than on $R2$.

4 Methods

As the cooperative co-evolutionary algorithm, we used the multiple populations approach (see Figure 3) firstly proposed by Potter and De Jong [16]; and later studied by other authors [22,15].

```
gen = 0
for each species s do
    Pop_s(gen) = initialized population
    evaluate(Pop_s(gen))
while not terminated do
    gen++
    for each species  s do
        Pop_s(gen) <- select(Pop_s(gen - 1))
        recombine(Pop_s(gen))
        evaluate(Pop_s(gen))
        survive(Pop_s(gen))
```

Fig. 3. The structure of a cooperative co-evolutionary algorithm

Table 1. Optimal population sizes (in the set of: 64, 128, 256, and 512), for both the standard genetic algorithm (SGA) and cooperative co-evolutionary algorithm (CCEA). For CCEA the size producing the best performance over all the number of species tested, was considered. L stands for the Royal Road function's string length.

	$R1$			$R2$		
	$L = 64$	$L = 128$	$L = 256$	$L = 64$	$L = 128$	$L = 256$
SGA	64	64	64	64	64	64
$CCEA$	128	256	256	64	128	256

In order to adapt the Royal Road functions to the co-evolutionary setting, a solution string is broken into equally sized sub-strings that contain one (or more) of the original function lower-order schemata. Each of these sub-strings represents a problem subcomponent, and is thus assigned to a separate population. A global solution is assembled by concatenating these sub-components. We used the simplest method of evaluating an individual in a given population; which is to couple it with the current best members of the remaining populations, apply the resulting string to the global function, and assign the resulting value as the fitness of the subcomponent. The initial fitness of each subpopulation member is computed by combining it with a random individual from each of the other species. We evaluated the cooperative co-evolutionary algorithm by comparing its performance with that of a standard genetic algorithm on several Royal Road functions (both $R1$ and $R2$). In order to maintain resemblance with the originally proposed Royal Road functions [13,4], we used functions (both $R1$ and $R2$) with lower-order schemata (building blocks - BBs) of length 8. With regard to the string length, we considered functions of L = 64, 128, and 256, that is, functions containing 8, 16 and 32 of these BBs. Several numbers of sub-populations (or species, SP) were considered starting from the minimum of two species, and doubling this number up to the maximum given by the number of BBs in the function. This corresponds to sub-populations having, respectively, a string length equal to half of the total Royal Road function length (half the number of BBs), down to sup-populations having a string length of 8 (i.e. a single BB). To set the size of each sub-population, we select a fixed number of individuals in the whole ecosystem, and thereafter distributed them equally among the sub-populations. For setting this ecosystem population size, we tested a range of values (64, 128, 256, and 512) and selected, for each string length and function, the size producing the best performance (see Table 1). We also selected the optimal population size, in this range, for the standard genetic algorithm, which turned out to be the smallest size explored (64) for all the functions. The remaining algorithm components were equal for the standard and cooperative genetic algorithm, and were held constant over the experiments. Specifically, we used binary tournament selection, 2-point crossover (rate = 0.8) and bit-flip mutation (rate $= 1/L$, $L=$ chromosome length), and 50 replicas for experiments. Further methodological details and the performance measures, are described in the following section.

5 Empirical Results and Analysis

For comparing the performances, the algorithms were allowed to continue until the optimum string was discovered, and the number of evaluations of this discovery was recorded. Table 2 shows the average number of evaluations ($\times 10^2$) to discover the optimum string, for the $R1$ (top) and $R2$ (bottom) functions. The standard deviations are shown within brackets. Note that 50 replicas for each experiment were carried out. From Table 2, we see that the CCEA (with any number of species) clearly outperformed the SGA on all the instances studied. On average, the CCEA (with the appropriate number of species) found the optimum a factor of about two times faster on $R1$, and three times faster on $R2$. Notice that, as has been reported before, the SGA performs better on $R1$ than on $R2$. This is not the case, however, for the CCEA where the algorithm has a similar best performance on both $R1$ and $R2$. Our explanation for the improved performance of the CCEA lies in the phenomenon of hitchhiking, described in section 3. By maintaining separate populations, the CCEA is able to avoid hitchhiking, since each sub-population samples independently each schema region. Thus, more than one desirable schema may appear simultaneously in the ecosystem, and thereafter these sub-components are aggregated when calculating the global function. In [12], the authors identify some features that would improve the performance of GAs as compared to other heuristic search algorithms. These are: (i) *independent samples*, (ii) *sequestering desired schemas*, and (iii) *instantaneous crossover* of desired schemas. It is clear that a cooperative genetic algorithm contains all these features, and entails a better implementation of the building-blocks hypothesis (see section 3).

Another interesting observation (Table 2 and Figure 4) is that the number of species (SP) on the CCEA, that produced the best performance was consistently (with the exception of $R1$ with 8 BBs - $L = 64$) achieved by $SP =$ half of the number of blocks in the function. This corresponds to a sub-population string length of 16 bits, namely two 8-bits BBs. This can be more clearly appreciated in Figure 4, which compares the algorithm's performance (SGA and CCEA with different a number of sub-populations) on both $R1$ and $R2$, with 16 and 32 blocks ($L = 128$ and 256). Thus, the optimal number of sub-populations for the CCEA (i.e. the number of problem sub-components) is not the same as the number of

Table 2. Average number of evaluations ($\times 10^2$) and standard deviations to find the optimum on the $R1$ (top) and $R2$ (bottom) functions. The sub-index in the CCEA corresponds to the number of species.

$R1$			
Algorithm	$L = 64$ (8 BBs)	$L = 128$ (16 BBs)	$L = 256$ (32 BBs)
SGA	227.8 (90.23)	665.1 (225.99)	2150.9 (805.70)
$CCEA_2$	165.4 (66.22)	571.4 (270.62)	2161.2 (768.20)
$CCEA_4$	142.2 (71.56)	402.2 (143.18)	1473.9 (577.42)
$CCEA_8$	**114.1**(43.30)	**327.1** (130.71)	1094.8 (464.03)
$CCEA_{16}$		365.2 (128.84)	**851.5** (319.73)
$CCEA_{32}$			1140.9 (352.26)

$R2$			
Algorithm	$L = 64$ (8 BBs)	$L = 128$ (16 BBs)	$L = 256$ (32 BBs)
SGA	241.3 (119.69)	947.3 (622.71)	3278.6 (1560.43)
$CCEA_2$	173.5 (74.76)	640.8 (301.91)	2480.1 (1141.24)
$CCEA_4$	**115.6** (51.22)	412.2 (214.98)	1432.1 (472.53)
$CCEA_8$	127.0 (56.36)	**305.7** (92.60)	1094.5 (430.02)
$CCEA_{16}$		399.9 (142.02)	**876.2** (330.46)
$CCEA_{32}$			1389.2 (365.93)

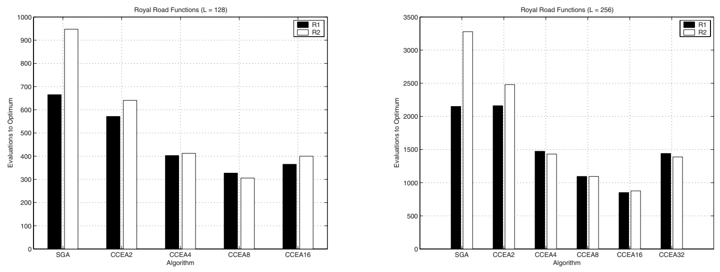


Fig. 4. Comparing the algorithm's performance on both $R1$ and $R2$ with $L = 128$ (left plot), and $L = 256$ (right plot). The bars measure the average number of evaluations ($\times 10^2$) to find the optimum.

pieces that the function can be linearly separated into, which, in principle, may appear to be a counter-intuitive observation. The following set of experiments, offers an explanation for this behavior.

5.1 Dynamic Behavior

In order to find an explanation for the observed optimal number of sub-populations in the CCEA (SP = half of the number of blocks in the function), we study, in this section, the algorithms's dynamic behavior. For the analysis we selected the $R1$ function with $L = 128$ (16 BBs), and a CCEA with 8 and 16 sub-populations, which were the SP values producing the best performance

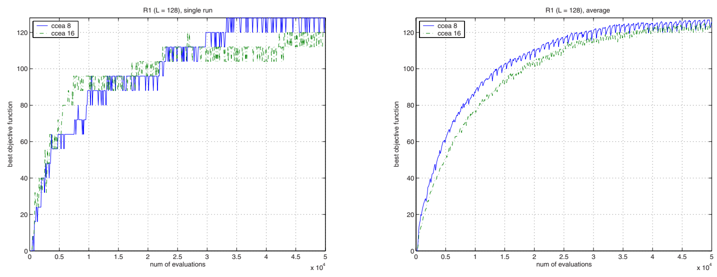


Fig. 5. Comparing the dynamic behavior of CCEA with 8 and 16 sub-populations on the $R1$ function with $L = 128$. The left-hand plot illustrates a single run, whereas the right-hand plot, averages 50 runs.

in this scenario. Figure 5 illustrates the performance curves for both a single run (left-hand plot) and averaging 50 runs (right-hand plot). Each point in the curves represents the global objective function value of the aggregated solution. Each run lasted 50 x 10^4 function evaluations, and the global objective value was sampled every 100 evaluations.

Notice that, on average, the CCEA with 8 species outperformed the CCEA with 16 species throughout the whole run. The single run dynamics looks more complicated, with both algorithms having similar performance at the initial stages of the search, and $CCEA_{16}$ dominating at some intermediate stages. Towards the final stages of the search, however, it is $CCEA_8$ which is producing and maintaining higher fitness values. Notice that in the process of the run, the fitness levels are discovered and lost several times before getting established, which suggests that the convergence behavior of the multi-population CCEA is slower and more complex than that of a standard genetic algorithm. The curves in Figure 5 show the behavior of the aggregate solution, hiding the information about the fitness contribution of each sub-population. In order to have a closer look at the fitness contribution of each schema or BB to the global objective function, Figure 6 illustrates the contribution of an example schema, where without loss of generality we select schema 9 (s_9). Both single run (top plot) and average (bottom plot) behavior are illustrated for the CCEA with 8 and 16 species[1].

From the single run plots of the fitness contribution of schema 9 (figure 6, top plots), it can be seen that the CCEA with 16 species is more unstable at maintaining the BBs of 8 consecutive ones (fitness contribution of 8), in other words the BB appears and disappears more easily throughout the search. This is because when $SP = 16$, an individual in each sub-population is composed of a single BB, so any bit mutation would break it. This is not the case with $SP = 8$, where each individual in a sub-population is composed of two consecutive BBs. In this scenario, a bit mutation may destroy one of the BBs, but keep

[1] The plots for all the other schemata were qualitatively similar, and are not shown due to space limits.

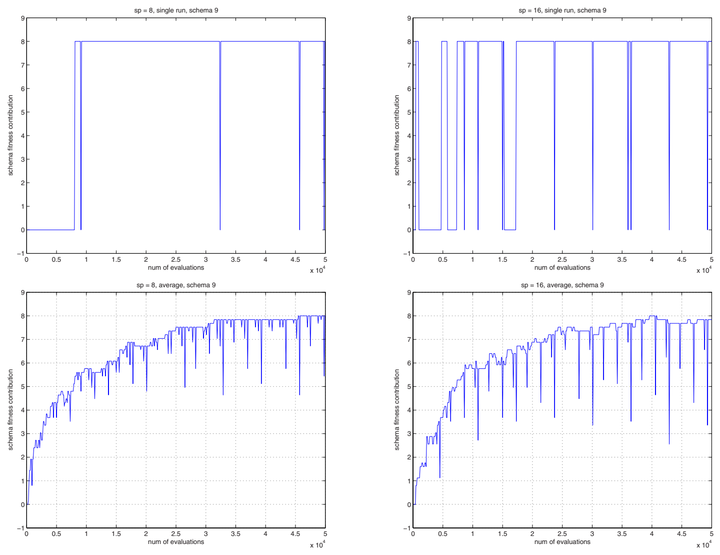


Fig. 6. Comparing the dynamic fitness contribution of a representative schema (s_9) in a CCEA with 8 (left-hand plots) and 16 (right-hand plots) species. The top plots illustrate single runs, whereas the bottom plots show the averages of 50 runs.

the other, and the correct concatenation of the two BBs can be easily recovered by a recombination event. This behavior is also reflected by the average plots (Figure 6, bottom plots) where the fitness contribution curve of schema 9 reaches lower levels (deeper drops) when $SP = 16$ (right-hand bottom plot). These plots therefore suggest that having sub-populations composed by individuals containing two BBs instead of a single BB, would benefit the stability and permanence of the appropriately set BBs in the sub-populations, thus supporting the overall better global behavior.

6 Conclusions

Cooperative co-evolution is suitable to decomposable problems; consequently, we have taken advantage of the modular and hierarchical structure of the Royal Road functions to adapt them as test functions for cooperative co-evolutionary algorithms (CCEAs). Our empirical results show that a CCEA clearly outperforms a standard genetic algorithm on the Royal Road functions, confirming our intuition that cooperative co-evolution helps in overcoming the so-called hitchhiking (or spurious correlation) effect, which is known to hinder the performance of evolutionary algorithms. This suggests that CCEAs may be an advantageous technique, even for static optimization, as they entail a better instantiation of the building-blocks hypothesis.

An advantage of the Royal Road functions as test functions in cooperative co-evolution is that they admit several natural decompositions, which makes them useful in studies to test the automated emergence of co-adapted components. Our results show that having two basic sub-components, instead of a single sub-component for sub-populations produced better overall performance, which suggests that caution should be taken when manually proposing a problem decomposition. A potential drawback of the Royal Road functions as test beds for cooperative co-evolution, is similar to that highlighted for standard evolutionary search; namely the independence (or separation) between the building blocks. To overcome this limitation, Watson et al. [20] have proposed the so called Hierarchical If-and-only-if (H-IFF) functions that have a hierarchical decomposable structure where sub-problems are not separable. In consequence, a natural extension of our contribution will be to propose a cooperative version of the H-IFF family of functions. Another interesting extension would be to compare and assess the scenarios where a single-population implementation of cooperative co-evolution (such as the Parisian approach [1]) would be advantageous over a multiple-population one.

References

1. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar IFS+parisian genetic programming=efficient IFS inverse problem solving. Genetic Programming and Evolvable Machines 1(4), 339–361 (2000)
2. Dunn, E., Olague, G., Lutton, E.: Parisian camera placement for vision metrology. Pattern Recognition Letters 27(11), 1209–1219 (2006)
3. Eriksson, R., Olsson, B.: Cooperative coevolution in inventory control optimisation. In: Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms, Springer, Heidelberg (1997)
4. Forrest, S., Mitchell, M.: Relative building-block fitness and the building block hypothesis. In: Foundations of Genetic Algorithms, vol. 2, pp. 109–126. Morgan Kaufmann, San Mateo (1993)
5. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI (1975)
6. Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. In: Pattern-directed inference systems. Academic Press, New York (1978)
7. Husbands, P.: Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: Fogarty, T.C. (ed.) AISB-WS 1994. LNCS, vol. 865, pp. 150–166. Springer, Heidelberg (1994)
8. Husbands, P., Mill, F.: Simulated co-evolution as the mechanism for emergent planning and scheduling. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 264–270. Morgan Kaufman, San Francisco (1991)
9. Kauffman, S.A., Johnsen, S.: Co-evolution to the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches. In: Artificial Life II, pp. 325–369. Addison-Wesley, Reading (1992)
10. Landrin-Schweitzer, Y., Collet, P., Lutton, E.: Introducing lateral thinking in search engines. Genetic Programming and Evolvable Machines 7(1), 9–31 (2006)
11. Louchet, J., Guyon, M., Lesot, M.-J., Boumaza, A.M.: Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. Pattern Recognition Letters 23(1-3) (2002)

12. Mitchell, M.: When will a genetic algorithm outperform hill-climbing? In: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufman, San Francisco (1993)
13. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Proc. of the First European Conference on Artificial Life, pp. 245–254. MIT Press, Cambridge (1992)
14. Moriarty, D.E., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation 5(4), 373–399 (1998)
15. Popovici, E., De Jong, K.A.: Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In: Genetic and Evolutionary Computation Conference, GECCO 2005, pp. 507–514. ACM, New York (2005)
16. Mitchell, A., Potter, De Jong, K.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
17. De Jong, K.A., Potter, M.A.: The Coevolution of Antibodies for Concept Learning. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 530–539. Springer, Heidelberg (1998)
18. Potter, M.A., De Jong, K.A.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8(1), 1–29 (2000)
19. Reeves, C., Rowe, J.: Genetic algorithms: Principles and perspectives. Kluwer, Dordrecht (2002)
20. Watson, R.A., Pollack, J.B.: Hierarchically consistant test problems for genetic algorithms. In: 1999 Congress on Evolutionary Computation, pp. 1406–1413 (1999)
21. Wiegand, R.P.: An analysis of cooperative coevolutionary algorithms, Ph.D. thesis, George Mason University (2004)
22. Wiegand, R.P., Liles, W., De Jong, K.: Analyzing cooperative coevolution with evolutionary game theory. In: Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, pp. 1600–1605 (2002)
23. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 1235–1242. Morgan Kaufmann, San Francisco (2001)

Conditioning, Halting Criteria and Choosing λ

Olivier Teytaud

Equipe TAO (Inria), LRI, UMR 8623 (CNRS - Université Paris-Sud),
bât 490 Université Paris-Sud 91405 Orsay Cedex France
olivier.teytaud@inria.fr

Abstract. We show the convergence of $1+\lambda$-ES with standard step-size update-rules on a large family of fitness functions without any convexity assumption or quasi-convexity assumptions ([3,6]). The result provides a rule for choosing λ and shows the consistency of halting criteria based on thresholds on the step-size.

The family of functions under work is defined through a condition-number that generalizes usual condition-numbers in a manner that only depends on level-sets. We consider that the definition of this condition-number is the relevant one for evolutionary algorithms; in particular, global convergence results without convexity or quasi-convexity assumptions are proved when this condition-number is finite.

1 Introduction

We consider here a $1+\lambda_t$-ES algorithm as in Algorithm 1. We will, in a more general framework than state of the art papers (in spite of the fact that the functions are unimodal), show: (i) conditions under which the halting criterion ensure a good final output (Section 2); (ii) how to choose λ (Sections 3 and 4); (iii) the convergence of the algorithm (Section 5).

The state of the art contains convergence proofs on simple functions (e.g. the sphere function [4,1,2]), or more general lower bounds ([7,10]), or for simplified algorithms. In fact, the positive results are essentially convergence results for convex of quasi-convex fitness functions (*i.e.*, functions for which level sets are convex); this is not close to the practice of evolutionary algorithms, which can follow long non-convex valleys as in e.g. Rosenbrock's banana function. We here show our convergence on hypothesis which do not imply neither convexity nor quasi-convexity.

2 The Model and the Consistency of the Halting Criterion

Assume that the fitness is such that

$$\forall v \in \mathbb{R}, \texttt{fitness}^{-1}(v) = g(v)E_v \tag{1}$$

where $E_v \subset \mathbb{R}^d$ and where g is an increasing mapping $[0,\infty[\rightarrow [0,\infty[$ with $g(0)=0$. This implies that the $\inf \texttt{fitness} = 0$ and $\texttt{fitness}(0) = 0$; as the

Algorithm 1. $1+\lambda_t$-ES. The population size λ_t depends on t. The halting criterion depends on the mutation strength σ. The $N_{t,i}$ are usually, but not necessarily, independent Gaussians. λ_t will be chosen as in Equation 13 (quasi-random case, Section 3) or Equation 15 (random case, Section 4).

```
initialize $x_1 \in \mathbb{R}^d$, $\sigma_1 > \sigma_0$, $t = 1$.
while $\sigma_t \geq \sigma_0$ do
  Update $\lambda_t$ (Equation 13 or 15).
  for $i \in \{1, \ldots, \lambda_t\}$ do
    $x(i) = x_t + \sigma_t N_{t,i}$.
  end for
  $x' = \arg\min_{x \in \{x_t, x(1), \ldots, x(\lambda)\}} \texttt{fitness}(x)$.
  if $\texttt{fitness}(x') < \texttt{fitness}(x_t)$ then
    Acceptance for time step $t$: $x_{t+1} = x'$.
    Choose $\sigma_{t+1} > \sigma_t$.
  else
    Rejection for time step $t$: $x_{t+1} = x_t$.
    Choose $\sigma_{t+1} < \sigma_t$.
  end if
  $t = t + 1$.
end while
Output $x'$.
```

algorithm is translation-invariant (both in the fitness-space and in the domain) this does not reduce the generality. As the algorithm only uses comparisons, we can equivalently consider Equation 2 (*i.e.*, $g(v) = v$):

$$\forall v \in \mathbb{R},\ \texttt{fitness}^{-1}(v) = vE_v \tag{2}$$

$$\text{and we assume } \forall v \in \mathbb{R},\ E_v \subset \bigcup_{B^o(z,1) \subset \bigcup_{v'<v} v'E_{v'}} S(z,1) \tag{3}$$

where $B^o(x,r) = \{t; ||t-x|| < r\}$ and $S(x,r) = \{t; ||t-x|| = r\}$.The constant 1 is arbitrary, but we can rescale E_v; in fact, the hypothesis is that for some ϵ, a level-set vE_v is included in the union of all spheres of radius $v\epsilon$ enclosing areas of lower fitness. We let

$$C(\texttt{fitness}) = \inf_{(E_v)_{v\in[0,\infty[}\text{such that (2) and (3) hold}} \sup_v \sup_{e \in E_v} ||e||.$$

This equation is not simple. The family $(E_v)_{v\in[0,\infty[}$ is not uniquely determined by the fitness function; we consider the inf for all possible families $(E_v)_{v\geq 0}$ such that Equations 2 and 3 hold. There is also a supremum of $||e||$ for all $v \geq 0, e \in E_v$.

$C(\texttt{fitness})$ depends on the shape of the level-sets of the fitness, can be seen as a condition number, dedicated to comparison-based algorithms. For example, for the sphere-function, $E_v = E$ (independent of v) and $E = S$ (the unit sphere in $\mathbb{R}^d$) and $C(\texttt{fitness}) = 1$. This number is finite for many, many fitness-functions; mainly, the level-sets have to be connected. For example, the fitness function with level-sets as in Fig. 1 has a finite $C(\texttt{fitness})$. Another nice property is that this condition-number is finite for quadratic fitness functions and generalizes the classical condition-number of quadratic fitness functions. Yet another feature is illustrated by experiments in Fig. 2: an infinite $C(\texttt{fitness})$ can lead to premature convergence of $1+\lambda$-ES.

We claim:

Main lemma for the halting criterion. *Assume that eqs 2 and 3 hold.*

$$\begin{aligned} \textit{If for all } t \in \mathbb{N}, \quad & \epsilon S \subset \cup_{i\in\{1,\ldots,\lambda_t\}} B^o(N_{t,i},\epsilon) && (4)\\ \textit{and if } \sigma_T < \sigma_0, \textit{ then} \quad & \texttt{fitness}(x_T) \leq \epsilon\sigma_{T-1} && (5)\\ \textit{and} \quad & ||x_T|| \leq \epsilon\sigma_{T-1}C(\texttt{fitness}). && (6)\end{aligned}$$

Proof: Assume that eqs 2 and 3 hold, and that for all t,

$$\epsilon S \subset \cup_{i\in\{1,\ldots,\lambda_t\}} B(N_{t,i},\epsilon).$$

Then,

$$\begin{aligned} \text{Equation 3 leads to} \quad & E_v \subset \cup_{B^o(z,1)\subset\cup_{v'<v}\frac{v'}{v}E_{v'}} S(z,1) && (7)\\ \text{which leads to} \quad & e \in E_v \Rightarrow \exists f; ||f-e|| = 1 \wedge B^o(vf,v) \subset \cup_{v'<v} v'E_{v'}. && (8)\end{aligned}$$

Assume that $\sigma_T < \sigma_0$, and that T is minimal with this condition (as $t \mapsto \texttt{fitness}(x_t)$ is non-increasing, there's no loss of generality in this assumption).

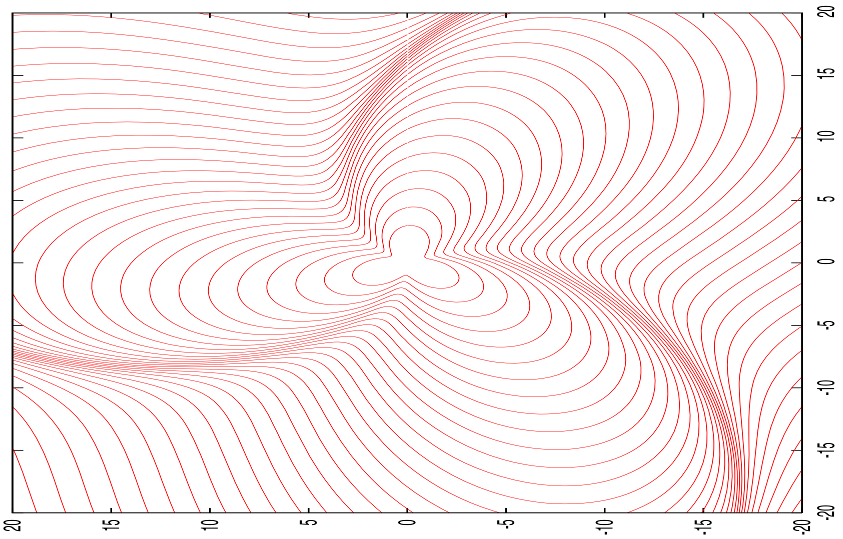


Fig. 1. An example of fitness-function (level sets are plotted) with finite $C(\texttt{fitness})$. The fitness is not convex; it is also not quasi-convex. Much more complicated examples can be defined; mainly, we need level sets which all contain a "wide" path to the optimum (at least with width scaling as the level set).

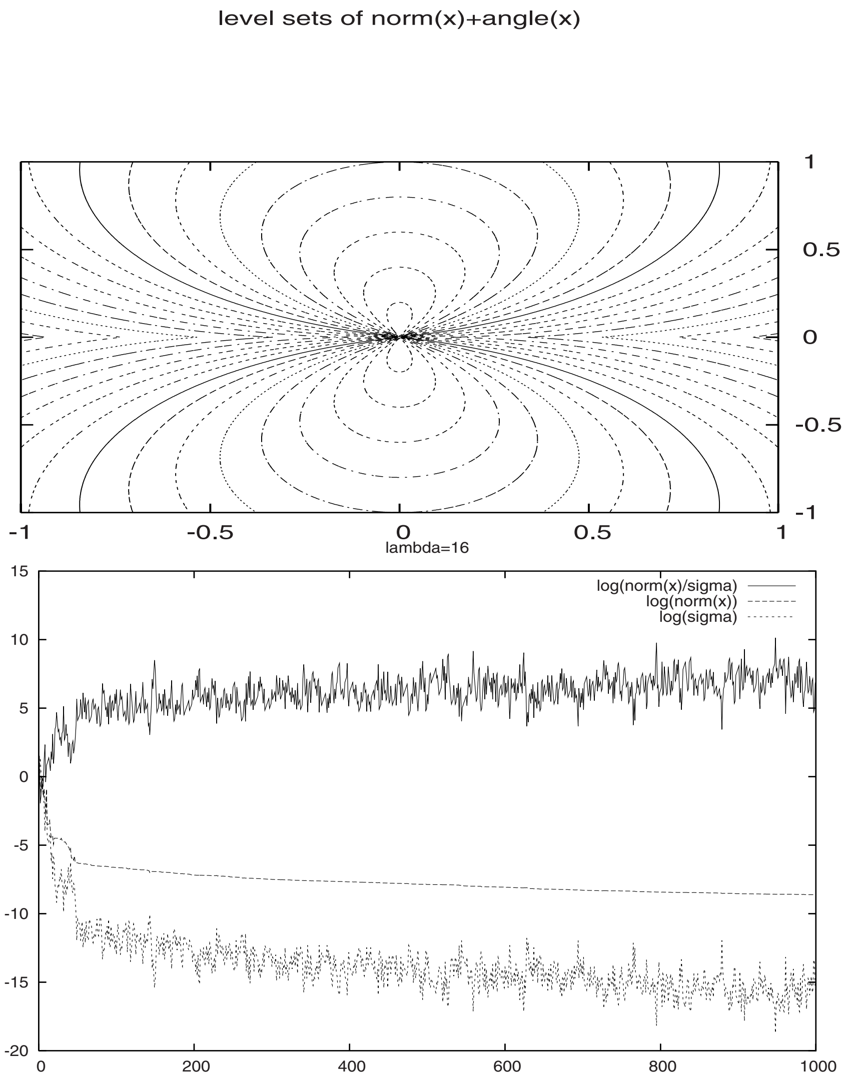


Fig. 2. Level sets of a simple function ($x \mapsto ||x|| + angle(x)^2$, with $angle(x)$ the angle between x and an axis) with infinite $C(\texttt{fitness})$ (top), and results of $1 + \lambda$-ES with $\lambda = 16$ and one-fifth rule ([9,8]) on this function (bottom). We see that σ falls down, without convergence: this is a premature convergence which illustrates Corollary 1.

This implies that at $t = T - 1$, we have a reject; therefore,

$$\forall i \in \{1, \ldots, \lambda_t\}, \texttt{fitness}(x(i)) \geq \texttt{fitness}(x_t)). \tag{9}$$

Let $x = x_t$ for short. Equation 2 implies that

$$x = \texttt{fitness}(x)e \text{ for some } e \in E_{\texttt{fitness}(x)}. \tag{10}$$

Equation 9 leads to, for all $i \in \{1, \ldots, \lambda_t\}$,

$$\begin{aligned}
&x + \sigma_t N_{t,i} \notin \cup_{v' < \texttt{fitness}(x)} v' E_{v'} \text{ and successively:}\\
&\texttt{fitness}(x)e + \sigma_t N_{t,i} \notin \cup_{v' < \texttt{fitness}(x)} v' E_{v'}\\
&\qquad \text{with } e \text{ as in Equation 10,}\\
&\texttt{fitness}(x)e + \sigma_t N_{t,i} \notin B^o(\texttt{fitness}(x)f, \texttt{fitness}(x))\\
&\qquad \text{with } f \text{ as in Equation 8,}\\
&\sigma_t N_{t,i} \notin B^o(\underbrace{\texttt{fitness}(x)f - \texttt{fitness}(x)e}_{=r}, \texttt{fitness}(x))\\
&\qquad \text{with } ||r|| = \texttt{fitness}(x) \text{ by Equation 8,}\\
&\sigma_t N_{t,i} \notin B^o(r, \texttt{fitness}(x)),\\
&N_{t,i} \notin B^o(r/\sigma_t, \texttt{fitness}(x)/\sigma_t),\\
&N_{t,i} \notin B^o(\delta, ||\delta||)\\
&\qquad \text{where } \delta = r/\sigma_t \text{ verifies } ||\delta|| = \texttt{fitness}(x)/\sigma_t.
\end{aligned}$$

We assume, to get a contradiction, that

$$||\delta|| = \texttt{fitness}(x)/\sigma_t \geq \epsilon. \tag{11}$$

Then, Equation 11, together with $c > 1 \Rightarrow B(a, ||a||) \subset B(c.a, c||a||)$, implies

$$\forall i \in \{1, \ldots, \lambda_t\}, N_{t,i} \notin B^o(\epsilon \frac{1}{||\delta||}\delta, \epsilon).$$

This is a contradiction with the assumption that for all t, $\epsilon S \subset \cup_{i \in \{1,\ldots,\lambda_t\}} B(N_{t,i}, \epsilon)$. Therefore, Equation 11 does not hold. Hence, $\texttt{fitness}(x) < \epsilon\sigma_t = \epsilon\sigma_0$. This leads to Equation 5; Equation 5 and Equation 2 lead to Equation 6. $\square$

3 Choosing λ in the Derandomized Setting

Equation 4 (recalled below, Equation 12, in the case $\epsilon = 1$) is the main assumption of the main lemma above:

$$S \subset \cup_{i \in \{1,\ldots,\lambda_t\}} B^o(N_{t,i}, 1). \tag{12}$$

We consider $\epsilon = 1$ as this hypothesis has moderate impact on the result; the results below are similar with other values of ϵ. We now study how to ensure Equation 12. A solution consists in using a minimal 1-cover of S; $\lambda_t = \lambda_{QR}$ where

$$\lambda_{QR} = \inf\{\lambda \in \mathbb{N}; d_1, \ldots, d_\lambda \in S^\lambda; S(0,1) \subset B^o(d_1, 1) \cup B^o(d_2, 1) \cup \cdots \cup B^o(d_\lambda, 1)\}. \tag{13}$$

It is known ([5]) that $\lambda_{QR} \geq c\cos(\phi_1)/\sin(\phi_1)^d d^{3/2} \ln\left(1 + d^2\cos(\phi_1)\right)$, with $\phi_1 = \arg\cos(1/2)$. This leads to λ_{QR} of order roughly $1/\sin(\phi_1)^d$; the exponential dependency in d can not be removed.

Let's show that we can not ensure the halting criterion without at least λ_{QR} points, for any deterministic offspring ($N_{t,i} = N_{1,i}$ deterministically fixed).

Corollary 1 (lower bound on λ for deterministic offsprings). *If $\lambda < \lambda_{QR}$ and for any fixed $N_{t,1}, \ldots, N_{t,\lambda}$ independent of t, then there exists an update rule for σ (see Algorithm 1), σ_0, and a function* `fitness` *verifying eqs 2 and 3, such that $\sigma_T < \sigma_0$ and $||x_T|| > \sigma_{T-1}C($*`fitness`*$)$.*

Proof: We build a counter-example with $T = 2$, $\mathtt{fitness}(x) = ||x||$, any update rule setting $\sigma_{t+1} = 0$ in case of rejection, $\sigma_1 = 1$. Then, for all $v > 0$, $E_v = S = S(0, 1)$.
We just have to choose $x_1 \in S$ such that for all $i \in \{1, \ldots, \lambda\}$,

$$x_1 + N_{t,i} \notin B^o(0, 1)$$

or equivalently, we need, for building the counter-example, an x such that for all $i \in \{1, \ldots, \lambda\}$,

$$N_{t,i} \notin B^o(-x_1, 1)$$

$$i.e., \quad ||N_{t,i} + x|| \geq 1;$$

such an x_1 exists by equation 13, as soon as $\lambda < \lambda_{QR}$, as the $B^o(N_{t,i}, 1)$ can't cover S. □

We note $d_1^{QR}, \ldots, d_{\lambda_{QR}}^{QR}$ the points realizing Equation 13; these points are by definition a minimal covering of the sphere by open balls of radius ones with centers on the sphere.

4 Choosing λ_t in the Random Case

We now consider $N_{t,1}, \ldots, N_{t,\lambda_t}$ independently randomly uniformly drawn in $S(0, 1)$. The question is: for which values of λ_t do we ensure Equation 4 (or 12) with probability $1 - \delta$? We set

$$N = \inf\{\lambda \in \mathbb{N}; y_1, \ldots, y_\lambda \in S^\lambda; S(0,1) \subset B^o(y_1, \frac{1}{2}) \cup B^o(y_2, \frac{1}{2}) \cup \cdots \cup B^o(y_\lambda, \frac{1}{2})\}. \tag{14}$$

The formula of N is close to Equation 13 but with radius $\frac{1}{2}$ instead of 1. [5] shows that $N \leq c\cos(\phi_2)/\sin(\phi_2)^d d^{3/2} \ln\left(1 + d\cos(\phi_2)^2\right)$ with $\phi_2 = 2\arg\sin(1/4)$; roughly, N is of order $O(1/\sin(\phi_2)^d)$. It is not possible to get rid of the exponential dependency in d.

Theorem 2. *Assume that*

$$\lambda_t \geq N\left(\log(N) + \log(t^2) + \log(1/\delta) - \log(\pi^2/6)\right) \tag{15}$$

and that the $N_{t,i}$ are independently uniformly drawn on $S(0, 1)$. Then, Equation 12 holds with probability at least $1 - \delta$.

Before the proof of this result, let's show a simple corollary, based on theorem 2 and on the main lemma:

Corollary 3 for algorithm 1. Assume that eqs 2 and 3 hold, and that

$$\lambda_t \geq N\left(\log(N) + \log(t^2) + \log(1/\delta) - \log(\pi^2/6)\right)$$

with $N_{t,i}$ independent random variables uniform on S. Then, with probability at least $1-\delta$, $\sigma_T < \sigma_0 \Rightarrow ||x_T|| \leq \sigma_{T-1} C(\texttt{fitness})$.

Remark A. If the step-size adaptation rule is of the form $\sigma_{n+1} = \beta\sigma_n$ in case of rejection, then the result implies $\sigma_T < \sigma_0 \Rightarrow ||x_T|| \leq \sigma_0 C(\texttt{fitness})/\beta$.

Remark B: Gaussian mutations. We use spheres instead of Gaussians as it is more parsimonious (λ smaller) than in the case of Gaussians; however, the result is essentially the same with Gaussians. With just have to add a multiplicative factor in Equation 17 in the proof below (the factor is polynomial in d).

Proof of the corollary: Application of theorem 2 and of the main lemma. □

Let's now show theorem 2.

Proof of Theorem 2: Assume that

$$\lambda_t \geq N\left(\log(N) + \log(t^2) + \log(1/\delta) + \log(\sum_{i\geq 1} 1/i^2)\right).$$

This is equivalent to Equation 15. We note δ_t the probability of Equation 4 with $\epsilon = 1$, namely δ_t is the probability of

$$S \subset \cup_{i\in\{1,\ldots,\lambda_t\}} B^o(N_{t,i}, 1). \tag{16}$$

We let $y_1, \ldots, y_n$ be elements of S realizing Equation 14. We see that if

$$\forall i \in \{1,\ldots,N\}, \exists j \in \{1,\ldots,\lambda_t\} N_{t,j} \in B^o(y_i, \frac{1}{2}),$$

then Equation 16 holds.

Therefore, with μ the uniform measure,

$$\delta_t \leq \sum_i \pi_j \left(1 - P(N_{t,j} \in B^o(y_i, \frac{1}{2}))\right),$$

$$\delta_t \leq N(1-1/N)^{\lambda_t} \text{ as } \mu\left(B^o(y_i, \frac{1}{2}) \cap S\right) \geq \mu(S)/N, \tag{17}$$

$$\log(\delta_t) \leq \log(N) + \lambda_t \log(1-1/N) \leq \log(N) - \lambda_t/N.$$

Then,
$$\sum_{t\geq 1} \delta_t \leq N\exp(-\lambda_t/N)$$

$$\leq \sum_t \delta\left(\sum_{i\geq 1} 1/i^2\right)/(t^2)$$

$$\leq \delta$$ which is the expected result. □

5 Convergence Issues: $1 + \lambda$-ES Almost Surely Halts

We have considered above the risk of raising the halting criterion before a good fitness value is met. This is meaningless, however, if we do not show that, after a finite time, the halting criterion will be met.

Theorem 4: almost sure convergence. *We assume that the update rules are as follows:*

- $\sigma_{t+1} = \min(\alpha\sigma_t, \sigma_{max})$ *in case of acceptance* ($\alpha > 1$)*;*
- $\sigma_{t+1} = \beta\sigma_t$ *in case of rejection* ($0 < \beta < 1$)*.*

We assume that Equations 2 and 3 hold for some $\epsilon > 0$*. We assume that the measure* $\mu([0,1[E_v)$ *of* $[0,1[E_v > G > 0$*. We assume that* $C(\texttt{fitness}) < \infty$ *and* $N_{t,i}$ *are independent standard multivariate Gaussians. We also assume that* $\lambda_t \leq Zt^\zeta$ *for some* $Z < \infty, \zeta < 1$*. Then, almost surely,* $\exists T > 0, \sigma_T < \sigma_0$*, i.e., the algorithm halts.*

Proof
We note $T = \inf\{t; \sigma_t < \sigma_0\}$ (possibly, a priori, $T = \infty$). We first point out some simple useful facts about the $(\sigma_t)_{t\in\mathbb{N}}$:

1. $\forall t > 0, \sigma_t \leq \sigma_{max}$.
2. $\forall t < T, \sigma_t \geq \sigma_0$.
3. If rejection holds at all steps $t+1, \ldots, t+n_0$, with $n_0 \geq \log(\sigma_{max}/\sigma_0)/\log(1/\beta)$, then $T \leq t + n_0 + 1 < \infty$.

Now, some simple facts about the $(x_t)_{t\in\mathbb{N}}$:

1. $t \mapsto \texttt{fitness}(x_t)$ is non-increasing.
2. $||x_t|| \leq C\texttt{fitness}(x_t) \leq C\texttt{fitness}(x_0)$.
3. Thanks to $t \leq T \Rightarrow (\sigma_t \geq \sigma_0 \wedge ||x_t|| \leq C.\texttt{fitness}(x_t) \leq C.\texttt{fitness}(x_0)$,

$$\begin{aligned} & P(\texttt{fitness}(x_t) < \epsilon | x_{t-1}, \sigma_{t-1}) \\ > {} & P(x_t + \sigma_t N_{t,1} \in cE_c | x_{t-1}, \sigma_{t-1}) \\ > {} & P(N_{t,1} \in (cE_c - x_t)/\sigma_t | x_{t-1}, \sigma_{t-1}) \\ > {} & c^d \mu(E_c) d\left((||x_t|| + cC(\texttt{fitness})) / \sigma_t \right) \\ > {} & K\epsilon^d \end{aligned}$$

   for some $K > 0$ that only depends on d, Z, and σ_0.
4. The previous point implies that $P(\exists u < t; \texttt{fitness}(x_u) < \epsilon) > 1 - (1 - K\epsilon^d)^t$, and therefore if $d' < d$,

$$P\left(\exists u < t; \texttt{fitness}(x_u) < (1/t)^{1/d'}\right) > 1 - \left(1 - K/t^{d/d'}\right)^t \to 1 \text{ as } t \to \infty.$$

5. The previous points implies that if $d' > d$, then almost surely, there exists $t_0 < \infty$ such that

$$t \geq t_0 \Rightarrow \texttt{fitness}(x_t) < t^{-1/d'}. \quad (18)$$

Let's now consider the probability p_t of rejection at steps t, conditionally to x_t and σ_t, conditionally to $t \leq T$.

We point out that if $\forall i \leq \lambda_t, \sigma_t||N_{t,i}|| > ||x_t|| + \texttt{fitness}(x_t)C$, then there is rejection (all x'_i have in that case norm $> C\texttt{fitness}(x_t)$ and therefore have fitness $> \texttt{fitness}(x_t)$). This implies that

$$\begin{aligned}
p_t &\geq 1 - (P(\sigma_t||N_{t,1}|| > ||x_t|| + C\texttt{fitness}(x_t)))^{\lambda_t} \\
&\geq 1 - (P(\sigma_0||N_{t,1}|| > C\texttt{fitness}(x_t) + C\texttt{fitness}(x_t)))^{\lambda_t} \\
&\qquad\qquad \text{as } \sigma_t \geq \sigma_0 \text{ and } ||x_t|| \leq C\texttt{fitness}(x_t) \\
&\geq 1 - (P(\sigma_0||N_{t,1}|| > 2C\texttt{fitness}(x_t)))^{Zt^\zeta} \text{ as } \lambda_t \leq Zt^\zeta \\
&\geq 1 - \left(P\left(\sigma_0||N_{t,1}|| > 2Ct^{-1/d'}\right)\right)^{Zt^\zeta} \text{ if } t \geq t_0 \text{ thanks to Equation 18} \\
&\geq 1 - \left(1 - 1/t^{d/d'}\right)^{Zt^\zeta} \\
&\geq p_0 > 0 \text{ if we choose } d' \text{ s.t. } d < d' < d/\zeta.
\end{aligned}$$

The probability of rejection at all steps $t+1, \ldots, t+n_0$, conditionally to x_t and σ_t, is therefore at least $1-(1-p_0)^{n_0} > p > 0$. This quantity is lower bounded by a positive number; this implies that almost surely, such a sequence of rejections almost surely occurs, hence the expected result. □

6 Discussion: Derandomization, Halting Criteria, Robustness, Conditioning

Let's summarize our results about the $1+\lambda$-ES for fitness functions with not-too-bad conditioning in the sense of Equations 2, 3 and $C(\texttt{fitness})$ with $\lambda_t = O(t^\zeta)$ for some $\zeta < 1$, and with an update rule for σ as in Theorem 4. By Theorem 4, we know that the algorithm converges almost surely (*i.e.*, it halts after a finite number of time steps). By Corollaire 3, we know that if the population size verifies Equation 15, then with probability at least $1-\delta$, the algorithm stops close to the optimum - within distance $\sigma_0 C(\texttt{fitness})\beta$. $C(f)$ quantifies the conditioning, and is finite also for many non-convex functions. Therefore, we have, for some λ_t logarithmic in t:

- global convergence with high probability;
- consistency of the halting criterion, *i.e.*no premature convergence.

A main strength of this result is that no convexity, no smoothness, no quasi-convexity is assumed and we have global convergence; see Fig. 1. As far as we know, there's no convergence proof of $1+\lambda$-ES that is not covered by the results in this paper. Another strength is that $C(\texttt{fitness})$ appears as an important relevant criterion for evolutionary algorithms: it generalizes the usual conditioning (which is a local criterion), and:

- Fig. 2 shows that very simple functions with $C(\texttt{fitness})$ lead to premature convergence;

– corollary 3 and Theorem 4 show that $C($`fitness`$)$ finite leads to both (i) convergence with high probability (ii) consistency of the halting criterion.

$C($`fitness`$)$ only depend on level sets, as well as the behavior of most evolutionary algorithms, and is finite for many fitness functions without convexity or quasi-convexity; mainly, it assumes that at each scale, the width of the path to the optimum scales as the diameter of the level set. We believe that the definition of $C($`fitness`$)$ is the main contribution of this paper.

A weakness is that we ensure convergence, and the efficiency of the halting criterion, but there's no convergence rate. However, evolutionary algorithms are more well known for robustness than for convergence rates. Moreover, a convergence rate can easily be derived under some slightly stronger assumptions.

Our results propose a rule for choosing λ_t as a function of t, δ, d (see Equations 14 and 15). This rule is reasonable for its dependency in t and δ (logarithmic dependency); the dependency in the dimension is prohibitively high, but it is a fact that evolutionary algorithms are not stable in front of large dimensionality.

We see in the results above that:

– the population size should scale as
  - $\log(t)$ (recall that $\log(t^2) = 2\log(t)$); population size should therefore increase with time (very slowly).
  - $\log(1/\delta)$; more robustness requires a bigger population size.
  - $N\log(N)$, which is exponential in d.

– we can compare the number of points required for avoiding too early convergence of the algorithm in the randomized and in the derandomized setting by comparing λ_{QR} (Equation 13) and λ_t (random case, Equation 15); in both cases, λ is exponential in d, but with a much better constant in the derandomized case. On the other hand, the convergence proof (theorem 3) only holds for the random case.

Acknowledgments. We thank Anne Auger for fruitful talks and Pierre Liardet for constructive remarks on a preliminar version of this paper. This work was supported in part by the Pascal Network of Excellence.

References

1. Auger, A.: Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible markov chains. Theoretical Computer Science 334, 35–69 (2005)
2. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: Press, A. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 445–452 (2006)
3. Bäck, T.: Evolutionary Algorithms in theory and practice. Oxford University Press, York (1995)
4. Bienvenue, A., Francois, O.: Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. Theor. Comput. Sci. 306(1-3), 269–289 (2003)

5. Boroczky, K., Wintsche, G.: Covering the sphere by equal spherical balls. In: Discrete and Computational Geometry, pp. 237–253 (2003)
6. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
7. Jagerskupper, J., Witt, C.: Runtime analysis of a (mu+1)es for the sphere function. Technical Report ISSN 1433-3325, University of Dortmund (2005)
8. Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning Probability Distributions in Continuous Evolutionary Algorithms - A Comparative Review. Natural Computing 3, 77–112 (2004)
9. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
10. Teytaud, O., Gelly, S.: General lower bounds for evolutionary algorithms. In: Tenth International Conference on Parallel Problem Solving from Nature (PPSN 2006) (2006)

Log-Linear Convergence and Optimal Bounds for the $(1+1)$-ES

Mohamed Jebalia[1], Anne Auger[1], and Pierre Liardet[2]

[1] TAO Team, INRIA Futurs
Université Paris Sud, LRI
91405 Orsay cedex, France
{mohamed.jebalia,anne.auger}@lri.fr
[2] Université de Provence
UMR-CNRS 6632, 39 rue F. Joliot-Curie
13453 Marseille cedex 13, France
liardet@cmi.univ-mrs.fr

Abstract. The $(1+1)$-ES is modeled by a general stochastic process whose asymptotic behavior is investigated. Under general assumptions, it is shown that the convergence of the related algorithm is sub-log-linear, bounded below by an explicit log-linear rate. For the specific case of spherical functions and scale-invariant algorithm, it is proved using the Law of Large Numbers for orthogonal variables, that the linear convergence holds almost surely and that the best convergence rate is reached. Experimental simulations illustrate the theoretical results.

1 Introduction

Evolutionary algorithms (EAs) are bio-inspired stochastic search algorithms that iteratively apply operators of variation and selection to a population of candidate solutions. Among EAs, adaptive Evolution Strategies (ESs) are recognized as state of the art algorithms when dealing with continuous optimization problems. Adaptive ESs sequentially adapt the parameters of the search distribution, usually a multivariate normal distribution, based on the history of the search. Several adaptation schemes have been introduced in the past. The one-fifth success rule [1,2] considers the adaptation of one parameter, referred as the step-size, based on the success probability. The most advanced adaptation scheme, the Covariance Matrix Adaptation (CMA), adapts the full covariance matrix of the multivariate normal distribution [3].

The first theoretical works carried out in the context of Evolution Strategies focused on the so-called progress rate defined as a one-step expected progress towards the optimum [1,4]. The progress rate approach consists in looking for step-sizes maximizing the expected progress. This amounts to investigating an artificial step-size adaptation scheme called scale-invariant, in which, at each iteration, the step-size is proportional to the distance to the optimum. The results derived in the context of the progress rate theory hold asymptotically in the dimension of the search space and the techniques used do not allow to obtain finite dimension estimations.

Finite dimension results were obtained in the context of 'comma' strategies on the class of the so-called sphere functions, mapping $\mathbb{R}^d$ into $\mathbb{R}$ (d being the dimension of the search space) and defined as

$$f(x) = \mathrm{g}(\|x\|^2)\,, \tag{1}$$

where $\mathrm{g} : [0, +\infty[\mapsto \mathbb{R}$ is an increasing function and $\|.\|$ denotes the usual euclidian norm on $\mathbb{R}^d$. On this class of functions, scale-invariant ESs [5] and self-adaptive ESs (which use a real adaptation rule) [5,6] do converge (or diverge) with order one, or log-linearly[1].

In this paper, finite dimension results are investigated and the focus is on the simplest ES, namely the (1+1)-ES. Section 2 introduces the mathematical model associated to the algorithm in a general framework and provides preliminary results. In Section 3, a sharp lower bound of the log-convergence rate is proved. In Section 4, it is shown that this lower bound is reached for a scaled-invariant algorithm on the class of sphere functions. The proof of convergence on the class of sphere functions uses the Law of Large Numbers for orthogonal random variables. A central limit theorem is also derived from this analysis. In Section 5 our results are discussed and related to previous works. Some numerical experiments illustrating the theoretical results are presented.

2 Mathematical Model for the $(1+1)$-ES

Let $\mathbb{R}^d$ be equipped with the Borel σ-algebra and the Lebesgue measure. In the sequel we always assume that $(\mathcal{N}_n)_n$ denotes a sequence of random vectors (r.vec.) independent and identically distributed (i.i.d.), defined on a suitable probability space (Ω, P), with common law the multivariate isotropic normal distribution on $\mathbb{R}^d$ denoted by $\mathcal{N}(0, I_d)$ ([2]). Let $(\sigma_n)_n$ be a given sequence of positive random variables (r.var.). We also assume that for each index n, σ_n is defined on Ω and is independent of $\mathcal{N}_n$; further we will also require that the sequences $(\sigma_n)_n$ and $(\mathcal{N}_n)_n$ are mutually independent. Finally, let $f : \mathbb{R}^d \to \mathbb{R}$ be an objective function (which is always assumed to be Lebesgue measurable) and let $\delta_n : \mathbb{R}^d \times \Omega \to \{0, 1\}$ ($n \geq 0$) be the measurable function defined by $\delta_n(x, \omega) := \mathbf{1}_{\{f(x+\sigma_n(\omega)\mathcal{N}_n(\omega)) \leq f(x)\}}$. In this paper, $(1+1)$-ES algorithms are modeled by the $\mathbb{R}^d$-valued random process $(X_n)_{n\geq 0}$ defined on Ω by the recurrence relation

$$\mathrm{X}_{n+1} = \mathrm{X}_n + \delta_n(\mathrm{X}_n, I_\Omega)\sigma_n\mathcal{N}_n\,, \tag{2}$$

where I_Ω is the identity function $\omega \mapsto \omega$ on Ω and X_0 is given.

[1] We say that the sequence $(\mathrm{X}_n)_n$ converges log-linearly to zero (resp. diverges log-linearly) if there exists $c < 0$ (resp. $c > 0$) such that $\lim_n \frac{1}{n} \ln \|\mathrm{X}_n\| = c$.

[2] $\mathcal{N}(0, I_d)$ is the multivariate normal distribution with mean $(0, \ldots, 0) \in \mathbb{R}^d$ and covariance matrix the identity I_d.

The classical terminology used for algorithms defined by (2) stresses the parallel with the biology: the iteration index n is referred as generation, the random vector X_n is called the parent, the perturbed random vector $\tilde{\mathrm{X}}_n = \mathrm{X}_n + \sigma_n \mathcal{N}_n$ is the n-th offspring. The scalar r.var. σ_n is called step-size. The r.var. δ_n translates the plus selection "+" in the $(1+1)$-ES: the offspring is accepted if and only if its fitness value is smaller than the fitness of the parent. Several heuristics have been introduced for the adaptation of the step-size σ_n, the most popular being the one-fifth success rule [1,2].

Notations and Preliminary Results

For a real valued function $x \mapsto h(x)$ we introduce its positive part $h^+(x) := \max\{0, h(x)\}$ and negative part $h^- = (-h)^+$. In other words $h = h^+ - h^-$ and $|h| = h^+ + h^-$. In the sequel, we denote by e_1 a unitary vector in $\mathbb{R}^d$. The following technical lemmas will be useful in the sequel.

Lemma 1. *Let $\mathcal{N}$ be a r.vec. of distribution $\mathcal{N}(0, I_d)$. The map $F : [0, \infty] \to [0, +\infty]$ defined by $F(+\infty) := 0$ and*

$$F(\sigma) := E\left[\ln^-\left(\|e_1 + \sigma\mathcal{N}\|\right)\right] = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \ln^-(\|e_1 + \sigma x\|) e^{-\frac{\|x\|^2}{2}} dx \qquad (3)$$

otherwise, is continuous on $[0, +\infty]$ (endowed with the usual compact topology), finite valued and strictly positive on $]0, \infty[$.

Proof. The integral (3) always exists but could be infinite. In any case, $F(\sigma)$ is independent of the choice of e_1 due to the invariance of $\mathcal{N}$ under rotations. For convenience we choose $e_1 = (1, 0, \ldots, 0)$ so that $\ln^-(\|e_1 + \sigma x\|) = 0$ if $x = (x_1, \ldots, x_d)$ with $x_1 \geq 0$. Let $f_1 : \mathbb{R}^d \times [0, \infty] \to [0, +\infty]$ be defined by

$$f_1(x, \sigma) = \ln^-(\|e_1 + \sigma x\|^2) e^{-\frac{\|x\|^2}{2}}$$

for $x \neq (-1/\sigma, 0, \ldots, 0)$ and $f_1((-1/\sigma, 0, \ldots, 0), \sigma) = +\infty$ (with $\sigma > 0$) and finally $f_1(x, +\infty) = 0$ $(= \lim_{\sigma \to +\infty} f_1(x, \sigma))$. Notice that $f_1(x, \sigma) = 0$ if $x_1 \geq 0$ and readily $f_1((x_1, x_2, \ldots, x_d), \sigma) = f_1((x_1, \epsilon_2 x_2, \ldots, \epsilon_d x_d), \sigma)$ for any $(\epsilon_2, \ldots, \epsilon_d)$ in $\{-1, +1\}^{d-1}$ so that we can restrict the integration giving $F(\sigma)$ to the domain $\mathcal{D} :=]-\infty, 0[\times]0, \infty[^{d-1}$, more precisely one has

$$F(\sigma) = \frac{1}{4}\left(\frac{2}{\pi}\right)^{d/2} \int_{\mathcal{D}} f_1(x, \sigma) dx \qquad (4)$$

with in addition f_1 is finite everywhere in $\mathcal{D}$. From the definition of $F(+\infty)$ and f_1 one has $\frac{1}{4}(2/\pi)^{d/2} \int_{\mathcal{D}} f_1(x, +\infty) dx = 0 = F(+\infty)$ so that (4) holds also for $\sigma = +\infty$. Now, for any real number $\sigma > 0$ fixed, the inequality $f_1(x, \sigma) > 0$ holds on $B_\sigma := \{x \in \mathcal{D}\, ;\ \|e_1 + \sigma x\| < 1\}$ which is a nonempty open set, therefore $F(\sigma) > 0$. In addition, $f_1(x, 0) = 0$ for all x and so, $F(0) = 0$. Passing to spherical coordinates (with $d \geq 2$)we obtain after partial integration

$$\int_{\mathcal{D}} f_1(x) dx = 2c_d \int_0^{+\infty} \int_0^{\pi/2} \ln^-(|\sigma r - e^{i\theta_1}|) r^{d-1} e^{-\frac{r^2}{2}} \sin^{d-2}\theta_1 dr\, d\theta_1$$

where

$$c_d = \int_0^{\pi/2} \cdots \int_0^{\pi/2} \sin^{d-3}(\theta_2) \ldots \sin(\theta_{d-2}) d\theta_2 \ldots d\theta_{d-1}$$

for $d \geq 3$ and $c_2 = 1$. With the classical Wallis integral $W_{d-2} = \int_0^{\pi/2} \sin^{d-2} \theta \, d\theta$ and the surface area of the d-dimensional unit ball $S_d = 2\pi^{d/2}/\Gamma(\frac{n}{2})$ we have $S_d = 2^d c_d W_{d-2}$ and after collecting the above results we get

$$F(\sigma) = \Big(\frac{1}{2\pi}\Big)^{d/2} \frac{1}{W_{d-2}\Gamma(\frac{d}{2})} \int_0^{+\infty} \int_0^{\pi/2} \ln^-(|\sigma r - e^{i\theta}|) r^{d-1} e^{-\frac{r^2}{2}} \sin^{d-2}(\theta) \, dr \, d\theta \, .$$

The integrand $g : (r, \theta, \sigma) \mapsto \ln^-(|\sigma r - e^{i\theta}|) r^{d-1} e^{-\frac{r^2}{2}} \sin^{d-2}(\theta)$ defined on the set $]0, +\infty[\times[0, \pi/2] \times [0, \infty]$ (with $g(r, \theta, +\infty) = 0$) is continuous. In fact, the continuity is clear at each point (r, θ, σ) with $\sigma \neq +\infty$ and for the points $(r, \theta, +\infty)$, one has $g(\rho, \alpha, \sigma) = 0$ on $]r/2, +\infty[\times[0, \pi/2]\times]\frac{4}{r}, +\infty]$. Moreover, g is dominated by $g_1 : (r, \theta) \mapsto \ln^-(\sin \theta) r^{d-1} e^{-r^2/2}$ *i.e.*, $g(r, \theta, \sigma) \leq g_1(r, \theta)$ for all (r, θ, σ) in $]0, +\infty[\times[0, \pi/2] \times [0, +\infty]$. Since g_1 is integrable, the continuity of F on $[0, +\infty]$ follows from the Lebesgue dominated convergence theorem. For the remaining case $d = 1$ the conclusions of the lemma follow easily from (4) that gives $F(\sigma) = \frac{1}{2\sqrt{2\pi}} \int_0^{\infty} \ln^-(|1 - \sigma r|) e^{-\frac{r^2}{2}} dr$. □

Corollary 1. *The supremum* $\tau := \sup F([0, +\infty])$ *is reached and* $\sigma_F := \min F^{-1}(\tau)$ *exists. Moreover* $0 < \sigma_F < +\infty$ *and* $0 < \tau < +\infty$.

Proof. This corollary is a straightforward consequence of the continuity of F according to Lemma 1 which implies that $F^{-1}(\tau)$ is nonempty and compact. □

Lemma 2. *Let X denote a r.vec. in $\mathbb{R}^d$ such that $\|X\|^{-1}$ is finite almost surely. Let σ be a non negative random variable and let $\mathcal{N}$ be a random vector in $\mathbb{R}^d$ with distribution $\mathcal{N}(0, I_d)$ and independent of $\sigma\|X\|^{-1}$. Assume that*

$$E\Big(\ln\Big(1 + r\frac{\sigma}{\|X\|}\Big)\Big) \in O(e^{cr})$$

with a constant $c \geq 0$, then the expectation of $\ln^+(\|X\|^{-1}\|X + \sigma\mathcal{N}\|)$ is finite.

Proof. Obviously $E(\ln^+(\|X\|^{-1}\|X + \sigma\mathcal{N}\|)) \leq E(\ln(1 + \frac{\sigma}{\|X\|}\|\mathcal{N}\|))$. Using the independency of $\sigma\|X\|$ and $\mathcal{N}$, and passing to the spherical coordinates, one gets

$$\begin{aligned} E\big(\ln\big(1 + \frac{\sigma}{\|X\|}\|\mathcal{N}\|\big)\big) &\leq E\big(\int_{\mathbb{R}^d} \ln(1 + \frac{\sigma}{\|X\|}\|x\|) e^{-\frac{\|x\|^2}{2}} dx\big) \\ &= S_d E\big(\int_0^{+\infty} \ln(1 + r\frac{\sigma}{\|X\|}) r^{d-1} e^{-\frac{r^2}{2}} dr\big) \\ &= S_d \int_0^{+\infty} E(\ln(1 + r\frac{\sigma}{\|X\|})) r^{d-1} e^{-\frac{r^2}{2}} dr \\ &\ll \int_0^{+\infty} r^{d-1} e^{cr - \frac{r^2}{2}} dr < +\infty \, . \end{aligned}$$

□

Remark 1. The assumption $E(\ln(1+r\frac{\sigma}{\|X\|})) \in O(e^{cr})$ (with $c = 0$) is verified if there exists $\alpha > 0$ such that the expectation of the r.var. $(\sigma/\|X\|)^{\alpha}$ is finite.

3 Lower Bounds for the (1 + 1)-ES

In this section, we consider a general measurable objective function $f : \mathbb{R}^d \to \mathbb{R}$. We prove that the (1 + 1)-ES defined by (2) for minimizing f, under suitable assumptions, satisfies for all x^* in $\mathbb{R}^d$ and all indices $n \geq 0$:

$$-\infty < E(\ln\|\mathrm{X}_n - x^*\|) - \tau \leq E(\ln\|\mathrm{X}_{n+1} - x^*\|) < +\infty \tag{5}$$

where τ is defined in Corollary 1.

If x^* is a limit point of (X_n) (that could be a local optimum of f), (5) means that the expected log-distance to x^* cannot decrease more than τ, in other words, $-\tau$ is a lower bound for the convergence rate of (1 + 1)-ES. The proof of this result uses the following easy Lemma whose proof is left to the reader.

Lemma 3. *Let Z and V be r.vec. and let Θ be any r.var. valued in $\{0,1\}$. Assume that the r.var. $\ln(\|Z\|)$ is finite almost surely. Then the following inequalities*

$$\begin{aligned}\ln(\|Z\|) - \ln^{-}(\|Z\|^{-1}\|Z+V\|) &\leq \ln(\|Z+\Theta V\|)\\ &\leq \ln(\|Z\|) + \ln^{+}(\|Z\|^{-1}\|Z+V\|)\end{aligned} \tag{6}$$

hold almost surely. □

We are ready to prove the following general theorem.

Theorem 1 (Lower bounds for the (1+1)-ES). *Let $(\mathrm{X}_n)_n$ be the sequence of random vectors verifying (2) with a given objective function f as above. Assume that for each step $n = 0, 1, 2, \ldots$ the random vector $\mathcal{N}_n$ is independent of both the random variable σ_n and the random vector X_n. Let x^* be any vector in $\mathbb{R}^d$ and suppose that $E(\big|\ln(\|\mathrm{X}_0 - x^*\|)\big|) < +\infty$ and for all $n \geq 0$,*

$$E\Big(\ln(1 + r\frac{\sigma_n}{\|X_n - x^*\|})\Big) \in O(e^{c_n r})$$

with a constant $c_n \geq 0$. Then

$$E\left(|\ln\left(\|\mathrm{X}_n - x^*\|\right)|\right) < +\infty\ ,$$

and

$$E(\ln(\|\mathrm{X}_n - x^*\|)) - \tau \leq E(\ln(\|\mathrm{X}_{n+1} - x^*\|))\,, \tag{7}$$

for all $n \geq 0$, where τ is defined in Corollary 1. In particular, the convergence of the $(1+1)$-ES is at most linear, in the sense that

$$\inf_{n\in\mathbb{N}} \frac{1}{n} E\big(\ln\left(\|\mathrm{X}_n - x^*\|/\|\mathrm{X}_0 - x^*\|\right)\big) \geq -\tau\,. \tag{8}$$

Proof. Set $Z_n = \mathrm{X}_n - x^*$, $\tilde{\mathrm{X}}_n = \mathrm{X}_n + \sigma_n \mathcal{N}_n$ and $\tilde{\mathrm{Z}}_n = \tilde{\mathrm{X}}_n - x^*$. We prove the integrability of $\ln(\|Z_n\|)$ by induction. By assumption $E\big(\ln(\|Z_0\|)\big)$ is finite. Suppose that $E\big(\ln\|Z_n\|\big)$ is finite, then $0 < \|Z_n\| < +\infty$ almost surely, hence $\ln\big(\|Z_{n+1}\|\big)$ is also finite almost surely. We claim that $E\big(\ln(\|Z_{n+1}\|)\big)$ is finite. By applying Lemma 3 we get (6) and derive

$$\ln^+(\|Z_{n+1}\|) \leq \ln^+(\|Z_n\|) + \ln^+\big(\|Z_n\|^{-1}(\|Z_n + \sigma_n \mathcal{N}_n\|)\big) . \tag{9}$$

By Lemma 2 the expectation of $\ln^+\big(\|Z_n\|^{-1}(\|Z_n + \sigma_n \mathcal{N}_n\|)\big)$ is finite and using (9) we conclude that $E\big(\ln^+(\|Z_{n+1}\|)\big) < +\infty$. It remains to show that $E\big(\ln^-(\|Z_{n+1}\|)\big)$ is also finite. Using the first inequality in (6) we obtain

$$\ln^-(\|Z_{n+1}\|) \leq -\ln(\|Z_n\|) + \ln^-\left(\left\|\frac{Z_n}{\|Z_n\|} + \frac{\sigma_n}{\|Z_n\|}\mathcal{N}_n\right\|\right) + \ln^+(\|Z_{n+1}\|) . \tag{10}$$

For each $n \geq 0$, let $\mathcal{F}_n$ denote the σ-algebra generated by the r.vec. X_n and the r.var. σ_n. Taking the conditional expectation we obtain

$$\begin{aligned} &E[\ln^-(\|Z_{n+1}\|) \,|\, \mathcal{F}_n] \\ &\leq -\ln(\|Z_n\|) + E\left[\ln^-\left(\left\|\frac{Z_n}{\|Z_n\|} + \frac{\sigma_n}{\|Z_n\|}\mathcal{N}_n\right\|\right) \,|\, \mathcal{F}_n\right] + E\left[\ln^+\left(\|Z_{n+1}\|\right) \,|\, \mathcal{F}_n\right] . \end{aligned}$$

Since the distribution $\mathcal{N}_n$ is invariant under rotation and independent of $\mathcal{F}_n$,

$$\begin{aligned} E\left(\ln^-\left(\left\|\frac{Z_n}{\|Z_n\|} + \frac{\sigma_n}{\|Z_n\|}\mathcal{N}_n\right\|\right) \,|\, \mathcal{F}_n\right) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \ln^-(\|e_1 + t_n x\|) e^{-\frac{\|x\|^2}{2}} dx \\ &= F(t_n) \end{aligned}$$

where e_1 is any unit vector on $\mathbb{R}^d$, $t_n = \sigma_n / \|Z_n\|$ (and F is the map introduced in Lemma 1). Using Lemma 1, we get $E\left[\ln^-(\|Z_{n+1}\|) \,|\, \mathcal{F}_n\right] \leq -\ln(\|Z_n\|) + \tau + E\left[\ln^+(\|Z_{n+1}\|) \,|\, \mathcal{F}_n\right]$ (recall that $\tau = \max F([0, +\infty])$). Passing to the expectation we get

$$E\left[\ln^-(\|Z_{n+1}\|)\right] \leq -E\left[\ln(\|Z_n\|)\right] + \tau + E\left[\ln^+(\|Z_{n+1}\|)\right] < +\infty .$$

Hence $E[|\ln(\|Z_{n+1}\|)|]$ is finite for all $n \geq 0$. Moreover, we also get

$$E(\ln\|Z_{n+1}\|) \geq E(\ln\|Z_n\|) - \tau$$

and after summing such inequalities we obtain

$$E\left(\ln\left(\|Z_n\| / \|Z_0\|\right)\right) \geq -\tau n$$

and (8) follows. □

When x^* is a local minimum of the objective function, $E(\ln\|\mathrm{X}_n - x^*\|) - E(\ln\|\mathrm{X}_{n+1} - x^*\|)$ represents the expected log-distance reduction towards x^* at the n-th step of iteration, called *log-progress* in [7]. Theorem 1 shows that the log-progress is bounded above by $\tau = F(\sigma_F)$.

4 Spherical Functions and the Scale-Invariant Algorithm

In this section we prove that the lower bound $-\tau$ obtained in Theorem 1 is reached for spherical objective functions when $\sigma_n = \sigma_F \|\mathrm{X}_n\|$ $(n \geq 0)$. Recall that sphere objective functions are defined by $f(x) = g(\|x\|^2)$ where g is any increasing map, so that the acceptance condition $f(\mathrm{X}_{n+1}) \leq f(\mathrm{X}_n)$ is equivalent to $\|\mathrm{X}_{n+1}\| \leq \|\mathrm{X}_n\|$. It follows that $(\|\mathrm{X}_n\|)_{n\geq 0}$ is a non-increasing sequence of positive random variables (finite almost surely), hence converges pointwise almost surely. For spherical functions, Lemma 3 becomes:

Lemma 4. *Let X and W be any random vectors and let $\Theta = \mathbf{1}_{\{f(X+W)\leq f(X)\}}$ and assume that the random variable $\ln(\|X\|)$ is finite almost surely. Then the equality*

$$\ln(\|X + \Theta W\|) - \ln(\|X\|) = -\ln^+(\|X\|^{-1}\|X + W\|) \tag{11}$$

holds almost surely.

Proof. The equality (11) emphasizes the fact that $\|X + \Theta\| \leq \|X\|$ with equality on the event $\{\Theta = 0\}$ $(= \{\|X + W\| > \|X\|\})$. □

Proposition 1. *Let $(\mathrm{X}_n)_n$ be the sequence of random vectors valued in $\mathbb{R}^d$ satisfying the recurrence relation (2) involving spherical function $f(x) = g(\|x\|^2)$ where $g : [0, \infty[\to \mathbb{R}$ is an increasing map. Assume that $E(\ln(\|X_0\|)$ is finite and that, at each step n, the random vector $\mathcal{N}_n$ is independent of both the random variable σ_n and the random vector X_n. Then $E(\ln(\|X_n\|)$ is finite for all indices n, the inequalities*

$$E(\ln(\|X_n\|) - \tau \leq E(\ln(\|X_{n+1}\|)$$

hold, where τ is defined above in Corollary 1, and

$$\ln(\|X_n\|) - \ln(\|X_{n+1}\|) = \ln^-(\|X_n\|^{-1}\|X_n + \sigma_n\mathcal{N}_n\|) < +\infty \; a.s. \tag{12}$$

Proof. By construction $\|X_{n+1}\| \leq \|X_n\| \leq \|X_0\|$ so that $E(\ln^+(\|X_{n+1}\|)) \leq E(\ln^+(\|X_0\|)) < +\infty$. Now assume that $\ln(\|X_n\|)$ is integrable, hence $0 < \|X_n\| < +\infty$ a.s. and so, by Lemma 4, to obtain the inequalities and equality asserted in the proposition it is enough to prove that $E(\ln^-(\|X_n\|^{-1}\|X_n + \sigma_n\mathcal{N}_n\|)) \leq \tau$. But similarly to the end part of the proof of Theorem 1 we have $E(\ln^-(\|X_n\|^{-1}\|X_n + \sigma_n\mathcal{N}_n\|)) = E(F(\sigma_n/\|X_n\|)) \leq \tau$. □

Now we pay attention to the particular case where $\sigma_n = \sigma\|\mathrm{X}_n\|$ with $\sigma > 0$ fixed. The resulting $(1+1)$-ES is said to be *scale-invariant*, and is modeled by the d-dimensional random process

$$\mathrm{X}_{n+1} = \mathrm{X}_n + \delta_n(\mathrm{X}_n, I_\Omega)\sigma\|\mathrm{X}_n\|\mathcal{N}_n \qquad (n \geq 0)\,. \tag{13}$$

For convenience of the reader we collect the hypothesis that govern the scale-invariant random process (13):

(HSI) *The sequence of random vectors $(\mathcal{N}_n)_n$ in $\mathbb{R}^d$ is i.i.d. with common law $\mathcal{N}(0, I_d)$, is independent of the initial random vector X_0 and $\ln(\|X_0\|)$ has a finite expectation.*

Notice that Assumption (HSI) implies in particular that for $m \geq n \geq 0$, $\mathcal{N}_m$ is independent of X_n and by Proposition 1, $\ln(\|X_n\|)$ has a finite expectation. The update rule (13) is not so realistic because in practice, at each step n, the distance of X_n to the optimum is unknown. Nevertheless, we will show that the stochastic process defined by (13) converges log-linearly for sphere functions and that for $\sigma = \sigma_F$ the convergence rate in log is equal to $-F(\sigma_F)$ $(= -\tau)$. In other words, the choice $\sigma_n = \sigma_F \|\mathrm{X}_n\|$ correspond to the adaptation scheme that gives the optimal convergence rate for isotropic Evolution Strategies.

It is usual for studying stochastic search algorithms to consider log-linear convergence of X_n by investigating the stability of $\ln\left(\|\mathrm{X}_{n+1}\|/\|\mathrm{X}_n\|\right)$. This idea was introduced in the context of ESs by Bienvenüe and François [5] and exploited in [6]. The process X_n given by (13) has a remarkable property expressed in terms of orthogonality of the random sequences $Y_n = \ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathcal{N}_n\right\|\right) - F(\sigma)$:

Proposition 2. *Consider the random variables*

$$Y_n := \ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathcal{N}_n\right\|\right) - F(\sigma)$$

where F is defined by (4) and let $\sigma > 0$. Under the hypothesis (HSI) *the followings hold:*

1. *For $n \geq 0$, $E(Y_n) = 0$ and $E(|Y_n|^2) < +\infty$.*
2. *Let $(Y'_n)_{n\geq 0}$ be the sequence of random variables*

$$Y'_n := \ln^-(\|e_1 + \sigma\mathcal{N}_n\|) - F(\sigma).$$

The random variables Y_n $(n \geq 0)$ are identically distributed and for every $n \geq 0$, Y_n and Y'_n follow the same distribution.

3. *The sequence of random variables $(Y_n)_{n\geq 0}$ is orthogonal, i.e. for all indices i, j, with $i \neq j$ one has $E(Y_i) = 0$, $E(Y_i^2) < +\infty$ and $E(Y_iY_j) = 0$.*

Proof. The isotropy of the standard d-dimensional normal distribution gives

$$E\Big(\ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathcal{N}_n\right\|\right) \mid \mathrm{X}_n\Big) = \frac{1}{(2\pi)^{d/2}}\int_{\mathbb{R}^d} \ln^-(\|e_1 + \sigma x\|)e^{-\frac{\|x\|^2}{2}}dx$$
$$= F(\sigma)$$

hence $E\left[\ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathcal{N}_n\right\|\right)\right] = E\left[F(\sigma)\right]$ and so, $E(Y_n) = 0$. Let $F_2 : [0, \infty] \to [0, +\infty[$ be defined by $F_2(\infty) = 0$ and, for $t \in [0, +\infty[$,

$$F_2(t) := \frac{1}{(2\pi)^{d/2}}\int_{\mathbb{R}^d} \left[\ln^-(\|e_1 + tx\|)\right]^2 e^{-\frac{\|x\|^2}{2}}dx\,. \tag{14}$$

Similarly to the proof of Lemma 1, we prove that F_2 is continuous, hence bounded. Now, from the definitions of F and F_2 one has

$$E(|Y_n|^2) = F_2(\sigma) - (F(\sigma))^2 < +\infty\,. \tag{15}$$

This ends the proof of the first point.

The random vectors Y_n and Y'_n have the same distribution if their characteristic functions are identical. But successively

$$\begin{aligned} E(e^{itY_n} \,|\, \mathrm{X}_n) &= e^{-itF(\sigma)} E\Big(e^{it\ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|}+\sigma\mathcal{N}_n\right\|\right)} \,|\, \mathrm{X}_n\Big) \\ &= \frac{e^{-itF(\sigma)}}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{it\ln^-\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|}+\sigma x\right\|\right)} e^{-\|x\|^2/2} dx \\ &= \frac{e^{-itF(\sigma)}}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{it\ln^-\left(\left\|e_1+\sigma x\right\|\right)} e^{-\|x\|^2/2} dx \\ &= E(e^{itY'_n})\,. \end{aligned}$$

Therefore $E(e^{itY_n}) = E(E(e^{itY_n} \,|\, \mathrm{X}_n)) = E(e^{itY'_n})$. To finish the proof we show the orthogonality property of the Y_n ($n \geq 0$). Let n and m be indices such that $n < m$. The random vector Y_n is $\sigma(\mathrm{X}_n, \mathcal{N}_n)$-measurable, so that

$$E(Y_m Y_n \,|\, \mathrm{X}_n, \mathrm{X}_m, \mathcal{N}_n) = Y_n E(Y_m | \mathrm{X}_n, \mathrm{X}_m, \mathcal{N}_n)\,.$$

Using the independency of $\mathcal{N}_m$ with the random vectors. X_n, $\mathcal{N}_n$ and X_m, we get

$$\begin{aligned} E(Y_m|\mathrm{X}_n, \mathrm{X}_m, \mathcal{N}_n) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \Big(\ln^-\big(\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma x\|\big)\Big) e^{-\frac{\|x\|^2}{2}} dx - F(\sigma) \\ &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \Big(\ln^-(\|e_1 + \sigma x\|)\Big) e^{-\frac{\|x\|^2}{2}} dx - F(\sigma) = 0\,, \end{aligned}$$

that implies $E(Y_m Y_n) = 0$. □

With the above notations define the random vectors $S_n = Y_0 + \cdots + Y_n$ and $S'_n = Y'_0 + \cdots + Y'_n$. Under the hypothesis (HSI), the characteristic function of S_n can be written as $E(itS_n) = E(E(itS_n \,|\, X_0, \mathcal{N}_0, \ldots, \mathcal{N}_{n-1}))$ and so, $E(itS_n) = E(itS'_n) = (E(itY'_0))^{n+1}$. But the random vectors Y_n are i.i.d. with expectation 0 and variance $F_2(\sigma) - F(\sigma)^2$ (see (15)). As a consequence, the central limit theorem holds for both $(Y_n)_n$ and $(Y'_n)_n$:

Theorem 2. *Under the hypothesis* (HSI) *one has*

$$\lim_{n\to+\infty} P\left(\frac{\ln(\|X_n\|) - \ln(\|X_0\|) + F(\sigma)n}{\sqrt{(F_2(\sigma) - F(\sigma)^2)n}} \leq t\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{u^2}{2}} du\,.$$

The pointwise stability of $\ln\left(\|\mathrm{X}_{n+1}\|/\|\mathrm{X}_n\|\right)$ is obtained by applying the following Law of Large Numbers (LLN) for orthogonal random variables (see [10, p. 458] where a more general statement is given).

Theorem 3 (LLN for Orthogonal Random Variables). *Let $(Y_n)_{n\geq 0}$ be a sequence of identically distributed real random variables with finite variance and orthogonal, i.e., for all indices i, j, with $i \neq j$ one has $E(Y_i) = 0$, $E(Y_i^2) < +\infty$ and $E(Y_iY_j) = 0$. Then*

$$\lim_n \frac{1}{n} \sum_{k=0}^{n-1} Y_k = 0 \quad a.s.$$

We are now ready to prove the following main result

Theorem 4. *Let $\sigma > 0$ and let $(\mathrm{X}_n)_n$ be the sequence of random vectors satisfying the recurrence relation (13) with $f(x) = g(\|x\|^2)$ where g is an increasing map. Assume that the hypothesis* (HSI) *holds. Then $(\mathrm{X}_n)_n$ converges log-linearly to the minimum, in the sense that*

$$\lim_n \frac{1}{n} \ln \left(\frac{\|\mathrm{X}_n\|}{\|\mathrm{X}_0\|} \right) = -F(\sigma)(< 0) \quad \text{a.s.} \tag{16}$$

where F is defined by (4). The optimal convergence rate is obtained for $\sigma = \sigma_F := \min F^{-1}(\max F)$ (see Corollary 1).

Proof. In case $\sigma_n = \sigma\|\mathrm{X}_n\|$ for all indices n the equality (12) becomes

$$\ln \|\mathrm{X}_{n+1}\| - \ln \|\mathrm{X}_n\| = -\ln^- \left(\left\| \frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma \mathcal{N}_n \right\| \right).$$

and after summing the equations for $k = 0, \ldots, n-1$, we obtain

$$\frac{1}{n} (\ln \|\mathrm{X}_n\| - \ln \|\mathrm{X}_0\|) = -\frac{1}{n} \sum_{k=0}^{n-1} \ln^- \left(\left\| \frac{X_k}{\|X_k\|} + \sigma \mathcal{N}_k \right\| \right).$$

Proposition 2 and Theorem 3 end the proof. □

5 Discussion and Conclusion

Theorems 1 and 4 show that optimal bounds for the convergence rate of an isotropic (1 + 1)-ES with multivariate normal distribution are reached for the scale-invariant algorithm with $\sigma_n = \sigma_F\|\mathrm{X}_n\|$ for the sphere function, where σ_F maximizes

$$F(\sigma) = E(\ln^- \|e_1 + \sigma\mathcal{N}\|) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \ln^-(\|e_1 + \sigma x\|) e^{-\frac{\|x\|^2}{2}} dx \ .$$

From (12) and from the isotropy of the multivariate normal distribution $\mathcal{N}$, it follows that finding σ maximizing F amounts to finding σ maximizing the log-progress $E(\ln \|\mathrm{X}_n\|) - E(\ln \|\mathrm{X}_{n+1}\|)$.

Most of the works based on the progress rate, consist in finding σ maximizing estimations of the expected progress $E(\|\mathrm{X}_n\|) - E(\|\mathrm{X}_{n+1}\|)$ (when d goes to infinity) [1,4]. Note that the definition of progress in those works does not consider

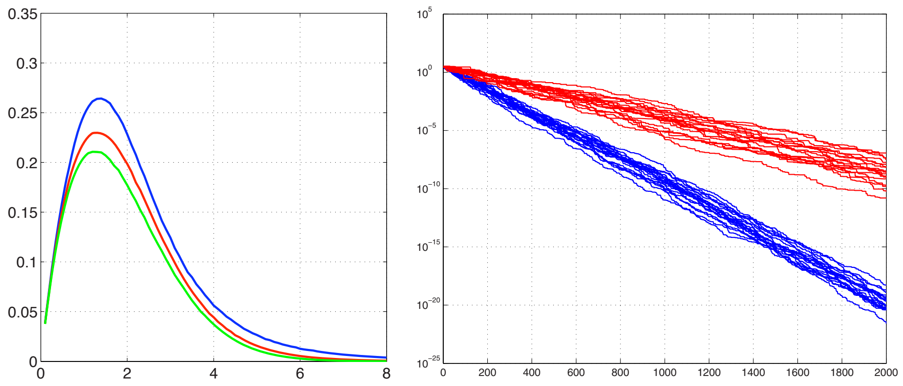


Fig. 1. Left: Plot of the function $\sigma \mapsto dF(\sigma/d)$ (Eq. (4)) versus σ for $d = 5$ (resp. 10, 30) and $0 \leq \sigma \leq 8$. The upper curve corresponds to $d = 5$, the middle one to $d = 10$ and the lower one to $d = 30$. Note that the function F defined in (4) implicitly depends on d. Using the more explicit notation F_d instead of F, the plots represent actually $\sigma \mapsto dF_d(\sigma/d)$. For $d = 10$, we see that σ_F maximizing F (defined in Corollary 1) approximately equals 0.13. The plots were obtained doing Monte Carlo estimations of F using 10^6 samples.

Right: Twenty realizations of the scale-invariant algorithm on the sphere function for $d = 10$. The y-axis shows the distance to the optimum (in log-scale) and the x-axis the number of iterations n. The twenty curves below correspond to the optimal algorithm, *ie.* $\sigma_n = \sigma_F \|\mathrm{X}_n\|$ for all n where σ_F equals to 0.13 (value maximizing the curve of F on the left for $d = 10$). The twenty curves above correspond to 20 realizations of the scale-invariant algorithm for $\sigma_n = 0.3\|\mathrm{X}_n\|$. Observed, the log-linear convergence as well as the optimality of the scale-invariant algorithm for $\sigma = \sigma_F$.

$\ln \|\mathrm{X}_n\|$ and so is different from the one underlying our study. Assuming that both definitions matches[3], our results give an interpretation of this approach in terms of lower bounds for convergence of ESs.

The lower bounds derived in this paper are tight. Consequently they can be used in practice to assess the performances of a given step-size adaptation strategy comparing the convergence rate achieved by the strategy with the optimal one, given by the scale-invariant algorithm.

The numerical estimation of the optimal convergence rate $-\tau$ can be achieved with a Monte Carlo integration: for different σ, $F(\sigma)$ equals the expectation $E(\ln^- \|e_1 + \sigma\mathcal{N}\|)$. This expectation can be estimated by summing independent samplings of the random variable $\ln^- \|e_1 + \sigma\mathcal{N}\|$. This is illustrated in Fig 1.

The analysis of the log-linear convergence carried out in this paper relies on the application of the Strong Law of Large Numbers for orthogonal random variables. This study uses deeply the invariance under rotations of the standard d-dimensional multivariate normal distribution and does not cover directly the usual case of stable Markov chains that will be investigated in future works.

[3] This will be true asymptotically in the dimension d, though we do not prove it rigorously in this paper.

Acknowledgments

The authors thank the referees for their constructive remarks on the previous version that lead to this new version and are very grateful to Nicolas Monmarché for his encouragements. This work receives partial supports from the ANR/RNTL project Optimisation Multidisciplinaire (OMD) and from the ACI CHROMALGEMA.

References

1. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart (1973)
2. Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning Probability Distributions in Continuous Evolutionary Algorithms - A Comparative Review. Natural Computing 3, 77–112 (2004)
3. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9, 159–195 (2001)
4. Beyer, H.G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
5. Bienvenüe, A., François, O.: Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. Theoretical Computer Science 306, 269–289 (2003)
6. Auger, A.: Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible Markov chains. Theoretical Computer Science 334, 35–69 (2005)
7. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 445–452 (2006)
8. Teytaud, O., Gelly, S.S.: General lower bounds for evolutionary algorithms. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 21–31. Springer, Heidelberg (2006)
9. Jägersküpper, J.: Lower bounds for hit-and-run direct search. In: Hromkovič, J., Královič, R., Nunkesser, M., Widmayer, P. (eds.) SAGA 2007. LNCS, vol. 4665, pp. 118–129. Springer, Heidelberg (2007)
10. Loève, M.: Probability Theory, 3rd edn. Van Nostrand, New York (1963)

Evolution Strategies for Laser Pulse Compression

Riccardo Fanciulli[1], Lars Willmes[2,⋆], Janne Savolainen[1], Peter van der Walle[1], Thomas Bäck[2], and Jennifer L. Herek[1,3]

[1] FOM Institute for Atomic and Molecular Physics (AMOLF)
Kruislaan 407, 1098 SJ Amsterdam, The Netherlands
[2] Natural Computing Group, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
[3] Optical Sciences Group, MESA+ Institute for Nanotechnology, Twente University
Drienerlolaan 5, 7522 NB Enschede, The Netherlands

Abstract. This study describes first steps taken to bring evolutionary optimization technology from computer simulations to real world experimentation in physics laboratories. The approach taken considers a well understood Laser Pulse Compression problem accessible both to simulation and laboratory experimentation as a test function for variants of Evolution Strategies. The main focus lies on coping with the unavoidable noise present in laboratory experimentation. Results from simulations are compared to previous studies and to laboratory experiments.

1 Introduction

The use of Evolutionary Algorithms (EAs) is already widespread in several branches of science and engineering, whenever a process or product design involving highly nonlinear and complex operating conditions needs to be optimized at a pre-production stage of modeling or planning. In some cases, however, the system to be optimized cannot be adequately modeled or it needs to adapt to some (slowly) changing external parameters that cannot be predicted at a design stage. In these cases, when feedback is measured directly from the physical system, fitness evaluations will always be polluted by noise. The effect of this noise on the performance of the EAs needs to be minimized as much as possible. Generally speaking, one expects different EAs to respond differently to noise, and even the same EA will display a different degree of robustness to noise depending on the values of certain initial settings. In the present paper we report for the first time a study on the effect of real world noise on two commonly applied variants of EAs, namely Evolution Strategies (ES) using the CMA and the DR2 selfadaptation scheme.

Taking advantage of a physical experiment of laser pulse compression through optimization (maximization) of the intensity of second harmonic generation (a

⋆ The first two authors contributed equally to this paper.

well understood process and an experimental set-up where noise can be monitored and controlled), we have run theoretical and experimental comparisons of CMA and DR2 as well as among different parameter settings of both algorithms.

Similar studies based on simulations of pulse shaping experiments have been performed in [1,2]. However, in contrast to the work presented here, the earlier simulations did not consider noise. Comparing results can reveal in how far algorithm performance changes due to the addition of noise. Additionally, in this study results of simulations are verified in the laboratory by physical experimentation.

2 Second Harmonic Generation (SHG) as Fitness Function

The goal of our optimizations is to make the pulses coming out of our laser as short as theoretically possible. In order to do that we send the laser pulses through two devices that, at this point, we will consider as black boxes. First, a control box (pulse shaper) acts on the pulses to lengthen or shorten them depending on some optimization parameters [3]. Second, a monitoring box (nonlinear crystal) produces light at double the frequency of the input pulses by a process called second harmonic generation (SHG) [4]. As we will see in more detail later, the stronger the intensity of the SHG pulses, the shorter the initial laser pulses will be. This allows us to use the intensity of the SHG pulses as the fitness function for our EAs. Next we will go into a little more detail about the system, the way to achieve control over the input pulses and the creation of second harmonic light.

2.1 Introduction to Optical Second Harmonic Generation

A Conceptual Approach. The effect of SHG of a laser beam through a crystal can be explained quite simply once some basic terminology is introduced. Whenever a pulsed laser is considered one refers to its shape and duration in time domain to characterize it. Imagining a plane being transversed by the laser pulse and recording over time the intensity of the light going through such a plane, the resulting plot of intensity vs. time is what is called the shape of the pulse in time domain. Now, to explain the concept of SHG, we will start from the idea that a pulse of light can be described as a superposition of sinusoidal functions with different frequencies and phases (Fourier description or description in frequency domain). The number of frequencies (i.e. colors) that are required to describe the pulse depends on its duration and on the complexity of its shape. The set of necessary frequencies is called the bandwidth of the pulse. The plot of the amplitude of these sinusoidals vs. their frequency is defined as the spectrum of the pulse, while the dependence of the phase term inside each sinusoidal on its associated frequency is the phase profile of the pulse. The spectrum tells us how much weight each sinusoidal has in describing the pulse, while the phase profile has to do with the relative delay of each frequency component with respect to a fixed reference (usually the frequency with the highest amplitude).

For our purpose a so called SHG crystal can be regarded as a black box in which all frequency components within the pulse spectrum are combined to give new sinusoidal components characterized by a frequency that is the sum of the input frequencies. When combining all these frequencies, the phase in each sinusoidal function plays a crucial role in determining the intensity of the SHG light in output.

A Mathematical Approach. Defining the pulse shape in time domain as $E(t)$ and its corresponding spectrum as $\tilde{E}(\omega)$, the SHG pulse in output from the crystal is given simply by:

$$E_{SHG}(t) = E^2(t) \tag{1}$$

In frequency domain, one can use the property of the Fourier transforms that relates a simple product of two functions in one domain to the convolution integral of the two functions separately-transformed in the other domain:

$$\tilde{E}(\omega) = \int_{-\infty}^{\infty} \tilde{E}_{SHG}(\omega - \omega')\tilde{E}(\omega')\, d\omega'. \tag{2}$$

If we change variable $\omega' = \frac{\omega+\omega''}{2}$, we find that Eqn. 2 can be rewritten as:

$$\tilde{E}_{SHG}(\omega) = \int_{-\infty}^{+\infty} \tilde{E}\left(\frac{\omega - \omega''}{2}\right) \tilde{E}\left(\frac{\omega + \omega''}{2}\right) d\omega''. \tag{3}$$

Equation 3 highlights better the fact that every frequency component ω in the $\tilde{E}_{SHG}$ spectrum is the result of the combination of a (infinite) number of frequency couples $\omega_{left} = \frac{\omega-\omega''}{2}$ and $\omega_{right} = \frac{\omega+\omega''}{2}$ whose center is located in ω.

2.2 The Role of the Phase

The last step to understand our experiment of SHG optimization consists in recognizing the importance of the phase of the input pulse for the intensity of the SHG pulse in output. In our mathematical approach to SHG, the phase terms have been hidden until now in the functions $E(\omega)$.

In the next equations we pull the phases out.

$$\tilde{E}_{SHG}(\omega) = \int_{-\infty}^{+\infty} |\tilde{E}\left(\frac{\omega - \omega''}{2}\right)| e^{i\phi\left(\frac{\omega-\omega''}{2}\right)} |\tilde{E}\left(\frac{\omega + \omega''}{2}\right)| e^{i\phi\left(\frac{\omega+\omega''}{2}\right)}\, d\omega'' \tag{4}$$

$$\phi(\omega) = \phi_0(\omega) + \phi_{sh}(\omega) \tag{5}$$

In Eqn. 5, now, the phase profile of the input pulse is explicitly written as the sum of two conceptually very different terms, the natural term $\phi_0(\omega)$ and a computer-controlled term $\phi_{sh}(\omega)$ introduced by means of our pulse shaper. The effect of the phase terms on the integral can be immediately understood if we regard the amplitude of the integrand as the length of a vector (length dependent on ω) and the phase as the direction of such vector (also dependent on ω). The best scenario that optimizes the output of the integral is then clearly when all the vectors are pointing in the same direction. This case translates in all the vectors having the same phase (modulus 2π), which is tantamount to having a constant phase profile $\phi(\omega)$ in the input pulses. In an ideal pulse the term $\phi_0(\omega)$ should be constant as all frequencies should have the same phase when coming out of the laser. Unfortunately, in real life applications, as the pulse travels through optical tools like amplifiers, non-linear crystals for frequency tuning, simple lenses, beam splitters etc., it picks up a significant frequency-dependent phase profile that severely reduces the amount of SHG light coming out of our monitoring black box.

2.3 Fitness and Free Parameters of the Search

We define the fitness parameter for our evolutionary search as the total intensity of the SHG light generated in the crystal. In mathematical terms this would be:

$$F = \int_{-\infty}^{+\infty} |\tilde{E}_{SHG}(\omega)|^2 d\omega \tag{6}$$

This function needs to be optimized by adding an adequate phase profile $\phi_{sh}(\omega)$ with our computer-controlled pulse shaper. In both our experiments and simulations, this additional phase function $\phi_{sh}(\omega)$ is described by 320 parameters free of varying within a $[0, 2\pi]$ range. The result of optimizing a fitness function F , as defined above, is to make the total phase function $\phi(\omega) = \phi_0(\omega) + \phi_{sh}(\omega)$ a constant (modulus 2π), which, in return, yields the shortest possible pulse for the given bandwidth (transform-limited (TL) pulse).

2.4 Properties of the Fitness Function, Search Landscape and Real-Life Noise

In this special case it is easy to prove that, modulus 2π, there is only one optimal solution with no local solutions. This lack of false directions or secondary solutions would seem to make the problem easy to solve. However, as we will show both with simulations and laboratory data, the most commonly used evolutionary algorithms do not always converge to the optimal, constant phase, solution. This behavior is expected in the presence of noise in the fitness function F. In the following sections we will show a comparison between CMA and DR2 algorithms and we will try to find their best settings to handle a (realistic) noise level. After estimating the noise level of the fitness function F in the laboratory, we modeled the SHG process based on Eqns. 3 and 6 and we introduced a random noise on top of the F function according to Eqn. 7 and 8. Due to software requirements the original problem had to be turned from a maximization problem to a minimization problem. Equation 8 also expresses the fact that F^{-1} was used as a fitness function[1].

$$n_c \sim \frac{5}{29} \cdot \mathcal{U}(0,1), \quad n_v \sim \frac{0.05}{29} \cdot \mathcal{U}(0,1) \tag{7}$$

$$F_{\text{opt}} = F^{-1} \cdot (1 + n_c) + n_v \tag{8}$$

where $\mathcal{U}(0,1)$ denotes uniformally distributed random numbers from the interval $[0,1)$.

3 Evolution Strategies

For this study two algorithms from the class of derandomized Evolution Strategies (ES) were applied. The defining feature of derandomized Evolution Strategies is a deterministic adaptation mechanism that derives new step size information from old step sizes and the magnitudes of successful mutation events.

The two variants applied here differ mainly in the distribution information that is adapted and used for creation of new offspring individuals: The simple Derandomized Adaptation (DR2) as suggested in [5] basically adapts the n variances of an n-dimensional Gaussian distribution while the more advanced Covariance Matrix Adaptation (CMA) as in [6] uses the $n(n+1)/2$ variances and covariances. Though the details of the two adaptation schemes differ they are built on identical concepts. The core idea of derandomized step size adaptation is to compare the size of actual realizations of mutation events ($|z|$) to the expected value of the originally proposed distribution ($E[|z|]$). Let σ denote a parameter of the mutation distribution, and let $\sigma' = A(\sigma, z, \theta)$ be the adapted parameter derived from the old σ, the successful mutation event z and some internal parameters θ. Then a derandomized adaptation function A (i.e. here either DR2 or CMA) basically implements a deterministic method that ensures the following conditions:

$$E[|z|] \geq |z| \longrightarrow \sigma \geq \sigma' = A(\sigma, z, \theta) \tag{9}$$

$$E[|z|] < |z| \longrightarrow \sigma < \sigma' = A(\sigma, z, \theta) \tag{10}$$

That means that whenever a *successful* mutation is observed whose magnitude is smaller than what would be expected from the current step size the step size will be reduced and vice versa.

To stabilize the adaptation process both DR2 and CMA use the concept of *cumulation path* which means that the random variable z used for adaptation is not directly the latest single successful mutation event but a weighted sum of successful mutations stretching over multiple generations. Therefore z needs to be updated with successful mutation events m by

$$z' = cm + (1 - c)z \tag{11}$$

where $c \in [0,1]$ is an algorithm specific constant[2].

[1] For future studies we would rather choose $-F$, but since F is strictly positive using F^{-1} is a feasible approach.

[2] c is usually assumed to depend on problem dimensionality.

The implementations of DR2 and CMA used for this study also share the use of weighted recombination as introduced in [6]. Although the DR2 algorithm was originally suggested as a $(1, \lambda)$ strategy without recombination the adaptation scheme is sufficiently general to allow application of weighted recombination, i.e. new individuals are created by recombination

$$x' = x_{1:\mu} \cdot w \tag{12}$$

where $x_{1:\mu}$ denotes the matrix of the μ best column vectors of design variables and w is a weights vector with

$$w_i = \frac{\log(\mu + 1) - \log(i)}{\sum w_i}, \quad 1 \leq i \leq \mu. \tag{13}$$

After recombination individuals are mutated by

$$m \sim \mathcal{N}(0, \Sigma) \tag{14}$$
$$x'' = x' + m \tag{15}$$

where $m \sim \mathcal{N}(0, \Sigma)$ denotes sampling m from a Gaussian distribution with expectation 0 and covariance matrix Σ.

Handling Box Constraints. As already mentioned in Section 2.4, optimal solutions to the SHG problem are unique modulus 2π. This introduces a certain difficulty for derandomized Evolution Strategies, since a successful mutation event might appear to be big on first sight, but reduce to a small change modulus 2π. In this situation the ES will falsely consider a too big mutation as a successful event. To avoid this kind of situation the Gaussian mutation operator was slightly extended to ensure $0 \leq x_i \leq 2\pi$ for all optimization variables x_i.

The basic idea of a mutation is a small change to the original data, such that evolution proceeds as a series of minor changes rather than a big singular, dramatic change. This is one of the main reasons for using Gaussian distributions as mutation operators in Evolution Strategies [7]. Since Gaussian distributions are at least theoretically unbounded the usual mutation operators as defined in Eqns. 14 and 15 cannot assure upper or lower bounds on the outcoming, mutated optimization variable x. Assume box constraints of the form $lb \leq x \leq ub$, where lb and ub denote lower and upper bounds on x, respectively. We propose a repair method working on top of the usual Gaussian mutation, where the following conditions should hold:

1. Whenever an infeasible Gaussian mutation event is detected, the new mutation operator should yield a mutation that is smaller or equal to the Gaussian mutation event. By doing so, we assume that the Gaussian mutation event always classifies as a small change, and ensuring that the new mutation event is even smaller and therefore qualifies as a mutation like small change.
2. Whenever an infeasible Gaussian mutation event is detected, the new mutation operator should create a mutation in the same direction, i.e. if the

Gaussian mutation event increased x then the new mutation event should also do that and vice versa. This is different from a simple resampling approach, where Gaussian mutations are performed repeatedly until a feasible value is generated. Resampling biases resulting values away from the interval bounds, which is undesirable.

To implement this we suggest to mutate Gaussian first, and whenever that yields infeasible values, to replace the Gaussian sampling by a uniform sampling in the interval given by the original x and the respective bound violated by the Gaussian mutation event. E.g. if the Gaussian mutation violates lb the mutation samples uniformly in the interval $[lb, x]$. Doing so of course assumes that x is always feasible, but that should not be problematic by initializing feasibly. This approach is formalized in the following equations:

$$m_0 \sim \mathcal{N}(0,1) \tag{16}$$

$$x' = \begin{cases} x + s \cdot m_0 & \text{if} \quad lb \leq x + s \cdot m_0 \leq ub \\ x + \mathcal{U}(0,1) \cdot (x - lb) & \text{if} \quad x + s \cdot m_0 < lb \\ x - \mathcal{U}(0,1) \cdot (ub - x) & \text{if} \quad x + s \cdot m_0 > ub \end{cases} \tag{17}$$

$$m = (x' - x) \cdot s^{-1} \tag{18}$$

where $\mathcal{U}(0,1)$ denotes sampling uniformally from $[0,1)$. Equation 18 is required for path accumulation (Eqn. 11) during derandomized adaptation. For CMA and DR2 the repair mechanism simply pretends that the mutation event was originally created by Gaussian mutation, such that search distributions can be adapted appropriately.

Derandomized Adaptation (DR2). The derandomized adaptation approach DR2 used in this study was first introduced in [5]. For an n-dimensional optimization problem it uses $n+1$ strategy or step size parameters, where one (σ_0) defines the global width of the Gaussian search distribution and the remaining ones (σ_i, $1 \leq i \leq n$) define the variances of n independent Gaussian distributions used to mutate the corresponding n optimization variables. Equation 19 shows the working principal of the adaptation scheme using the accumulated path z as defined in Eqn. 11.

$$\sigma_0' = \sigma_0 \cdot \exp\left(c_1 \left(\frac{|z|}{c_2} - E[|z|]\right)\right), \quad \sigma_i' = \sigma_i \cdot \exp\left(c_3 \left(\frac{|z_i|}{c_4} - E[|z_i|]\right)\right) \tag{19}$$

where $E[|z|]$ and $E[|z_i|]$ denote approximations to the expected value of the length of the z vector and the absolute value of its components, respectively, and $c_{1,\ldots,4}$ are normalization constants. For further details sees [5,8][3].

Covariance Matrix Adaptation (CMA). The following equations give a bird's eye view of the main aspects of the Covariance Matrix Adaptation scheme.

[3] In [5] the adaptation of the local step size σ_i in Eqn. 19 has a slightly different form, yet in our experience both versions perform equally well.

Further details, especially on the vastly simplified setting of the various normalization constants, can be found in [6]. Consider Σ the covariance matrix to be adapted and B the corresponding matrix of eigenvectors and D the corresponding eigenvalues, and let $\mathcal{N}(\mu, \Sigma)$ denote a Gaussian random variable with expectation μ and covariance matrix Σ. The eigenvectors and eigenvalues are used for rotation and scaling such that a vector of independently $\mathcal{N}(0, 1)$ distributed Gaussian random variables can be turned into a vector of $\mathcal{N}(0, \Sigma)$ distributed Gaussian random variables and vice versa. Assume further that $m_{0,1}$ is the vector of $\mathcal{N}(0, 1)$ distributed Gaussian random variables that happened to be a successful mutation[4].

CMA first of all adapts a global scalar step size σ_0 that determines the overall width of the next search distribution. To do so CMA compares the length of the accumulated successful mutation information vector to its expected value $E[|z'_{\sigma_0}|]$.

$$z'_{\sigma_0} = c_1 z_{\sigma_0} + c_2 B m_{0,1}, \quad \sigma'_0 = \sigma_0 \exp\left(c_3 \left(\frac{|z'_{\sigma_0}|}{E[|z'_{\sigma_0}|]} - 1\right)\right) \tag{20}$$

The actual covariance matrix is adapted after accumulation by adding a rank one matrix yield from multiplying the accumulated path information by itself.

$$p'_{\Sigma} = c_4 p_{\Sigma} + c_5 B D m_{0,1}, \quad \Sigma' = c_6 \Sigma + c_7 p'_{\Sigma} p'^{T}_{\Sigma} \tag{21}$$

4 SHG Optimization on Simulations

The simulation study considers the following parameters of the optimization algorithms:

Adaptation Mechanism. The core method implementing self adaptiveness of the Evolution Strategy.

Number of Parents. The μ parameter in a (μ, λ) Evolution Strategy

Number of Offspring. The λ parameter in a (μ, λ) Evolution Strategy

Initial Step size. The magnitude σ_0 of the initial step sizes, i.e. width of the search distribution.

These parameters are probably the most frequently used parameters to tune Evolution Strategies. Although by now there are useful suggestions available for automatically setting population sizes ([6]) these heuristics are not necessarily applicable for noisy functions. It has been observed before ([9]) that population sizes are highly influential for noisy fitness functions. From early on the numbers of parents and offspring have not been considered as independent parameters (see e.g. [7]), it has rather been assumed that the ratio of μ and λ is a key parameter. Therefore, in this simulation study, combinations of a number of parents μ and the parent to offspring numbers ratio μ/λ were tested rather than

[4] In accordance with Eqns. 14 and 15 $m = BDm_{0,1}$ holds true.

combinations of μ and λ directly. The following parameter values were combined for the simulation study:

$$\mu \in \{1, 5, 10, 20\}, \quad \frac{\mu}{\lambda} \in \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{10}\right\}, \quad \sigma_0 \in \{0.2, 0.1, 0.01\}$$

Table 1 summarizes some of the results achieved with simulation runs. First of all the table contains the best parameter sets detected. As quality measure the median of the best fitness value recorded in 15 repetitions of the same optimization run was used. Table 1 also shows the mean, minimum, maximum, and standard deviation of the 15 best fitness values associated with each parameter set. Additionally some of the worse results are listed for reference. Due to limited space not all results can be given. It is obvious from the table that with the exception of the last row there are no trials listed using the smallest initial step size of $\sigma_0 = 0.01$. It turned out that almost all optimizations using this small step size achieved only very bad results. This can be seen in Table 2, where performance metrics were aggregated over all results with identical initial step size. The aggregation averages over very differently performing strategies,

Table 1. Performance of different parameter settings after 5000 evaluations showing median (**Median**), mean (**Mean**), minimum (**Min**), maximum (**Max**), and standard deviation (**Std**) of the best fitness values of 15 repeated optimization runs. The **Rk** column indicates the position of the row if ordered by the respective value. Rows marked with $\star$ were tested in the laboratory.

Alg	μ	λ	σ_0	**Median**	**Rk**	**Min**	**Rk**	**Max**	**Rk**	**Mean**	**Rk**	**Std**	**Rk**
$\star$ CMA	20	40	0.10	2.471	1	2.207	2	3.117	2	2.525	1	0.229	4
$\star$ DR2	20	80	0.20	2.703	2	2.073	1	3.932	9	2.834	3	0.532	13
CMA	20	40	0.20	2.716	3	2.441	4	3.114	1	2.747	2	0.210	3
CMA	10	40	0.10	2.876	4	2.515	6	3.928	8	3.024	5	0.462	11
CMA	10	20	0.10	2.937	5	2.453	5	3.291	3	2.981	4	0.237	5
DR2	10	100	0.20	2.938	6	2.432	3	6.258	22	3.248	8	0.988	27
CMA	10	20	0.20	3.018	7	2.635	7	3.759	7	3.047	6	0.355	8
DR2	10	40	0.20	3.125	8	2.740	8	3.613	4	3.192	7	0.271	7
CMA	20	80	0.10	3.336	9	2.989	13	3.739	6	3.317	9	0.188	1
CMA	5	20	0.10	3.340	10	2.797	9	5.156	15	3.543	11	0.588	15
CMA	10	40	0.20	3.415	11	3.165	17	3.719	5	3.446	10	0.193	2
DR2	20	40	0.20	3.607	12	3.093	15	6.656	23	4.014	16	1.130	30
$\star$ DR2	5	50	0.10	4.043	18	2.963	12	5.537	16	3.969	15	0.733	21
$\star$ CMA	5	10	0.10	4.256	21	3.835	24	4.970	12	4.275	20	0.266	6
$\star$ DR2	20	40	0.10	7.256	32	5.025	30	12.497	31	7.477	32	2.010	42
DR2	1	10	0.20	9.886	35	7.269	35	12.513	32	9.993	35	1.669	40
DR2	1	10	0.10	10.308	36	7.254	34	13.978	35	10.168	36	1.974	41
CMA	1	10	0.10	11.282	37	8.758	39	16.961	38	12.198	37	2.336	44
CMA	1	10	0.20	13.553	38	9.254	40	15.974	37	13.032	38	2.195	43
$\star$ CMA	20	40	0.01	16.967	42	14.955	45	18.409	40	16.747	40	1.030	28

Table 2. Performance of initial stepsize values (σ_0) (Column headings as in Tab. 1)

σ_0	Median	Rank	Min	Rank	Max	Rk	Mean	Rank	Std	Rank
0.10	5.16	1	2.21	2	74.46	2	15.08	2	20.67	3
0.20	5.52	2	2.07	1	72.40	1	14.75	1	20.13	1
0.01	26.97	3	13.15	3	83.11	3	35.84	3	20.28	2

as can be seen from the respective minimum, maximum, and standard deviation columns. Still there seems to be a dramatic difference in the mean and median values for $\sigma_0 \in \{0.1, 0.2\}$ on the one hand and $\sigma_0 = 0.01$ on the other hand.

Coming back to the results of Table 1 significant differences between the SHG Problem with and without added noise can be seen. While in [2] the (1,10)-ES using DR2 adaptation outperformed a (1,10)-ES using CMA adaptation, which itself outperformed an (8,17)-ES using CMA, the addition of noise clearly discards the (1,10)-Strategies, which finished with ranks 35-38. Interestingly though, within the set of (1,10)-Strategies DR2 still performs better than CMA, so one might conclude that DR2 manages the basic characteristics of the SHG problem more successful than CMA, without noise changing this big picture for a (1,10)-ES.

From Table 1, no clear answer to the question of which adaptation scheme to prefer can be deduced. Although the table contains more well performing results based on CMA than on DR2, the second best performing strategy uses DR2. The fact that there are more rows filled with CMA results than with DR2 results basically means that CMA is less sensitive to its parameter settings, i.e. it is less critical to determine good population sizes for CMA than it is for DR2. In general, a certain improved reliability turns out to be an advantage of CMA that is also visible in the standard deviation column of Table 1. Apart from less sensitive parameter settings CMA's higher reliablity can also be seen in smaller variance of optimization results. With the exception of the (10,40)-ES using DR2 all CMA results have considerably less spread than their DR2 counterparts.

5 SHG Optimization in the Laboratory

One of the main motivations for this study apart from studying the influence of laboratory noise on optimization performance was to assess the transfer of knowledge gained from simulated optimization runs to real world laboratory experimentation. This is largely because laboratory time is considerably more expensive and restricted than computation time. A single optimization run in the laboratory takes up to 30 minutes, neglecting considerable time to set up the experiment correctly. Due to limited laboratory infrastructure optimization runs cannot be parallelized which is most easily done with simulations. So in the light of limited experimentation time available for this study we decided to trade widespread parameter testing for statistical significance, and ran multiple repetitions of a limited number of paremeter settings in the laboratory. In order to not bias results by selecting only the best parameter settings for both CMA and

Table 3. Performance of different parameter settings in laboratory experimentation. $\bar{L}_.$ denote average performance of laboratory experiments, $\bar{S}_.$ denotes average performance of simulation experiments. $\{\bar{S}, \bar{L}\}^\star$ denote the best average performance achieved with the respective adaptation scheme.

Alg	μ	λ	σ_0	$\bar{L}$	$\bar{S}$	$\bar{L}/\bar{L}^\star$	$\bar{S}/\bar{S}^\star$
CMA	20	40	0.1	70.42	2.47	1.00	1.00
CMA	5	10	0.1	142.86	4.26	2.03	1.72
CMA	20	40	0.01	500.00	16.97	7.10	6.87
DR2	20	80	0.2	156.25	2.70	1.00	1.00
DR2	5	50	0.1	222.22	4.04	1.42	1.50
DR2	20	40	0.1	250.00	7.26	1.60	2.68

DR2, the best, a mediocre, and a rather bad setting as found in the simulations were tested. In Table 1 these settings are marked with a $\star$.

Table 3 summarizes the results achieved in the laboratory together with the respective simulation experiments. For technical reasons the numerical values for simulations and experiments in the table do not match exactly. To extract more meaningful information on the link between simulations and experiments, we separately normalized both type of results to their best value. Comparing these ratios in the last two columns of Table 3 we can see how the results of simulations and experiments are strikingly similar.

In contrast to the simulation results, though, CMA yields better results than DR2, except when used with the much too small step size of $\sigma_0 = 0.01$. The issue of too small initial step size is likely to be more problematic in the laboratory than it was for simulations, since there are potentially additional sources of noise acting on the inputs that were not covered during simulations. Too small initial step sizes may easily lead to variable changes on the same scale as the noise, which makes optimization practically impossible.

6 Conclusions and Outlook

In this study we compared two different adaptation schemes of Evolution Strategies, namely DR2 and CMA, together with variations of their parameter settings in a prototype laboratory situation. We found significant differences between simulations with and without added noise. Nonetheless, the results from the noisy simulations transferred nicely to the laboratory, as verified by a number of experiments. Although both variants of the Evolution Strategy in principle produced good results both in simulations and in the laboratory, CMA turned out to be the more reliable algorithm. Since reliability is of fundamental importance in the laboratory, CMA is considered the method of choice for future experimentation.

The results of this study suggest that Second Harmonic Generation can be used as a prototype application for online control of laboratory experiments by Evolutionary Algorithms. The obvious next steps are improvement of algorithm performance for the targeted noisy fitness functions on the one hand, using SHG as a primary test function. On the other hand, more challenging applications can be addressed with the knowledge gained by the SHG experiments. Especially laboratory optimization experiments for which today no feasible computer simulations are available, may be tackled using the methods and insights available from continuing improvement of the algorithms.

Acknowledgements

This work is part of the research program of the *Stichting voor Fundamenteel Onderzoek der Materie (FOM)*, which is financially supported by the *Nederlandse organisatie voor Wetenschappelijk Onderzoek (NWO).*

References

1. Shir, O., Siedschlag, C., Bäck, T., Vrakking, M.: Evolutionary algorithms in the optimization of dynamic molecular alignment. In: G.G.Y., et al, (ed): IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 2912–2919. IEEE Press Los, Alamitos (2006)
2. Shir, O.M., Bäck, T.: The second harmonic generation case-study as a gateway for ES to quantum control problems. In: Lipson, H. (ed.) Genetic and Evolutionary Computation Conference, ACM, New York (2007)
3. Weiner, A.M.: Femtosecond pulse shaping using spatial light modulators. Review of Scientific Instruments 71(5), 1929–1960 (2000)
4. Trebino, R.: Frequency-Resolved Optical Gating: The Measurement of Ultrashort Laser Pulses. Kluwer Academic, Dordrecht, The Netherlands (2002)
5. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size adaptation based on non-local use of selection information. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 189–198. Springer, Heidelberg (1996)
6. Hansen, N., Ostermeier, A.: Completley derandomized selfadaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
7. Schwefel, H.P.: Evolution and Optimum Seeking. Wiley, New York (1995)
8. Hansen, N., Ostermeier, A., Gawelczyk, A.: On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In: Eshelman, L. (ed.) Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco CA (1995)
9. Beyer, H.G.: Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. Computer Methods in Applied Mechanics and Engineering 186(2–4), 239–267 (2000)

Fully Three-Dimensional Tomographic Evolutionary Reconstruction in Nuclear Medicine

Aurélie Bousquet[1], Jean Louchet[1], and Jean-Marie Rocchisani[1,2]

[1]INRIA Rocquencourt/ COMPLEX, Domaine de Voluceau, B.P. 105. 78153 Le Chesnay, France
[2] Paris XIII University, UFR SMBH & Avicenne hospital, 74 rue Marcel Cachin. 930013 Bobigny, France
aurelie.bousquet@inrets.fr, jean.louchet@inria.fr, jean-marie.rocchisani@{avc.aphp.fr, inria.fr}

Abstract. 3-D reconstruction in Nuclear Medicine imaging using complete Monte-Carlo simulation of trajectories usually requires high computing power. We are currently developing a Parisian Evolution Strategy in order to reduce the computing cost of reconstruction without degrading the quality of results. Our approach derives from the Fly algorithm which proved successful on real-time stereo image sequence processing. Flies are considered here as photon emitters. We developed the marginal fitness technique to calculate the fitness function, an approach usable in Parisian Evolution whenever each individual's fitness cannot be calculated independently of the rest of the population.

Keywords. Computer Tomography, Emission Tomography, Artificial Evolution, Parisian Evolution, FlyAlgorithm, Compton scattering, Nuclear Medicine.

1 Introduction

1.1 Nuclear Medicine

In Nuclear Medicine diagnosis, radioactive substances are administered to patients using a tracer molecule containing a radioactive marker. The distribution of radioactivity in the body is then estimated from the radiation detected by gamma cameras. In order to get an accurate estimation, a three-dimensional tomography is built from two-dimensional scintigraphic images.

Some artefacts due to scattering and absorption are then to be corrected. Existing analytical and statistical methods are costly and require heavy computing. The main variants of Nuclear Imaging are SPECT (Single Photon Emission Computed Tomography) and PET (Positon Emission Tomography). Radioactive tracers are photons or positons emitters. Compared to other tomography techniques as X-ray scanning or Magnetic Resonance Imaging, Nuclear Imaging brings useful information on biological and metabolic function. The marker most widely used in SPECT is Technetium 99m (99mTc), with a half-life of about 6 hours and emitting photons at an energy level of 140keV, which is well adapted to current gamma-camera technology. In planar mode, the gamma camera is fixed and collects a plain two-dimensional projection of the radioactive tracer concentration. In tomographic mode, the gamma camera rotates around the patient. A gamma camera can also be used in static or dynamic mode, allowing to monitor how the radioactive tracer concentration evolves in the body. The main limitations of this technology are:

– sensor performance (resolution, sensitivity),
– physical effects (absorption, scattering, noise),
– motion of patient (long exposure times),
– accuracy of reconstruction algorithms.

1.2 The Compton Effect and Its Consequences

Rayleigh effect occurs when a photon meets an atom without disturbing its electronic structure. The photon then gets deviated but keeps its original energy. With high energy photons, this effect is negligible except in the gamma camera crystal itself. In photoelectric interaction, our photon is completely absorbed by an atom which then emits a fluorescence photon to carry the excess energy. This is the basic process of photon detection in the gamma camera.

The Compton effect is by far the dominant perturbation during the transit of high energy photons from the tracer through the body. Both absorption and scattering induce important effects on image quality, as they vary with the nature and thickness of the part of the body involved. Compton scattering with important energy losses will have a larger than average deviation angle.

The photons with an energy level close to the initial level have a small probability of having been deviated. However, the energy resolution of gamma cameras is not sufficient to always ascertain whether a photon has been deviated or not. In order to correct attenuation, it is possible to use a X-ray CT scanner image which gives an accurate representation of the attenuation map in the body; however, a uniform attenuation map may be used when the organ is homogeneous enough. On the other hand, scattering is more difficult to deal with. The main algorithm families [1, 2, 7] are:

– subtraction algorithms using energy windows to filter primary electrons,
– deconvolution algorithms which consider scattering is a uniform process,
– recombination algorithms (based *e.g.* on principal component analysis).

Our long-term aim is to correct Compton scattering using Evolutionary Computation in order to get faster results with a level of quality similar to present high cost algorithms. The first step presented in this paper is the validation of an evolutionary 3-D reconstruction algorithm with a simplified propagation model, allowing future replacement of the propagation module with a more accurate model [6] including the modelling of Rayleigh and Compton effects. Contrary to standard reconstruction algorithms: Filtered back projection (FBP) and Ordered Subset Expectation Maximisation (OSEM) where the problem is split into parallel 2-D slices and not all sensor data are used, our method pertains to the family of "fully 3-D" reconstruction methods.

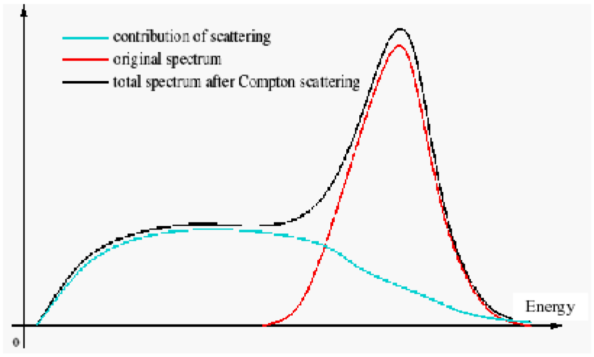


Fig. 1. Detection spectrum of gamma photons

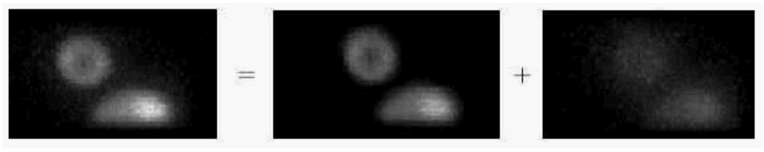


Fig. 2. Contributions of primary and scattered photons

2 A Parisian Evolutionary Approach

2.1 The Classical Fly Algorithm

The original Fly Algorithm [5] is a 3-D evolutionary reconstruction method based on the Parisian Approach. Each individual ("fly") represents a point of space and the whole population of flies is the representation of the object detected. It uses a fitness function based on the consistency of grey level properties of the projections of the fly on the images taken by each camera. It has been first used on stereovision [4].

Here, the semantics of the fly is enriched as we will now consider the fly is a photon emitter. Again, the algorithm evolves a population of flies which eventually converge to the three dimensional shape to be detected. While this approach has been validated in its principles, computation costs were still high due to the complexity of physically modelling random photon trajectories, and the reconstruction results were not quite up to the vquality expected or obtained through more classical methods. Following this, we developed an innovative evaluation function based on a specific approach to fitness calculation, called "marginal fitness", giving encouraging results on both simplified synthetic data and real scintigraphic images.

2.2 Monte-Carlo Simulation

Monte-Carlo simulation [3] is well adapted to nuclear medicine with its particle emission, propagation and detection random processes. Each photon trajectory is processed separately. The photon is propagated through space cells where it can be absorbed or scattered conforming to suitable random depending on the local environment. Each photon thus carries his own including which fly was its source. In a first approach, we considered the patient's body as a homogeneous cylinder. A later, more refined approach consists of using an absorption map in function of the material involved.

2.3 Building a Fitness Function

Radioactive tracers are only present in the central search zone, which contains the patient's body. The "screens" are the different positions of the gamma camera crystals. A fly is defined as a photon emitter and is described by its coordinates (x, y, z).

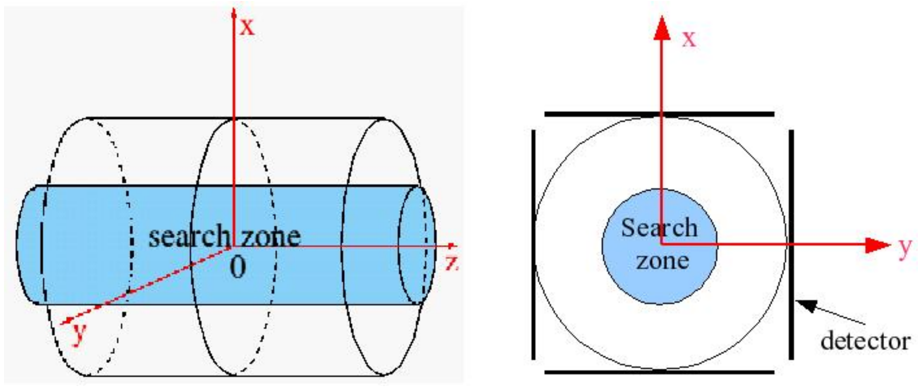


Fig. 3. Modeling the tomographic system: lateral view (left), axial view (rigth)

We first validated the principles of an evolutionary approach using a simplified, homogeneous model of the body, and a bonus-based fitness function: each simulated photon that reached a detector cell had a contribution to the fitness of its originator fly proportional to the actual number of photons received by this cell. The high number of Monte-Carlo simulations led to unrealistic processing time. In a second approach, in order to speed up processing, we defined a number of archetypal flies characterised by their distance from the detectors, while the search space was still considered homogeneous. Monte-Carlo simulation was performed on each archetypal fly and the results stored to be used as a lookup table in the evolutionary process. Evolution was then run calculating fitness values based on pre-calculated Monte-Carlo results, leading to less than half the original computation time. However this approach cannot take into account the heterogeneity of matter and it lacks precision, so that we had to concentrate on developing a fitness function that be fast, accurate and open to heterogeneity.

The overall process is summarized by the following diagram:

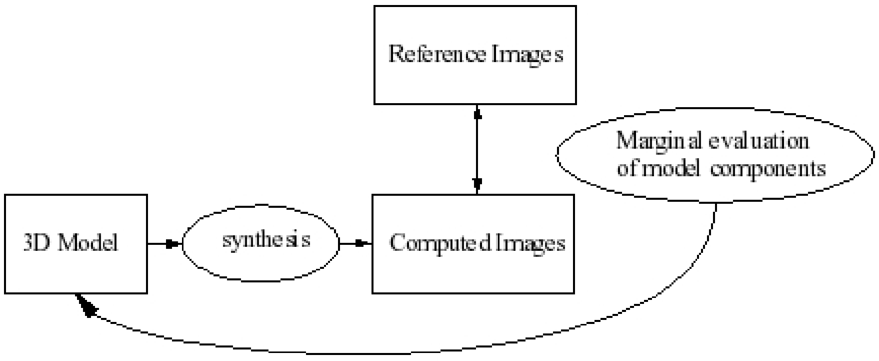


Fig. 4. The 3D Evolutionary reconstruction

Bonus Fitness. While reducing the number of photons emitted by each fly to only 4 photons initially oriented orthogonally to the detectors gave a substantial improvement in calculation time, experience showed an important backside of bonus-based fitness: in presence of several bright objects, the flies will tend to accumulate on the brightest or biggest object at the expense of the other ones. This is illustrated on the following images: the 3-D scene consists of two cubes of different size and brightness; the image on the left shows what an ideal reconstruction algorithm should have given, and the right image what it actually gave using bonus fitness. The same behaviour was found on all similar data.

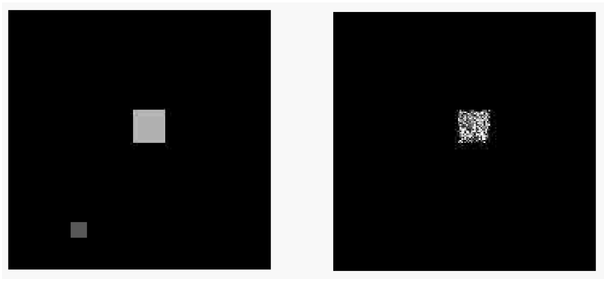


Fig. 5. Bonus fitness: loss of smaller objects (left: ideal image; right: actual reconstruction, side view)

Marginal Fitness. Rather than evaluating a fly independently of its context, we introduced marginal evaluation by defining the fitness of a particular fly as its contribution (positive or negative) to the whole population's Fitness:

$$\text{fitness}(i) = \text{Fitness}(\text{population} - \{ i\}) - \text{Fitness}(\text{population}) \qquad (1)$$

To this end, the Fitness of a given population is given by the likeness of the projection images generated through Monte-Carlo simulation, with the actual images given by the sensors. As the grey level of the synthesised images depends on the number of flies, a normalization factor must be introduced in order to compare the natural and synthetic projections.

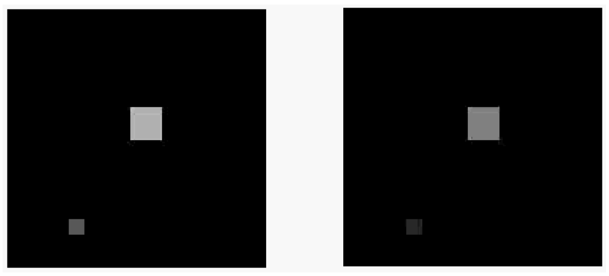


Fig. 6. Marginal fitness: better detection of smaller objects (left: ideal image; right: actual reconstruction), side view

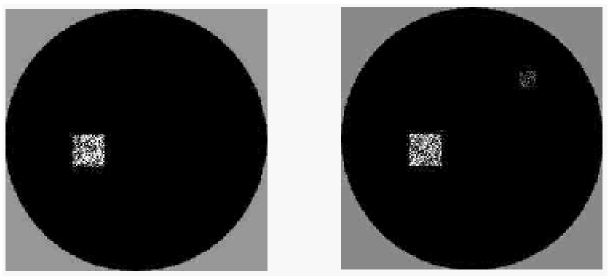


Fig. 7. Top views of results with bonus (left) and marginal (right) fitness functions

Rotating Screens. Rotating screens are often used in SPECT imaging, with up to 128 screen positions. In order to exploit all the data while keeping memory requirements down, only 4 screens are used for fitness calculation and periodically rotated.

3 Results

The following results have been calculated from real SPECT images using the algorithm described above. In the current state of research, we did not include detailed Monte-Carlo simulation of absorption and Compton scattering into these experiments which only demonstrate the validity of the fly-based reconstruction algorithm. Integration of a fast Monte-Carlo simulation into the algorithm will be necessary to obtain high quality results.

3.1 Example 1

The objects are three cylinders with different brightness and diameter. The parameters are given in table 1.

As this is usual with Parisian Evolution, high mutation rates are used while crossover is not always essential to performance.

Table 1. Parameters used in synthetic data reconstruction

projection image size	128*128
number of flies	266000
number of screens used at each generation	4
total number of screens	128
screens rotated every	5 generations
number of generations	1000
probability of mutation	50,00%
decrease of pm per generation	5,00%
probability of crossover	0,00%
mutation factor	1cm

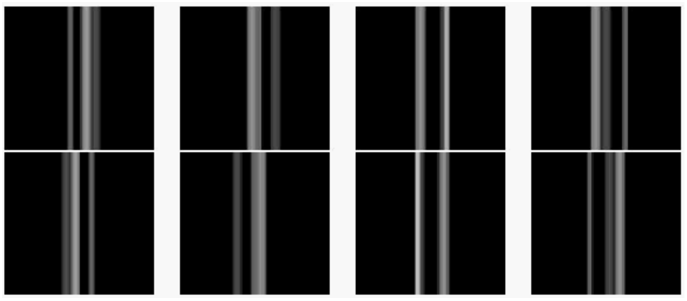


Fig. 8. Synthesised projections of the object, viewed under different angles: n / 4, n □ [0,7]}

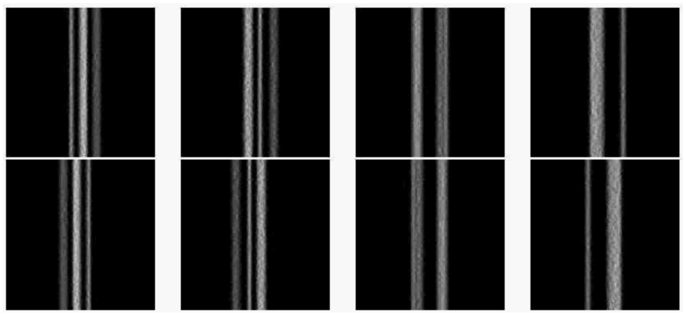


Fig. 9. Side views of the 3-D reconstructed object (flies) under the same angles

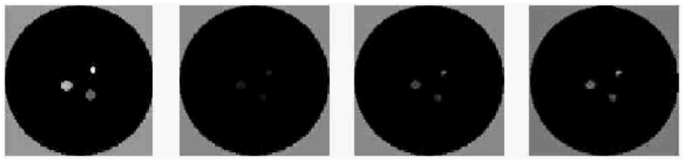


Fig. 10. Views of original object (left) and reconstructions (slices of 5, 10 and 20 pixels)

3.2 Example 2

Here, there are 3 nested objects with different brightness. The algorithm parameters are the same as in the previous example.

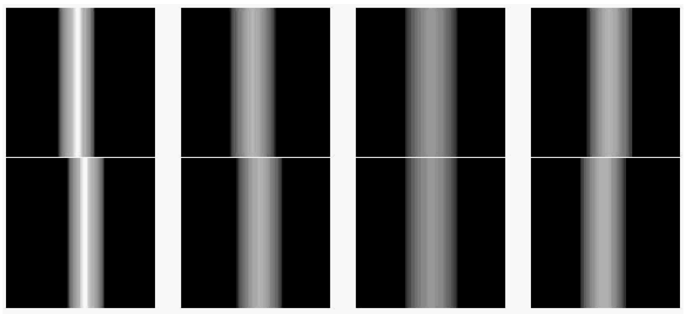


Fig. 11. Synthesised projections of the object, viewed under different angles

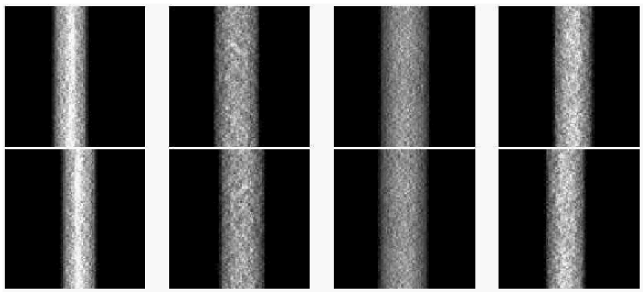


Fig. 12. Side views of the 3-D reconstructed object (flies) under the same angles

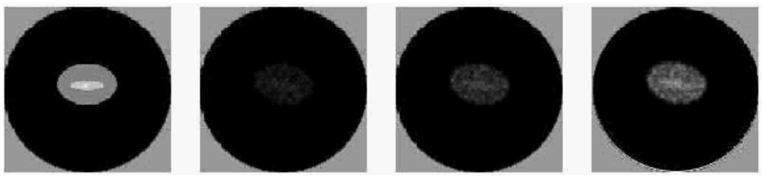


Fig. 13. Top views of the 3-D reconstructed object: original object (left) and its reconstructions (slices of 5, 10 and 20 pixels)

3.3 Example 3: Real Data

We tested the algorithm on actual images of bone scintigraphy, with the following parameters:

Table 2. Parameters used in bone scintigraphy reconstruction

projection image size	128*128
number of flies	1017500
number of screens used at each generation	4
total number of screens	64
screens rotated every	5 generations
number of generations	1000
probability of mutation	50,00%
decrease of pm per generation	5,00%
probability of crossover	0,00%
mutation factor	1cm

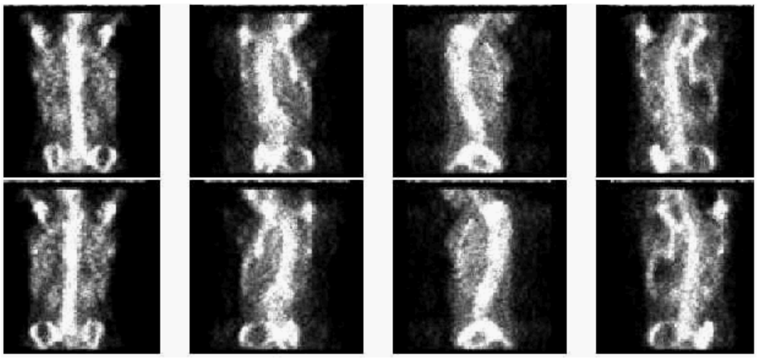


Fig. 14. Original projection acquired around the patient

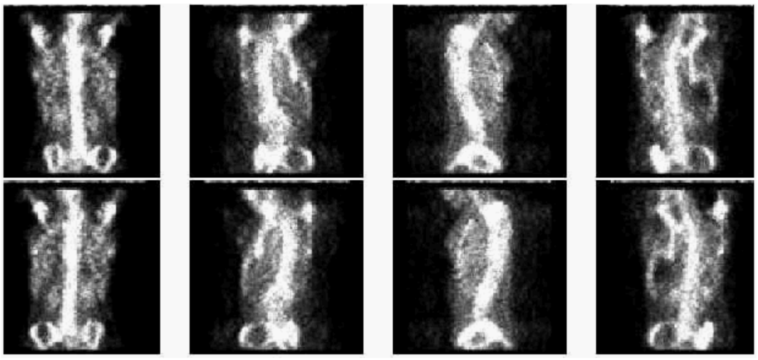


Fig. 15. Side views of the 3-D reconstructed object (flies)

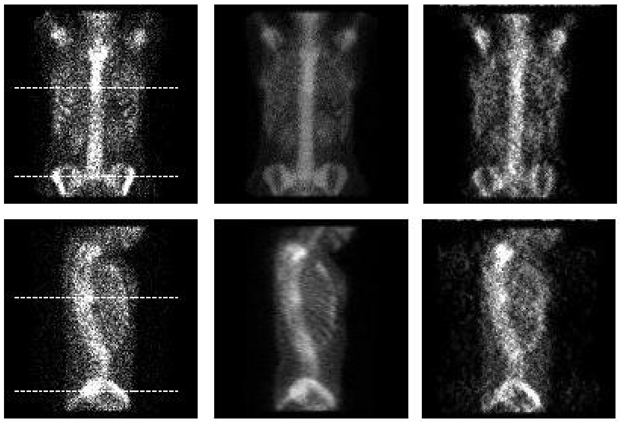


Fig. 16. Projections of the 3-D reconstructed object (flies): Acquisition (left), OSEM (middle), Flies (right)

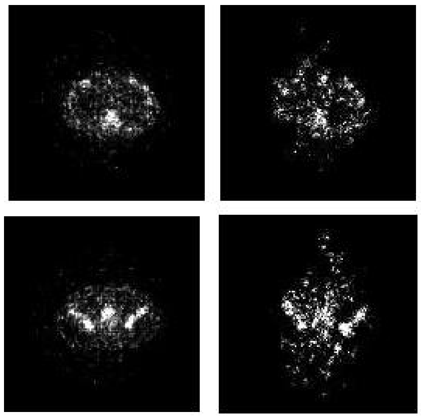


Fig. 17. Axial views of the same 3-D reconstructed object (flies), at waist (upper row) and thorax (lower) levels. (Dash lines of figure 16). OSEM (left) and Flies (right) reconstructions

Our algorithm was compared with a commercial 2D OSEM reconstruction. It well recovers shapes and contrast, with a processing time around 10 times longer than the optimized OSEM.

3.4 Example 4: Noise Resistance

In this test example, a Gaussian noise has been added to the images of example 1 (Fig. 9) and the same algorithm parameters have been used.

We observe that reconstruction is fairly noise resistant, probably thanks to the fact screen redundancy is exploited by the algorithm, although the brightness differences between the three cylinders are not rendered as clearly as with the noiseless images.

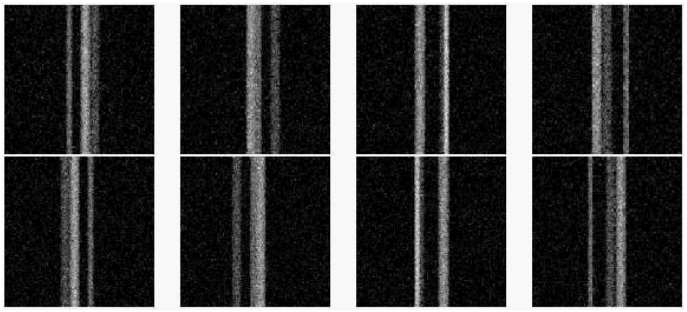


Fig. 18. Noisy synthesised projections of the object

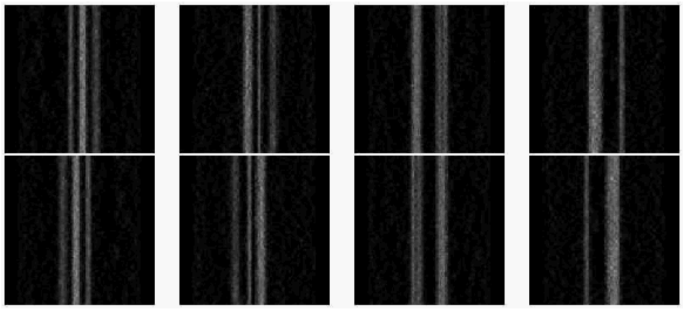


Fig. 19. Side views of the 3-D reconstructed object

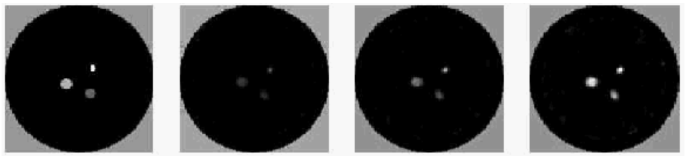


Fig. 20. Top views of the object reconstructed from noisy images (slices of 5, 10 and 20 pixels)

4 Conclusion

We demonstrated the validity of a generalization of the Fly algorithm introducing the marginal fitness calculation method, to constructing the 3-D shape of radioactive tracer concentration from SPECT images. Contrary to more classical approaches, our "fully 3-D reconstruction method" exploits all the projection images. The next stages of this research will concentrate upon building simplified but accurate models of scattering and absorption derived from complete Monte-Carlo simulation of Compton and Rayleigh scattering, exploit energy level information and x-ray absorption data, in order to get high quality results in realistic times. More elaborate validations than visual inspection must be achieved with ground truth images than only could be obtained by sophisticated Monte Carlo simulations.

Acknowledgements. We wish to specially thank the students who contributed to the ideas and methods presented in this paper and their implementation and testing, in particular: Sébastien Gaucher, Charles Brintet, Lionel Castillon, Marie Thépaut, Maria Rodriguez Lopez.

References

1. Bonnin, F., Buvat, I., Benali, H., di Paola, R.: Scatter correction in scintigraphy: the state of the art. European Journal of Nuclear Medicine 21(7) (July 1994)
2. Kalos, M., Wittlock, P.: Monte-Carlo methods. John Wiley and Sons, Chichester (1986)
3. Louchet, J.: Stereo Analysis Using Individual Evolution Strategy. In: ICPR 2000, Barcelona, Spain (September 2000)
4. Louchet, J.: Model-based Image Analysis using Evolutionary Strategies. In: Cagnoni (ed.) Genetic and Evolutionary Computation in Image and Computer Vision, Springer, Heidelberg (2007)
5. SIMSET Simulation System for Emission Tomography software, University of Washington, http://depts.washington.edu/simset/html/simset_main.html
6. Yanch, J.C., Flower, M.A., Webb, S.: A comparison of deconvolution and windowed subtraction techniques for scatter compensation, SPECT. IEEE transactions on Medical Imaging 7(1) (1988)

A Study of Crossover Operators for Gene Selection of Microarray Data

Jose Crispin Hernandez Hernandez, Béatrice Duval, and Jin-Kao Hao

LERIA, Université d'Angers,
2 Boulevard Lavoisier, 49045 Angers, France
{josehh,bd,hao}@info.univ-angers.fr

Abstract. Classification of microarray data requires the selection of a subset of relevant genes in order to achieve good classification performance. Several genetic algorithms have been devised to perform this search task. In this paper, we carry out a study on the role of crossover operator and in particular investigate the usefulness of a highly specialized crossover operator called GeSeX (GEne SElection crossover) that takes into account gene ranking information provided by a Support Vector Machine classifier. We present experimental evidences about its performance compared with two other conventional crossover operators. Comparisons are also carried out with several recently reported genetic algorithms on four well-known benchmark data sets.

Keywords: Microarray gene expression, Feature selection, Genetic algorithms, Support vector machines.

1 Introduction

Recent advances in DNA microarray technologies enable to consider molecular cancer diagnosis based on gene expression. Classification of tissue samples from gene expression levels aims to distinguish between normal and tumor samples, or to recognize particular kinds of tumors [9,2]. Gene expression levels are obtained by cDNA microarrays and high density oligonucleotide chips, that allow to monitor and measure simultaneously gene expressions for thousands of genes in a sample. So, the currently available data in this field concern a very large number of variables (thousands of gene expressions) relative to a small number of observations (typically under one hundred samples). This characteristic, known as the "curse of dimensionality", is a difficult problem for classification methods and requires special techniques to reduce the data dimensionality (gene selection) in order to obtain reliable predictive results.

Gene selection for microarray data is a special kind of feature selection that aims at finding a (small) subset of informative features from the initial data in order to obtain high classification accuracy [13]. Given the particular characteristic of "curse of dimensionality" of microarray data, gene selection for microarray data is known to be particularly difficult.

The literature offers a large number of solution methods for gene selection which are based on genetic algorithms, often combined with other approaches

[19,6,18,17,8,16,4,22]. For instance, the so-called wrapper approach uses GAs to search over the space of gene subsets, the fitness of each subset being evaluated by its classification performance obtained by a given classifier.

In this paper, we are interested in studying the genetic algorithms for gene selection. In particular, we focus our investigation on the very role of the crossover operator. Indeed, it is now well recognized that among the different components of a GA, the crossover operator may make a difference if it is carefully designed for the targeted problem.

The main contributions of the paper is to present in details a highly specialized crossover operator called GeSeX (GEne SElection crossover) introduced in [12] and to report extensive comparative studies of GeSeX with two other conventional crossover operators (uniform and single point). These results help to understand the behavior of these crossover operators and their relative performance when they are applied with a GA. Comparisons are also carried out with several recently reported genetic algorithms on four well-known benchmark data sets.

The paper is organized as follows; in Section 2, we review the main characteristics of Support Vector Machines (SVM) that are used in our approach. In Section 3, we describe the specialized crossover operator GeSeX and the other components of our GA. Experimental results and comparaisons are presented in Section 4 before conclusions are given in Section 5.

2 SVM Classification and Gene Selection

It is common in wrapper approaches for gene selection to use a classifier to evaluate the quality of a proposed gene subset. SVM classifier can be used for such purposes. The originality of our genetic algorithm is that a SVM classifier is used not only in the fitness evaluation of gene subsets but also in the genetic operators: actually, the characteristics of the SVM classifier are used to propose a specialized crossover operator. This section recalls the main characteristics of SVM and explains how a feature selection process can be guided by the informations provided by a SVM classifier.

2.1 Support Vector Machines

SVMs have been successfully used for gene selection and classification [11,20,15]. SVMs are state-of-the-art classifiers that solve a binary classification problem by searching a decision boundary that has the maximum margin with the examples. SVMs handle complex decision boundaries by using linear machines in a high dimensional feature space, implicitly represented by a kernel function. In this work, we only consider linear SVMs because they are known to be well suited to the datasets that we consider and they offer a clear biological interpretation.

For a given training set of labeled samples, a linear SVM determines an optimal hyperplane that divides the positively and the negatively labeled samples with the maximum margin of separation. A noteworthy property of SVM is that the hyperplane only depends on a small number of training examples called the support vectors, they are the closest training examples to the decision boundary and they determine the margin.

Formally, we consider a training set of n samples belonging to two classes; each sample is noted $\{X_i, y_i\}$ where $\{X_i\}$ is the vector of attribute values describing the sample and y_i the class label.

A soft-margin linear SVM classifier aims at solving the following optimization problem:

$$\min_{w,b,\xi_i} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \tag{1}$$

subject to $y_i (w \cdot X_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, $i = 1, ..., n$.

In this formulation, w is the weight vector that determines the separating hyperplane; C is a given penalty term that controls the cost of misclassification errors. To solve this optimization problem, it is convenient to consider the dual formulation [5]:

$$\min_{\alpha_i} \frac{1}{2} \sum_{i=1}^{n} \sum_{l=1}^{n} \alpha_i \alpha_l y_i y_l X_i \cdot X_l - \sum_{i=1}^{n} \alpha_i \tag{2}$$

subject to $\sum_{i=1}^{n} y_i \alpha_i = 0$ and $0 \leq \alpha_i \leq C$.

The decision function for the linear SVM classifier with input vector X is given by $f(X) = w \cdot X + b$ with $w = \sum_{i=1}^{n} \alpha_i y_i X_i$ and $b = y_i - w \cdot X_i$.

The weight vector w is a linear combination of training samples. Most weights α_i are zero and the training samples with non-zero weights are the support vectors.

2.2 Feature Ranking by SVM

As discussed in [11], the weights of a linear discriminant classifier can be used to rank the features for selection purposes. More precisely, in a backward selection method, the idea is to start with all the features and to iterate the removal of the least informative feature. To determine which feature can be removed, one can consider the feature that has the least influence on the cost function of the classification process. For a linear SVM, the cost function is defined by $\frac{1}{2}||w||^2$. So given a SVM with weight vector w, we can define the ranking coefficient vector c given by:

$$\forall i, c_i = (w_i)^2 \tag{3}$$

Intuitively, that means that in order to select informative genes, the orientation of the separating hyperplane found by a linear SVM can be used. If the plane is orthogonal to a particular gene dimension, then that gene is informative, and vice versa. As we show in the next section, the coefficient vector c can provide a dedicated crossover operator with very useful ranking information.

3 A Dedicated Genetic Algorithm for Gene Selection and Classification

Our genetic algorithm for gene selection begins by a pre-selection step that enables to reduce the gene subset space. For a given microarray dataset, we first filter the most interesting genes by the BW ratio criterion introduced in [7]; the number p of pre-selected genes is fixed at 50. From this reduced subset, we will determine an even smaller set of genes (typically < 10) which allows to give the highest classification accuracy. To achieve this goal, we propose a dedicated Genetic Algorithm which integrates, in its genetic operators, specific knowledges on our gene selection and classification problem. It relies on a linear SVM classifier to evaluate each individual and the ranking coefficient vector of this SVM enables to propose a highly informed crossover operator. In what follows, we present the main elements of this GA, focusing on the crossover operator. Other characteristics of our approcah can be found in [12].

3.1 Problem Encoding

An individual $I = < I^x, I^y >$ is composed of two parts I^x and I^y called respectively *gene subset vector* and *ranking coefficient vector*. The first part, I^x, is a binary vector of fixed length p. Each bit I^x_i $(i = 1...p)$ corresponds to a particular gene and indicates whether or not the gene is selected. The second part, I^y, is a positive real vector of fixed length p and corresponds to the ranking coefficient vector c (Equation 3) of the linear SVM classifier. I^y indicates thus for each selected gene the importance of this gene for the SVM classifier.

Therefore, an individual represents a candidate subset of genes with additional information on each selected gene with respect to the SVM classifier. The gene subset vector of an individual will be evaluated by a linear SVM classifier while the ranking coefficients obtained during this evaluation provide useful information for the evolutonary process.

3.2 SVM Based Fitness Evaluation

Given an individual $I = < I^x, I^y >$, the gene subset part I^x, is evaluated by two criteria: the abily to obtain a good classification in this gene subset representation and the number of genes contained in this subset. More formally, the fitness function is defined as follows:

$$f(I) = \frac{CA_{SVM}(I^x) + \left(1 - \frac{|I^x|}{p}\right)}{2} \tag{4}$$

The first term $(CA_{SVM}(I^x))$ is the classification accuracy obtained with a linear SVM classifier trained on this subset and evaluated via 10-fold cross-validation. The second term ensures that for two gene subsets having an equal classification accuracy, the smaller one is preferred.

For a given individual I, this fitness function leads to a positive real fitness value $f(I)$ (higher values are better). At the same time, the ranking vector c obtained from the SVM classifier is calculated and copied in I^y.

3.3 Specialized Crossover Operator [12]

Crossover is one of the key evolution operators for any effective GA and needs a particularly careful design. As our goal is to obtain small subsets of selected genes with a high classification accuracy, we have designed a highly specialized crossover operator following two fundamental principles: 1) to conserve the genes shared by both parents and 2) to preserve "high quality" genes from each parent even if they are not shared by both parents. The notion of "quality" of a gene here is defined by the corresponding ranking coefficient stored in I^y. Notice that applying the first principle will have as main effect of getting smaller and smaller gene subsets while applying the second principle allows us to keep up good genes along the search process.

Let $I =< I^x, I^y >$ and $J =< J^x, J^y >$ be two selected individuals (parents), we combine I and J to obtain a single child $K =< K^x, K^y >$ using the following steps:

1. Extract the subset of genes shared by both parents by boolean logic AND operator ($\otimes$) and arrange them in an intermediary gene subset vector F.

$$F = I^x \otimes J^x$$

2. For the subset of genes obtained from the first step, extract the maximum coefficients max_I and max_J accordingly from their original ranking vectors I^y and J^y.

$$max_I = max\,\{I_i^y \mid i\ such\ that\ F_i = 1\}$$

and

$$max_J = max\,\{J_i^y \mid i\ such\ that\ F_i = 1\}$$

3. This step aims to transmit high quality genes from each parent I and J which are not retained by the logic AND operator in the first step. These are genes with a ranking coefficient greater than max_I and max_J. The genes selected from I and J are stored in two intermediary vectors AI and AJ

$$AI_i = \begin{cases} 1 & if\ I_i^x = 1\ and\ F_i = 0\ and\ I_i^y > max_I \\ 0 & otherwise \end{cases}$$

and

$$AJ_i = \begin{cases} 1 & if\ J_i^x = 1\ and\ F_i = 0\ and\ J_i^y > max_J \\ 0 & otherwise \end{cases}$$

4. The gene subset vector K^x of the offspring K is then obtained by grouping all the genes of F, AI and AJ using the logical "OR" operator ($\oplus$).

$$K^x = F \oplus AI \oplus AJ$$

The ranking coefficient vector K^y will be filled up when the individual K is evaluated by the SVM based fitness function.

3.4 The General GA and Its Other Components

An initial population P is randomly generated such that the number of genes by each individual varies between p and $p/2$ genes. From this population, the fitness of each individual I is evaluated using the function defined by the formula 4. The ranking coefficient vector c of the SVM classifier is then copied to I^y.

To obtain a new population, a temporary population P' is used. To fill up P', we first copy the two best individuals of P to P' (elitism). The rest of P' is completed with individuals obtained by crossover and mutation. Precisely, Stochastic Universal Selection is applied to P to generate a pool of $|P|$ candidate individuals. From this pool, crossover is applied $0.49 * |P|$ times to pairs of randomly taken individuals, each new resulting individual being inserted in P'. Similarly, random mutation is applied $0.49 * |P|$ times to randomly taken individuals to fill up P'. Once P' is filled up, it replaces P to become the current population. The GA stops when a fixed number of generations is reached.

4 Comparison

In this section we present two comparative studies. The first compares the crossover operator GeSeX with two well-known crossover operators. In the second study, we carry out a comparison with four highly effective GA-based gene selection approaches [17,22,8,16].

4.1 Data Sets

We applied our approach on four well-known data sets that concern leukemia, colon cancer and two lymphoma data sets.

The leukemia data set consists of 72 tissue samples, each with 7129 gene expression values. The samples include 47 acute lymphoblastic leukemia (ALL) and 25 acute myeloid leukemia (AML). The data were produced from Affymetrix gene chips. The data set was first used in [9] and is available at http://www-genome.wi.mit.edu/cancer/.

The colon cancer data set contains 62 tissue samples, each with 2000 gene expression values. The tissue samples include 22 normal and 40 colon cancer cases. The data set is available at http://microarray.princeton.edu/oncology/affydata/index.html and was first studied in [2].

The first lymphoma data set is based on 4026 variables describing 47 samples (24 and 23 of which are respectively considered as GC B-Like samples and activated B-Like samples). The data set was first analyzed in [1]. The data set is available at http://llmpp.nih.gov/lymphoma/data.shtml.

The second lymphoma data set contains 58 patients with DLBCL each with 7129 gene expression values, 32 with cured disease and 26 with fatal or refractory disease. This is available at http://broad.mit.edu/cgi-bin/cancer/datasets.cgi. The data set was reported in [21].

Prior to running our method, we apply a linear normalization procedure to each data set to transform the gene expressions to mean value 0 and standard deviation 1.

4.2 Comparison of Crossover Operators

The purpose of the first experiment is to evaluate the performance of two well known crossover operators (single point and uniform crossovers) against our GeSeX crossover operator. The evaluation takes into account two aspects: the capacity to generate new potentially promising individuals and the ability to keep a diversified population. Both characteristics are very important in the whole search process because they represent the classical trade-off between exploration and exploitation.

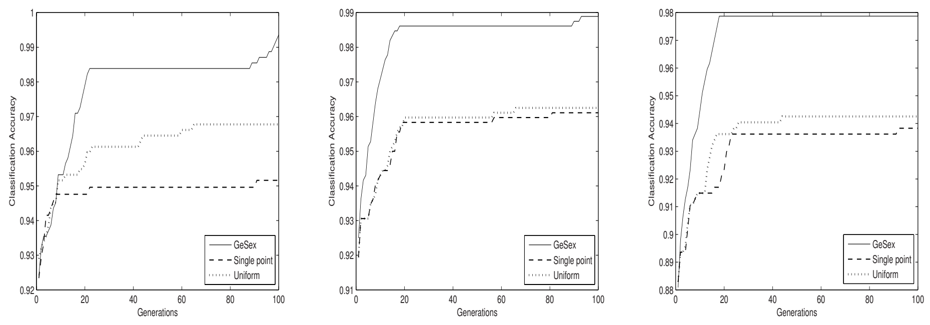
The first criterion is measured by the quality of the best individual of a population. For an individual, that is a gene subset, we measure its quality by the classification accuracy of a SVM classifier built on this gene subset. This accuracy is evaluated via a 10-fold cross validation, so for an individual I, it is exactly $CA_{SVM}(I^x)$ (see Section 3). The population diversity is calculated with the entropy measure proposed in [10] and recalled in Equation 5, where n_{ij} represent the number of times the gene i is set to the value j in the population P. This function takes values in the interval [0, 1]. An entropy of 0 indicates that all the individuals in the population are identical, while an entropy of 1 means that all the individuals are different.

$$Entropy\,(P) = \frac{\sum_{i=1}^{n}\sum_{j=0}^{1}\left(\frac{n_{ij}}{|P|}\right) log\left(\frac{n_{ij}}{|P|}\right)}{nlog2} \tag{5}$$

In order to enable a fair comparison, all the crossover operators were tested under the same conditions on three microarray datasets (Colon, Leukemia and the first lymphoma data set [1]). The following parameters were used in this experiment: a) population size $|P| = 100$, b) maximal number of generations is fixed at 100. We use a classical mutation where each bit of an individual has a mutation probabilty of 0.3. For the single point and uniform crossover operators, we use a crossover probability of 0.5, whereas the general settings for GeSeX operator are explained in subsection 3.4.

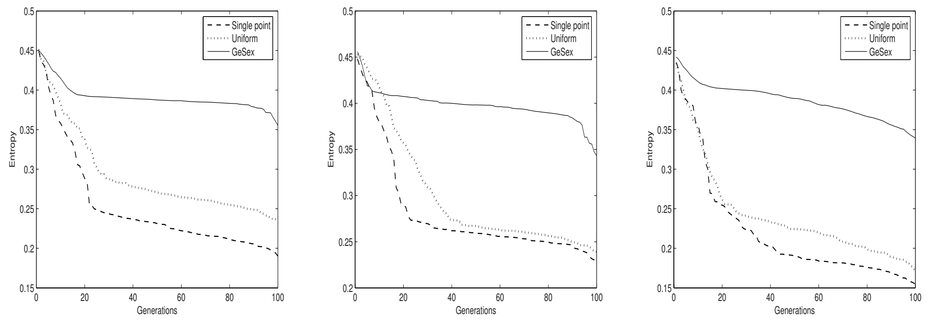
Due to the non deterministic character of GA, 10 independent runs were executed for each dataset/operator combination. The results are shown in figure 1 and in figure 2.

In figure 1 the X axis represents the number of generations, while the Y axis represents the accuracy of the best individual of a population, both averaged over the 10 runs. This figure shows clearly that the GeSeX operator allows us to obtain better results for the three datasets because it constantly reaches a higher classification accuracy. More specifically, let us examine the case of the Leukemia dataset. With the GeSeX operator, an average accuracy of 98.611% is rapidly reached by the best individual within 20 generations, meaning that for each of the 10 experiences, only one sample out of the 72 is misclassified in the cross-validation process. With the two other crossover operators, an average accuracy is onlt around 96% because in most experiences 3 examples out of 72 is misclassified and for one or two experiences, two samples out of 72 are misclassified. We can notice also that after 90 generations, the curve for GeSeX



a) Colon b) Leukemia c) Lymphoma

Fig. 1. Average classification accuracy of the best individuals of populations for Single point, Uniform and GeSeX crossover operators using three microarray datasets



a) Colon b) Leukemia c) Lymphoma

Fig. 2. Population entropy for Single point, Uniform and GeSeX crossover operators

leaves the stage of 98.611% because for one or two experiences among the 10, the best individual reaches the maximal accuracy of 100%.

In figure 2 we show how the population entropy evolves with the number of generations. Each point represents the average population entropy over all runs. Observe that GeSex keeps a higher population entropy that the other crossover operators. Therefore GeSex provides a good balance between quality and diversification of the population.

4.3 Comparison with Other Genetic Algorithms

In this section we carry out a comparison of our GA+GeSeX with four highly effective GA-based gene selection approaches.

Genetic Approaches. In [17] the authors propose a gene selection method that relies on two steps. The first step is a pre-selection that rank the genes according to an original filtering criterion proposed by the authors; the top genes are selected to construct a reduced search space, that the GA explores in order to minimize the number of selected genes. Their GA is a classical one with a multiple-point crossover. The paper reports the best classification accuracies estimated by LOOCV on the whole set of samples for a single run. Notice that the lymphoma data set is the one analysed in [21] and they find an final subset with 11 selected genes. For the colon cancer, they report a subset with 9 selected genes and for the leukemia data set they select a subset of 8 genes.

In [22] the authors propose a hybrid algorithm using SVM and GA. In the first step of their approach, a gene subset of size p is selected by Least Square Support Vector Machine to construct the search space of the GA. In the second step, they apply a GA to carry out gene selection. The particularity of their GA is that crossover and mutation operators are designed to keep the same number p of genes. So their objective is to explore all the subsets of size p in order to find the best one. The fitness function of a gene subset uses the information entropy of the classes represented on that gene subset. When the GA terminates, they evaluate the quality of the selected gene subset by the accuracy of a SVM classifier. For colon cancer, the test set has 32 samples and the best accuracy (over one run) is obtained with 20 selected genes. For leukemia, the test set has 34 samples and their best result is obtained with 15 selected genes.

In [8] the authors propose a genetic method that is not a wrapper approach: the GA explores the space of subsets and each candidate subset is evaluated by two clustering measures. The idea is to consider the two classes of the data set as two given clusters and to compare the quality of the clusters when the gene subset used to represent the data is changed. Such a GA-Filter approach requires a lower computational burden since the fitness evaluation does not require a classifier training. For each data set, 10 runs of GA-Filter are executed and each time, the gene subset selected by GA-Filter is evaluated by a classification experiment where different classifiers are tried. The paper presents the average and standard deviation of the classification accuracy over these 10 runs. We retain for comparison the best result reported in the paper, for each dataset that we consider. Notice that the lymphoma is the one presented in [1]. The number of selected genes were respectively 15, 17 and 10 together with respectively 34, 22, and 13 testing samples for the leukemia, colon and lymphoma datasets.

In [16] the authors combine SVM and GA in another way. Their SVM uses a kernel function that combines a set of simple kernel functions and they propose a new learning method exploiting Evolutionary Algorithm technique to obtain an optimal decision model. So their genetic search aims to find out the optimal set of features but also the optimal set of parameters for the combined kernel function. The average of the classification accuracy over 10 independent runs is provided for colon and leukemia datasets. The number of selected genes were 15 in both datasets.

Experimental Context and Results of Comparison. In order to compare our approach with each of these four works, we apply our genetic algorithm to each data set with exactly the same experimental conditions as those reported in the corresponding paper. More precisely, we fix the number of genes in

Table 1. Comparison of four GA-based selection approaches and our method. The table gives the number of genes and the classification accuracy reported by each author (*Reported*) and the classification accuracy obtained by our approach (**GeSeX**) when we fix the number of genes to the value used in the corresponding paper.

Data set	[17]			[22]			[8]			[16]		
	Reported		**GeSeX**	*Reported*		**GeSeX**	*Reported*		**GeSeX**	*Reported*		**GeSeX**
Leukemia	*8*	*98.6*	**100**	*15*	*97.1*	**100**	*15*	*99.70*	**98.82**	*15*	*77.06*	**98.82**
Colon	*9*	*95.1*	**100**	*20*	*90.6*	**93.75**	*17*	*77.50*	**85.9**	*15*	*75.33*	**86.0**
Lymph.[1]	-		-	-		-	*10*	*96.15*	**96.92**	-		-
Lymph.[21]	*11*	*100*	**100**	-		-	-		-	-		-

our method, that means that for each data set and each previously cited work [17,22,8,16], we determine which classification accuracy can be obtained by our GA for the number of genes reported in this work. Moreover, we evaluate the classifier accuracy with the same number of runs: for [17] and [22], the result is the best accuracy obtained in one run while for [8] and [16], this is the average over 10 runs. We also use the same test samples as the authors for each dataset, this is important because previous studies have shown that the accuracy estimate may be biased and may have an important variance [3]. In this experiment, our genetic algorithm uses also a specialized mutation operator [12] that uses ranking information provided by the SVM and stored in the ranking coefficient vector I^y to eliminate "mediocre" genes.

Table 1 summarizes the comparison: the number of genes and the classification accuracy reported in the papers are in front of the classification accurary obtained by our method. Some cells of the table contain no information because the experiment on the corresponding data set is not available in the papers.

From Table 1, we observe that the results of our GA are better than those published results, except for the result of leukemia reported in [8]. As indicated, in these experiments we restrict our method to consider the same number of selected genes as in the reported works. In fact, our method is able to optimize two criteria: the number of selected genes and the classification accuracy. So, our method is able to select smaller subsets of informatives genes with high classification accuracy. Concretely, we have experimented our method on 50 trials for Leukemia and Colon data sets and we obtain the following results [12]. For Leukemia, the number of selected genes was respectively 3.17 ± 1.16 and the accuracy (evaluated by a 10-fold cross validation) was 91.5 ± 5.9; for Colon, the number of selected genes was 7.05 ± 1.07 and the accuracy was 84.6 ± 6.6. Those numbers cannot be compared with [17] and [22], which do not provide averaged results but they are comparable with those of [16,8] and better in the sense that the number of genes is smaller.

5 Conclusions and Future Work

We have presented a study on the role of the crossover operators for gene selection of microarray data. We have presented a specialized crossover operator

GeSeX that is used in a wrapper genetic algorithm. Contrary to conventional crossover operators, GeSex takes into account the information provided by the SVM classifier used by our fitness function.

Our experimental analysis shows that this crossover operator behaves more efficiently than traditional crossover operators and that it ensures a good trade-off between exploration and exploitation of the search space. We also compare our GA+GeSeX approach to other recently proposed GA devoted to the task of gene selection and classification of microarray data. These experimentations show that GA+GeSex gives globally very competitive results.

We are currently studying alternative fitness functions to provide a more effective guidance of the genetic process. Moreover we are developing a local search based mutation operator in order to intensify the genetic search.

Acknowledgments. This work is partially supported by the French Ouest Genopole® and the "Bioinformatique Ligérienne" projects. The first author is supported by a Mexicain PROMEP-DGEST scholarship. The authors of this paper would like to thank the reviewers for their useful comments.

References

1. Alizadeh, A., Eisen, M.B., Davis, E., Ma, C., Lossos, I., Rosenwald, A., Boldrick, J., Sabet, H., Tran, T., Yu, X., Powell, J.I., Yang, L., Marti, G.E., Hudson Jr, J., et al.: Distinct types of diffuse large B–cell lymphoma identified by gene expression profiling. Nature 403, 503–511 (2000)
2. Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proceedings of the National Academy of Sciences USA 96, 6745–6750 (1999)
3. Ambroise, C., McLachlan, G.J.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proceedings of the National Academy of Sciences USA 99(10), 6562–6566 (2002)
4. Bonilla Huerta, E., Duval, B., Hao, J.-K.: A hybrid ga/svm approach for gene selection and classification of microarray data. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 34–44. Springer, Heidelberg (2006)
5. Boser, B.E., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. ACM Press, New York (1992)
6. Deb, K., Reddy, A.R.: Reliable classification of two-class cancer data using evolutionary algorithms. Biosystems 72(1-2), 111–129 (2003)
7. Dudoit, S., Fridlyand, J., Speed, T.P.: Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association 97(457), 77–87 (2002)
8. Feres de Souza, B., de Carvalho, E.C.P.L.F.: Gene Selection Using Genetic Algorithms. In: Barreiro, J.M., Martín-Sánchez, F., Maojo, V., Sanz, F. (eds.) ISBMDA 2004. LNCS, vol. 3337, pp. 479–490. Springer, Heidelberg (2004)

9. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286, 531–537 (1999)
10. Grefenstette, J.J.: Incorporating Problem Specific Knowledge into Genetic Algorithms. In: Davis, L. (ed.) Genetic Algorithms and Simulated Annealing, pp. 42–60. Morgan Kaufmann Publishers, London (1987)
11. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning 46(1-3), 389–422 (2002)
12. Hernandez Hernandez, J.C., Duval, B., Hao, J.-K.: A genetic embedded approach for gene selection and classification of microarray data. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 90–101. Springer, Heidelberg (2007)
13. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–324 (1997)
14. Liu, J., Iba, H.: Selecting informative genes using a multiobjective evolutionary algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, pp. 297–302. IEEE Press, Los Alamitos (2002)
15. Marchiori, E., Jimenez, C.R., West-Nielsen, M., Heegaard, N.H.H.: Robust svm-based biomarker selection with noisy mass spectrometric proteomic data. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 79–90. Springer, Heidelberg (2006)
16. Nguyen, H.-N., Ohn, S.-Y., Park, J., Park, K.-S.: Combined Kernel Function Approach in SVM for Diagnosis of Cancer. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3610, pp. 1017–1026. Springer, Heidelberg (2005)
17. Ni, B., Liu, J.: A Novel Method of Searching the Microarray Data for the Best Gene Subsets by Using a Genetic Algorithm. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1153–1162. Springer, Heidelberg (2004)
18. Paul, T.K., Iba, H.: Selection of the most useful subset of genes for gene expression-based classification. In: Proceedings of the 2004 Congress on Evolutionary Computation, pp. 2076–2083. IEEE Press, Los Alamitos (2004)
19. Peng, S., Xu, Q., Ling, X.B., Peng, X., Du, W., Chen, L.: Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. FEBS Letters 555(2), 358–362 (2003)
20. Rakotomamonjy, A.: Variable selection using svm-based criteria. Journal of Machine Learning Research 3, 1357–1370 (2003)
21. Shipp, M.A., Ross, K.N., Tamayo, P., Weng, A.P., Kutok, J.L., Aguiar, R.C., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G.S., Ray, T.S., et al.: Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nature Medecine 8(1), 68–74 (2002)
22. Xiong, W., Zhang, C., Zhou, C., Liang, Y.: Selection for Feature Gene Subset in Microarray Expression Profiles Based on a Hybrid Algorithm Using SVM and GA. In: Min, G., Di Martino, B., Yang, L.T., Guo, M., Rünger, G. (eds.) ISPA Workshops 2006. LNCS, vol. 4331, pp. 637–647. Springer, Heidelberg (2006)

Searching for Glider Guns in Cellular Automata: Exploring Evolutionary and Other Techniques

E. Sapin and L. Bull

Faculty of Computing, Engineering and Mathematical Sciences,
University of the West of England,
Bristol BS16 1QY, UK
emmanuelsapin@hotmail.com, Larry.Bull@uwe.ac.uk

Abstract. We aim to construct an automatic system for the discovery of collision-based universal cellular automata that simulate Turing machines in their space-time dynamics using gliders and glider guns.

In this paper, an evolutionary search for glider guns with different parameters is described and other search techniques are also presented as benchmark. We demonstrate the spontaneous emergence of an important number of novel glider guns discovered by genetic algorithms.

1 Introduction

The emergence of computation in complex systems with simple components is a hot topic in the science of complexity [1]. A uniform framework to study emergent computation in complex systems are cellular automata [2]. They are discrete systems in which an array of cells evolves from generation to generation on the basis of local transition rules [3].

The well-established problems of emergent computation and universality in cellular automata has been tackled by a number of people in the last thirty years [4], [5], [6], [7], [8] and remains an area where amazing phenomena at the edge of theoretical computer science and non-linear science can be discovered.

The most known universal automaton is the Game of Life [9]. It was shown to be universal by Conway [10] who employed *gliders* and *glider guns*. Gliders are mobile self-localized patterns of non-resting states, and glider guns are patterns which, when evolving alone, periodically recover their original shape after emitting some gliders.

The search for gliders was notably explored by Adamatzky et al. with a phenomenological search [11], Wuensche who used his Z-parameter and entropy [12] and Eppstein [13]. Sapin et al. have considered the emergence of gliders-based universality by use of a genetic algorithm [14].

We aim to construct an automatic system for the discovery of computationally universal cellular automata, spatially. Inspired by the link between universality and the presence of gliders and glider guns in cellular automata, we are here interested in the emergence of glider guns. In this paper, three search methods for glider guns are compared.

The paper is arranged as follows: Section 2 describes previous related work. Section 3 sets out the characteristics of the search methods. Then the result of the best search method are described in Section 4. The last section summarizes the presented results and discusses directions for future research.

2 Previous Work

In this section, some previous work about cellular automata are presented. Brief descriptions of some search methods are given. Then some previous work about using an evolutionary approach to search for automata are presented.

2.1 Cellular Automata

In [15], Wolfram studies the space $\mathcal{I}$ of 2D isotropic CA, with rectangular 8-cell neighbourhoods: if two cells have the same neighbourhood states by rotations and symmetries, then these two cells take the same state at the next generation. There are 512 different rectangular 8-cell neighbourhood states. An automaton of I can be described as shown figure 1 by telling what will become of a cell in the next generation, depending on its subset of isotropic neighbourhood states.

There are 102 subsets of isotropic neighbourhood states, meaning that there are 2^{102} different automata in $\mathcal{I}$.

2.2 Search Methods

In order to search for universal automata, we have examined the search methods such as monte carlo, taboo search [16] and an evolutionary algorithm [17], as briefly described here.

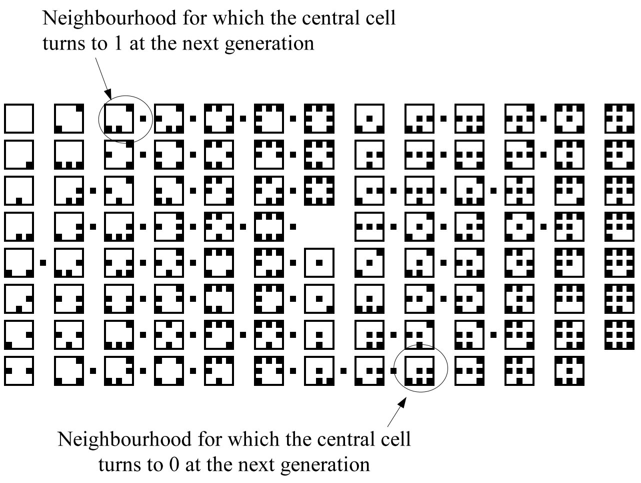

Neighbourhood for which the central cell turns to 1 at the next generation
Neighbourhood for which the central cell turns to 0 at the next generation

Fig. 1. The squares are the 102 neighbourhood states describing an automaton of $\mathcal{I}$. A black cell on the right of the neighbourhood state indicates a future central cell.

The monte carlo method consists solely of generating random solutions and testing them.

Tabu search traverses the solution space by testing mutations of an individual solution. Tabu search generates many mutated solutions and moves to the best solution of those generated. In order to prevent cycling and encourage greater movement through the solution space, a tabu list is maintained of partial or complete solutions. It is forbidden to move to a solution that contains elements of the tabu list, which is updated as the solution traverses the solution space.

Evolutionary algorithms have been used with cellular automata in a number of ways, after [18].

2.3 Evolving Cellular Automata

Previously, several good results from the evolution of cellular automaton rules to perform some useful tasks have been published. Mitchell et al. [19,20,21,22] have investigated the use of evolutionary computing to learn the rules of uniform one-dimensional, binary cellular automata. Here a Genetic Algorithm produces the entries in the update table used by each cell, candidate solutions being evaluated with regard to their degree of success for the given task — density and synchronization.

Sipper [23] has presented a related approach, which produces non-uniform solution. Each cell of a one or two-dimensional cellular automata is viewed as a genetic algorithm population member, mating only with its lattice neighbours and receiving an individual fitness. He shows an increase in performance over Mitchell et al.'s work, exploiting the potential for spatial heterogeneity in the tasks. Koza et al. [24] have also repeated Mitchell et al.'s work, using Genetic Programming [25] to evolve update rules. They report similar results.

3 Search for Glider Guns

This section describes the used search for glider guns. To compare the parameters of the search methods, glider guns that emitting the glider in figure 2 are searched for. The first search method is an evolutionary algorithm. Monte carlo algorithm and tabu search are also used as benchmark.

3.1 Evolutionary Algorithm

The parameters of the evolutionary algorithm are described here. The choice to try different parameters has been taken to find the best ones.

Fitness Function

Two fitness functions have been tried.

- First fitness function

  The computation of the fitness function is based on the one used in [26]. A random configuration of cells is evolved by the tested automaton. After this evolution, the presence of gliders G is checked by scanning the result of the configuration of the cells. The value of the fitness function is the number of gliders that appeared divided by the total number of cells.

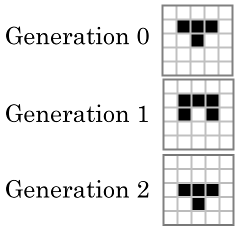


Fig. 2. The glider emitted by the searched glider gun

– Second fitness function
  The computation of the second fitness function is based on the first one. A random configuration of cells is evolved by the tested automaton. After this evolution, the presence of gliders G is checked by scanning the result of the configuration of the cells. The size of the biggest square S without cells in the evolved configuration of cells is computed. The value of the fitness function is the multiplication of the number of gliders that appeared by the size of the square S.

Initialization

The search space is the set $\mathcal{I}$ described in Section 2. Cell-state transition table can describe an automaton of this space. An individual is an automaton coded as a bit string of 102 Booleans representing the values of a cell at the next generation for each neighbourhood state.

The research has been guided by chosing automata for initialisation that accept the glider of figure 2. In order to chose these automata, the 102 bits of an automaton are divided into two subsets. The first subset, called invarious subset, is the neighbourhood states used by the glider G and their values are determined by the evolution of G. The process that determines these neighbourhood states is detailed for the glider of figure 2 in the figure 3. The other neighbourhood states are in the second subset, called unused subset, and are initialised at random.

The population size is 100 individuals.

Genetic Operators

The research has been guided again by chosing mutated automata that still accept the glider of figure 2. The mutation function then simply consists of mutating one bit among the second subset of the 102 bits. The rates 1,5 and 10 percent are tried, together with three crossover operators.

– No crossover
  The genetic algorithm is tried without crossover.
– Central point
  A single point crossover with a locus situated exactly on the middle of the genotype is tried.
– Random point
  A last kind of recombination is tried with a single point crossover with a locus randomly situated.

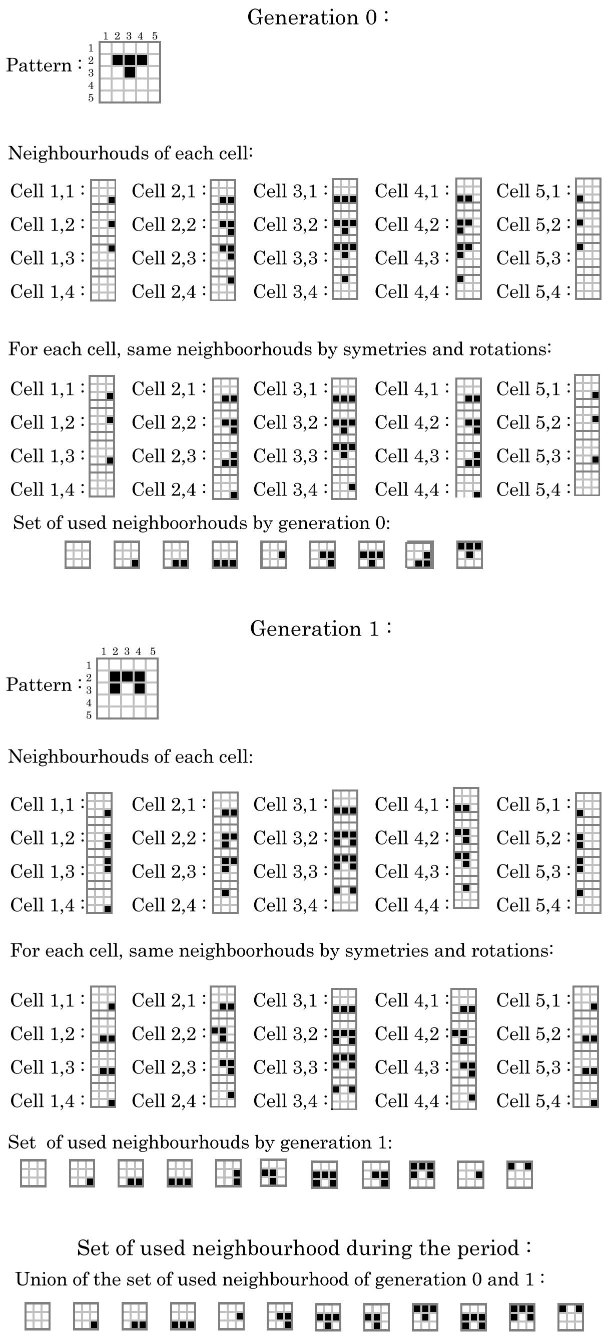

Generation 0 :
Pattern :
1 2 3 4 5
Neighbourhouds of each cell:
Cell 1,1 : Cell 2,1 : Cell 3,1 : Cell 4,1 : Cell 5,1 :
Cell 1,2 : Cell 2,2 : Cell 3,2 : Cell 4,2 : Cell 5,2 :
Cell 1,3 : Cell 2,3 : Cell 3,3 : Cell 4,3 : Cell 5,3 :
Cell 1,4 : Cell 2,4 : Cell 3,4 : Cell 4,4 : Cell 5,4 :
For each cell, same neighboorhouds by symetries and rotations:
Set of used neighboorhouds by generation 0:
Generation 1 :
Pattern :
Neighbourhouds of each cell:
For each cell, same neighboorhouds by symetries and rotations:
Set of used neighbourhouds by generation 1:
Set of used neighbourhood during the period :
Union of the set of used neighbourhood of generation 0 and 1 :

Fig. 3. Detail of the construction of set of neighbourhood states that are used by a glider

A linear ranking selection and a binary tournament selection of size 2 are tried.

Evolution Engine

An elitist strategy in which the best half of population is kept and a non-elitist strategy in which the new population is made of only children are tried.

Stopping Criterion

The presence of a glider gun is continuously checked. The test is inspired by Bays' test [27] and also used in [28]. After the evolution of the random configuration of cells, the pattern is isolated and tested in an empty universe. If a pattern P reappears at the same place with gliders around then the pattern P is a glider gun. When a glider gun is found the algorithm stops.

Some executions of the algorithm can be very long so the choice to stop the algorithm if better automata are not found has been taken. Then the value of the fitness function and the generation of the best rule are memorized. If after ten new generations the algorithm has not found a better rule the algorithm stops.

Thanks to these stopping criteria, an execution of the algorithm stops after an average of 38 generations.

3.2 Monte Carlo Method

In one million randomly generated automata, the presence of glider guns after the evolution of a random configuration of cells was tested. The test is the one used for the stopping criterion of the algorithm described in Section 3.1. There are not any guns found by this method.

3.3 Tabu Search

A random automaton A is generated, two fitness functions are tried to measure the performance of this automaton:

All the automata obtained by mutating one bit among the unused subset, as described section 2.5, are tested by the fitness function. The best one who is not in a list L of the last chosen automaton is chosen to become the new automaton A. The sizes of 10, 100 and 1000 are tried for the list L. The presence of glider guns is checked in all the tested automata.

The algorithm stops when the best automaton, among the automata obtained by mutation, who is not in a list L is not better than the current automaton. With this stopping criterion, an execution of the algorithm stop after an average of 49, 42 and 35 generations depending on the size of the list L.

3.4 Discussion

For each of the values of the parameters, the number of executions which find a gun are shown in table 1.

The best parameters for the evolutionary algorithm, among the tested ones, are a mutation rate of 1, a non elitist strategy, a tournament selection and a

Table 1. Number of executions from a total of 100 per experiment that find a gun under a given combination of parameters or operators. The three numbers correspond to the 1,5, and 10 mutation rates. No guns were found with the monte carlo algorithm.

Evolutionary algortihm:

	First fitness function:				Second fitness function:			
	Elitist		Non Elitist		Elitist		Non Elitist	
	Tournament	Ranking	Tournament	Ranking	Tournament	Ranking	Tournament	Ranking
No Crossover	16, 32, 8	14, 14, 13	42, 11, 12	8, 1, 4	15, 43, 12	19, 16, 14	50, 13, 15	8, 3, 5
Middle Crossover	21, 18, 17	21, 14, 10	64, 20, 19	54, 23, 19	31, 23, 16	18, 14, 16	84, 22, 25	64, 33, 23
Random Crossover	3,2,2	3, 3, 2	4, 2, 0	1, 2, 1	1,1,4	0, 2, 3	3, 4, 1	0, 3, 0

Tabu search:

	First fitness function	Second fitness function
List of size 10	12	20
List of size 100	15	25
List of size 1000	16	28

central crossover and second fitness function. The evolutionary algorithm with these parameters have been chosen to obtain the glider guns described in the next Sections.

The good results of the central crossover can be explained by the fact that the first 51 neighbourhood states determine the birth of cells, while the other 51 determine how they survive or die. The elitist strategy that kept half of the population is worse than the non-elitist strategy. An elitist strategy that just kept one parent could be tried, however.

The results of a monte carlo algorithm and tabu search,presented as benchmarks, are not as good as the evolutionary approach. The results of the evolutionary algorithms without crossover are about the same of the tabu search.

4 Results of the Algorithm

The results of the genetic algorithm with the best parameters for the glider in figure 2 are described here.

4.1 Number of Guns

In order to determine how many different glider guns were found, an automatic system that determines if a gun is new is required. So, in order to determine if a gun is new, the set of neighbourhood states used by the given gun are compared to the ones of the other guns. For each gun and each neighbourhood state, three cases exist:

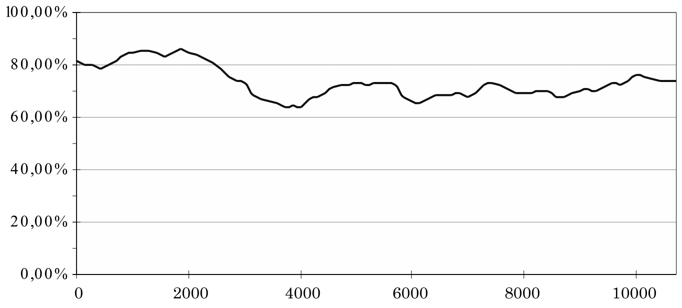


Fig. 4. Evolution of the percentage of new guns among 1000 different found guns

– The neighbourhood state is not used by this gun.
– The neighbourhood state is used by this gun and the value of the central cell at the next generation is 0.
– The neighbourhood state is used by this gun and the value of the central cell at the next generation is 1.

Two guns are different iff at least one neighbourhood state is not in the same case for the two guns.

Thank to this qualification of different guns leads, through the experimentations, 10008 different glider guns were discovered. All these guns have emerged spontaneously from random configurations of cells. The 10008 guns can be found in [29] in Life format.

The total number of different guns findable by this algorithm is unknown but the evolution of the percentage of new guns among the last 1000 different found guns is given by the figure 4.

In order to estimate the total number of different guns findable by this algorithm, Suppose each gun has the same probability to be found.

Let N be the total number of guns findable by this algorithm. The probability of a gun found by the algorithm to be new would be $1 - 10008/N$. The number of new guns among the last 1000 different found guns is 755. So the total number of guns findable by the algorithm could be estimated by $N = 10008 * 1000/245$ about 40849.

4.2 The Most Discovered Gun

The most discovered gun is shown figure 5. This gun emits two gliders toward two opposite directions. These two gliders are lined up and dephased. This gun is exhibited by the rule in figure 6.

5 Synthesis and Perspectives

This paper deals with the emergence of computation in complex systems with local interactions. A search for glider guns has been presented, building on previous work in [26,28].

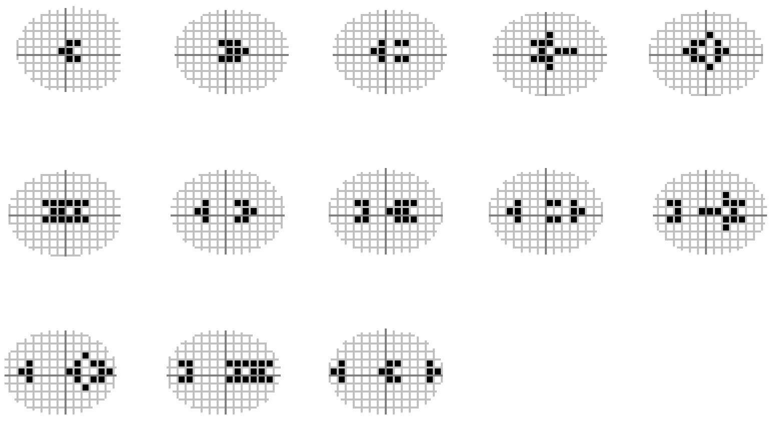


Fig. 5. The most discovered gun during a period

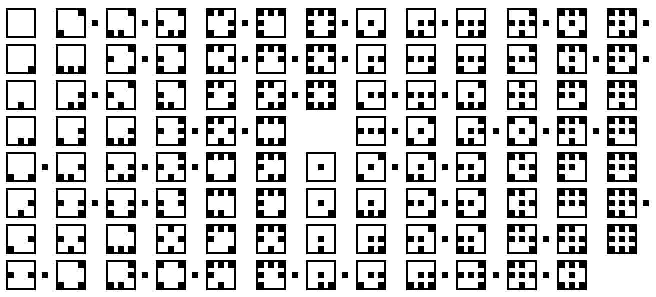


Fig. 6. The transition rule of the cellular automata in which the most discovered gun during a period was discovered

In particular, monte carlo method, tabu search and evolutionary algorithms are explored with different parameters. The best results are found for an evolutionary algorithm. The experimentation showed that cross over in the evolutionary algorithm plays a key role in the search process. Future work will consider other search techniques like a meta-EA to explore the search space of operators/rates could be implemented.

The algorithm succeeded in finding 10008 glider guns [29] for the glider of figure 2. The discovery of the emergence and existence of so many different glider guns for the same glider represent a contribution to the area of complex systems that considers computational theory.

Further goals can be to find all the glider guns possible and to calculate how many automata exhibit these guns. All these automata may be potential candidates for being shown universal automata thanks to an automatic system for the demonstration of universal automata that can be developed. Then, another domain that seems worth exploring is how this approach could be extended to automata with more than 2 states or more than 2 dimensions.

Future work could also calculate for each automata some rule-based parameters, e.g., Langton's lambda [30]. All automata exhibing glider guns may have similar values for these parameters that could lead to find areas between chaos and order, to a better understanding of the link between the rule transition and the emergence of computation in cellular automata and therefore the emergence of computation in complex systems with local interactions.

Acknowledgements

The work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom, Grant EP/E005241/1.

References

1. Wolfram, S.: A New Kind of Science. Wolfram Media, Inc., Illinois, USA (2002)
2. Von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana, Ill (1966)
3. Wolfram, S.: Universality and complexity in cellular automata. Physica D 10, 1–35 (1984)
4. Banks, E.R.: Information and transmission in cellular automata. PhD thesis, MIT (1971)
5. Margolus, N.: Physics-like models of computation. Physica D 10, 81–95 (1984)
6. Lindgren, K., Nordahl, M.: Universal computation in simple one dimensional cellular automata. Complex Systems 4, 299–318 (1990)
7. Morita, K., Tojima, Y., Katsunobo, I., Ogiro, T.: Universal computing in reversible and number-conserving two-dimensional cellular spaces. In: Adamatzky, A. (ed.) Collision-Based Computing, pp. 161–199. Springer, Heidelberg (2002)
8. Adamatzky, A.: Universal dymical computation in multi-dimensional excitable lattices. International Journal of Theoretical Physics 37, 3069–3108 (1998)
9. Gardner, M.: The fantastic combinations of John Conway's new solitaire game "life". Scientific American 223, 120–123 (1970)
10. Berlekamp, E., Conway, J.H., Guy, R.: Winning ways for your mathematical plays. Academic Press, New York (1982)
11. Adamatzky, A., Martinez, G.J., McIntosh, H.V.: Phenomenology of glider collisions in cellular automaton rule 54 and associated logical gates chaos. Fractals and Solitons 28, 100–111 (2006)
12. Wuensche, A.: Discrete dinamics lab (ddlab) (2005), http://www.ddlab.org
13. Eppstein, D., http://www.ics.uci.edu/~eppstein/ca/
14. Sapin, E., Bailleux, O., Chabrier, J.J., Collet, P.: A new universel automata discovered by evolutionary algorithms. In: Deb, K., al., e. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 175–187. Springer, Heidelberg (2004)
15. Wolfram, S., Packard, N.H.: Two-dimensional cellular automata. Journal of Statistical Physics 38, 901–946 (1985)
16. Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13, 533–549 (1986)
17. Holland, J.H.: Adaptation in natural and artificial systems, University of Michigan (1975)

18. Packard, N.H.: Adaptation toward the edge of chaos. In: Kelso, J.A.S., Mandell, A.J., Shlesinger, M.F. (eds.) Dynamic Patterns in Complex Systems, pp. 293–301. World Scientific, Syngapore (1988)
19. Mitchell, M., Crutchfield, J.P., Hraber, P.T.: Evolving cellular automata to perform computations: Mechanisms and impediments. Physica D 75, 361–391 (1994)
20. Hraber, P.T., Mitchell, M., Crutchfield, J.P.: Revisiting the edge of chaos: Evolving cellular automate to perform computations. Complex systems 7, 89–130 (1993)
21. Hordijk, W., Crutchfield, J.P., Mitchell, M.: Mechanisms of emergent computation in cellular automata. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature-V, vol. 866, pp. 344–353. Springer, Heidelberg (1998)
22. Das, R., Crutchfield, J.P., Mitchell, M., Hanson, J.E.: Evolving globally synchronized cellular automata. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 336–343 (1995)
23. Sipper, M.: Evolution of parallel cellular machines. In: Stauffer, D. (ed.) Annual Reviews of Computational Physics, pp. 243–285. V. World Scientific, Singapore (1997)
24. Andre, D., Koza, J.R., Bennett III, F.H., Keane, M.A.: Genetic programming iii: Darwinian invention and problem solving. Morgan Kaufmann, San Francisco, CA (1999)
25. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA (1992)
26. Sapin, E., Bailleux, O., Chabrier, J.J.: Research of complex forms in the cellular automata by evolutionary algorithms. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 373–400. Springer, Heidelberg (2004)
27. Bays, C.: Candidates for the game of life in three dimensions. Complex Systems 1, 373–400 (1987)
28. Sapin, E., Bailleux, O., Chabrier, J.J.: Research of a cellular automaton simulating logic gates by evolutionary algorithms. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 414–423. Springer, Heidelberg (2003)
29. Sapin, E.: `http://uncomp.uwe.ac.uk/sapin/ea/gun`
30. Langton, C.L.: Computation at the edge of chaos. Physica D 42 (1990)

A Genetic Algorithm for Generating Improvised Music

Ender Özcan and Türker Erçal

Yeditepe University,
Department of Computer Engineering,
34755 Kadıköy/İstanbul, Turkey
`{eozcan,tercal}@cse.yeditepe.edu.tr`

Abstract. Genetic art is a recent art form generated by computers based on the genetic algorithms (GAs). In this paper, the components of a GA embedded into a genetic art tool named AMUSE are introduced. AMUSE is used to generate improvised melodies over a musical piece given a harmonic context. Population of melodies is evolved towards a *better* musical form based on a fitness function that evaluates ten different melodic and rhythmic features. Performance analysis of the GA based on a public evaluation shows that the objectives used by the fitness function are assembled properly and it is a successful artificial intelligence application.

1 Introduction

Evolutionary art allows the artists to generate complex computer artwork without a need to delve into the actual programming used. This can be provided by creative evolutionary systems. Such systems are used to aid the creativity of users by helping them to explore the ideas generated during the evolution, or to provide users new ideas and concepts by generating innovative solutions to problems previously thought to be only solvable by creative people [1]. In these systems, the role of a computer is to offer choices and a human is to select one. This can be achieved in two ways. The choices of the human(s) can be used interactively or can be specified before and get autonomous results without interaction with the computer. The latter art form is also referred as the *genetic art*. In any case a human specifies some criteria and the computers are used for their capacity to investigate large search spaces. Composing a musical piece involves many stages, but surely, search forms a substantial part. Searching the right notes and durations can be an example. The existence of such a search process might require a pruning process to throw out the useless or unsuitable ideas. Musical composition can be considered as an exploration for an optimal musical piece in a very large search space.

Genetic algorithms (GAs) represent a class of algorithms used for search and optimization inspired from the natural evolution. They were rediscovered by J. Holland [6], and have been used to solve many difficult problems [5, 11, 12]. In GAs, a set of *chromosomes*, representing candidate solutions is evolved towards an optimal solution. This set is referred to as *population*. A chromosome consists of *genes*, where each gene receives a value from an *allele* set. The most common representation scheme is binary encoding that uses {0, 1} as an allele set. Depending on the problem, other appropriate encodings are allowed. In an evolutionary cycle, a set of genetic operators, such as, *crossover* and *mutation* is employed on initially randomly generated chromosomes. *Good building blocks*, possibly some part of an optimal solution are maintained within the population during the evolutionary process, while the *poor* ones are pruned based on an evaluation measured by a *fitness function*. With these properties, GAs can be considered as a suitable approach for generating musical composition. At the first glance, it might seem to be impossible for an algorithm to simulate the creative thinking process of a human. But GAs have achieved considerable results in the artistic fields.

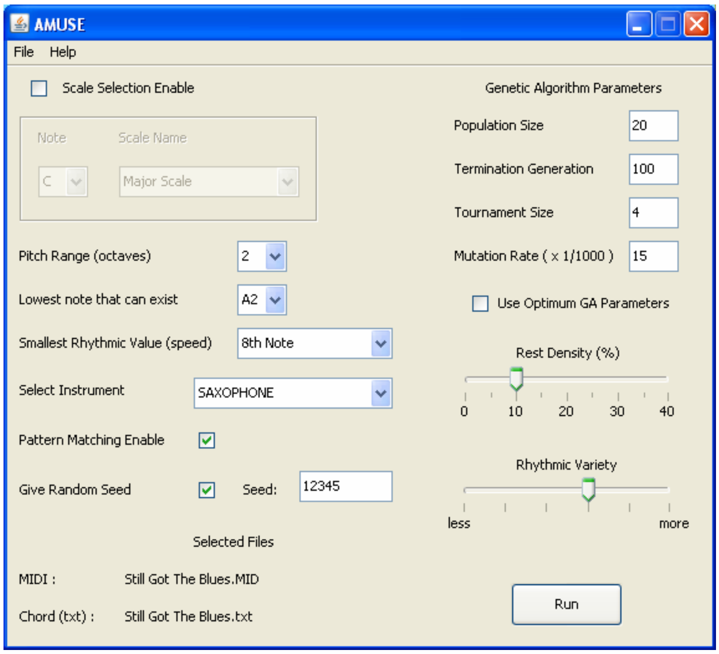
The suitability of GAs for musical composition is researched and explained by Jones et al. [2]. The opportunities for evolutionary music composition are presented by Brown in [3]. Probably one of the most famous software about the topic is *GenJam* by Biles [4]. It uses an interactive GA to generate jazz solos over a given chord progression. Jacob also used interactive GA to implement a composing system and did research on the algorithmic composition [7, 8]. Papadopoulos et al. [13] and Wiggins et al. [16] produced a similar work to Biles. Instead of user intervention for evaluation, a fitness function is utilized in their implementation. Amnuaisuk et al. studied harmonizing chorale melodies [14]. 21 different melodic features of a musical fitness function are presented in [15]. Johnson used an evolutionary algorithm to generate melodies in the style of church hymnody and presented important fitness objectives for this purpose [9].

In this paper, a creative GA is presented. It is embedded into a Java user interface as a musical expert, named as AMUSE for generating improvised melodies over a musical piece given a harmonic context. AMUSE integrates a modified representation scheme and different fitness objectives under a GA approach for generating melodies automatically without a human feedback. The evaluation of such a subjective work is not trivial. In this study, the implementation is evaluated with respect to the GA performance, influence of each objective within GA and the variance of the generated melodies from that of a human composer by a group of human listeners.

In the next section, an overview of the GA used is presented. The representation scheme is explained in detail with examples and descriptions are given on how the genetic operators work. In Section 3, the objectives are explained and illustrated with example cases. In Section 4, the results of three different public evaluations are presented and finally, the conclusions derived from those evaluations are discussed and the performance of AMUSE is assessed in different aspects.

2 A Genetic Algorithm for Generating a Melody

AMUSE (A MUSical Evolutionary assistant) is a graphical user interface with which a user controls the parameters of GA and the attributes of the melody or solo to be generated as illustrated in Fig. 1. AMUSE can be downloaded from the web address: http://cse.yeditepe.edu.tr/ARTI/projects/AMUSE. AMUSE allows a user to load the musical piece (substructure) for which an improvised melody will be generated in MIDI format (see http://www.midi.org for more).



Scale Selection Enable ☐

Note: C

Scale Name: Major Scale

Pitch Range (octaves): 2

Lowest note that can exist: A2

Smallest Rhythmic Value (speed): 8th Note

Select Instrument: SAXOPHONE

Pattern Matching Enable ✓

Give Random Seed ✓ Seed: 12345

Selected Files

MIDI : Still Got The Blues.MID

Chord (txt) : Still Got The Blues.txt

Genetic Algorithm Parameters

Population Size: 20

Termination Generation: 100

Tournament Size: 4

Mutation Rate (x 1/1000): 15

Use Optimum GA Parameters ☐

Rest Density (%): 0 10 20 30 40

Rhythmic Variety: less more

Fig. 1. A snapshot of AMUSE graphical user-interface

The GA requires a special representation scheme for generating melodies. The first scheme that comes to mind is having each gene to denote a musical note. Unfortunately, such a scheme is not sufficient alone, since it does not cover the duration of each note. This problem can be overcome by keeping two different allele values at each locus in a chromosome; a musical note and a rhythmic value. But, this produces another problem. The duration of each separate note can only be changed during mutation, limiting the variety of different rhythmic patterns. The representation scheme used in AMUSE addresses these issues.

The GA in AMUSE makes use of the traditional operators whose related parameters are chosen with respect to the chromosome length. One point and two point crossovers are both used with equal probability. One of them is employed to each selected individual pairs (mate) in the population. During the preliminary experiments, it is observed that this hybridized crossover performs better than applying just one type of crossover. All mates are chosen using tournament selection with a tour size of four. The population size is one third of the chromosome length. The offspring pool size is the same as the population size. The traditional trans-generational replacement scheme is used. The mutation operator in AMUSE randomly perturbs an allele within an allowed range using a mutation rate of 1/*chromosome-length*. Choice of default values for GA parameters is arbitrary and based on the previous studies in [10-12]. AMUSE allows the user to modify these default values.

2.1 Representation

In AMUSE, a chromosome (*individual*) encodes an improvised melody having the contribution of some other parameters. An *allele* (set of values a gene can have) value represents the order of a note in a given scale. The advantage of this representation scheme is that it is impossible to generate non-scale notes. For example, a ‘4’ indicates the 4th note in the corresponding scale, ‘0’ indicates a rest. The maximum allowed allele value indicates a *hold event* that makes the previous note to continue. Although the representation scheme is similar to the one used in *GenJam* [4], there are some differences. In *GenJam*, there are two different populations: *measure* and *phrase* populations. An individual in the measure population maps to a sequence of MIDI events. An individual in the phrase population maps to the indices of measures in the measure population. But, in AMUSE, a single population of individuals is maintained; each individual corresponds to a whole melody. Another difference is that; in our implementation, the range of the notes can be expanded or narrowed and the durations represented by each gene can be adjusted.

As stated before, some additional information is required to obtain the actual musical notes from the chromosome. For example, durations of the notes are needed. Those durations are derived from two values; one of them is the *hold event*. Hold events in a chromosome stretch the duration of the notes that are placed right before them in a chromosome. The other one is the *rhythmic value* that is defined the same for all genes. This value provides us the duration of each note identified by a gene (e.g. ‘Eighth Note’ or a ‘Whole Note’). In a way, the rhythmic value specifies the shortest note that can be heard in the melody. On the other hand, it is a parameter that increases (decreases) the search space size with the chromosome length when the duration represented by each gene is shortened (elongated). Such relevant information can be entered into the AMUSE as a user preference, providing more flexibility

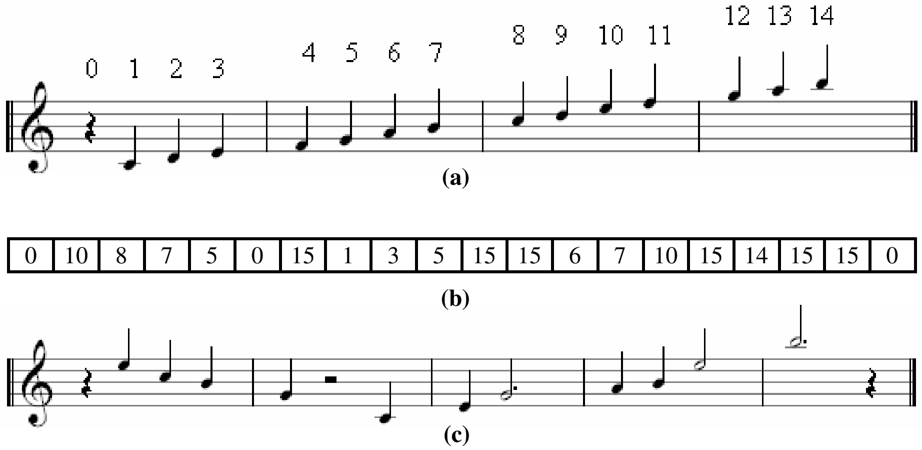


Fig. 2. **(a)** Gene values and the corresponding eighth notes in C Major Scale **(b)** An example chromosome, and **(c)** The actual notes that might be decoded from that chromosome

for the user over the melody. Also, the maximum allele value can be adjusted for expanding or narrowing the pitch range of the melody to be generated. Furthermore, the *beginning note* can be modified for shifting the pitch range to higher or lower octaves. This value is added to all of the alleles when creating the melody from the genotype and it takes the melody to higher or lower octaves.

As an example; assuming that the given scale is C Major, the rhythmic value of all notes is a quarter note, the maximum allowed allele value (hold event) is 15 (corresponds to a pitch range of two octaves for a seven-note scale) and the beginning note of the pitch range is A2 (the third lowest A note in a piano). According to these parameters, mapping of the gene values and musical notes are shown in Fig. 2 (a) and the example chromosome in Fig. 2 (b) is decoded as in Fig. 2 (c). But, sometimes the user might not want to generate a melody using just a single pitch and scale. Then, each gene will be mapped to a different scale and pitch. In such a case, the same mechanism can be still employed. Only, the values in a chromosome will be mapped to the actual notes based on different scales and pitches.

3 Objectives and the Fitness Function

One of the most important components of a GA for generating a pleasant melody is the fitness function. Fitness function can evaluate ten different objectives (Eq. 1). Some of these objectives are also used by Johnson in a different manner [9]. Note that some objectives might not be evaluated depending on the user's preferences. Eight objectives (f_1 to f_5 and f_8 to f_{10}) generate a value in [0,1) for a given candidate melody x that are equally weighted with w_i=1/|no.of_objectives_used|. Two of them (f_6 and f_7) check for some condition and apply punishment, generating a value in (-1,0]. Fitness value varies in the range (-2,1). A higher fitness value denotes a better melody.

$$f(x) = \sum_{i=1}^{10} w_i f_i(x) \tag{1}$$

Objectives can be divided into two groups; *core* and *adjustable* objectives. Core objectives can not be manipulated by the user, whereas the adjustable objectives are maintained according to the initial preferences of a user:

- **Core objectives:** Chord note (f_1), relationships between notes (f_2), directions of notes (f_3), beginning note (f_4), ending note (f_5), over fifth (f_6), drastic duration change (f_7)
- **Adjustable objectives:** Rest proportion (f_8), hold event proportion (f_9), pattern matching (f_{10})

Chord note objective checks whether an actual note decoded from a gene is in the notes of the chord of the same beat or not. The objective value denotes the percentage of such notes over all notes. This evaluation is provided only when a chord file is defined. Otherwise, only the scale is known and this objective is not considered.

Several different type of relationship between notes are analyzed and evaluated. The objective function for the relationships between notes returns a weighted sum of all sub-objective values. All consecutive notes are considered one by one. The number of such note groups is counted with respect to the category each of them falls, returning the sub-objective value. In order to get an overall evaluation of the objective, a scaling that will be referred as *overall scaling* is performed by dividing the total by the number of actual notes minus one. The sub-objectives are as follows:

- *One Step* (Fig. 3 (a)): Next note's scale degree is one step higher or lower than the previous note's degree. When the notes belong to different scales, it is checked whether the next note is one or two half steps higher or lower. The weight is 1.0.
- *Two Steps* (Fig. 3 (b)): Next note's scale degree is two steps higher or lower than the previous note's degree. When the notes belong to different scales, it is checked whether the next note is three or four steps higher or lower. The weight is 1.0.
- *Same Note* (Fig. 3 (c)): Next degree is the same as the previous degree. (0.9).
- *Three Steps* (Fig. 3 (d)): Next note's scale degree is three steps higher or lower than the previous note's degree. When the notes belong to different scales, it is checked whether the next note is five steps higher or lower. The weight is 0.8.
- *Four Steps* (Fig. 3 (e)): Next note's scale degree is four steps higher or lower than previous note's degree. When the notes belong to different scales, it is checked whether the next note is six or seven steps higher or lower. The weight is 0.7.

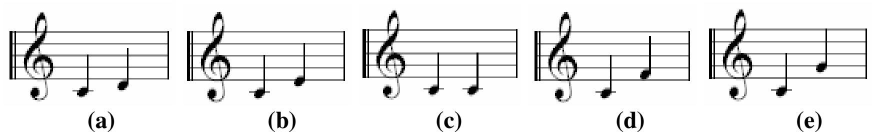


Fig. 3. Examples of relationships between notes in C major scale

When the tonal distance between two consecutive notes becomes smaller, it is more likely that they will sound better. If the tonal gap between notes becomes larger, then the notes should be carefully organized to provide a better sound. Therefore, higher weights are assigned for smaller tonal distances to get a melody progressing with small steps.

In music, *direction*, in other words, *contour* of the melody is also important. A melody might fall, rise or stay stable (Fig. 4). The objective function for the directions of notes evaluates three sub-objectives simultaneously where each objective counts the note triplets that fall into their category in a given melody. The note triplets represent all three actual consecutive notes in the melody. Finally, overall scaling is performed on the weighted sum of all sub-objective values by the number of actual notes minus two:

- *A Falling Melody* (Fig. 4 (a)): This sub-objective checks whether a given three consecutive actual notes form a falling melody or not. The weight is 1.0.
- *A Rising Melody* (Fig. 4 (b)): This sub-objective checks whether a given three consecutive actual notes form a rising melody or not. The weight is 1.0.
- *A Stable Melody* (Fig. 4 (c)): All three notes are the same. The weight is 0.9.

We do not desire the generated melodies to change direction up and down rapidly, because that sounds unpleasant in general. Instead, a smoothly flowing melody is desired that has a progress in one direction for a while. The weights of the falling and

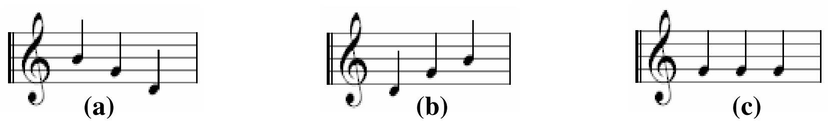


Fig. 4. Example of **(a)** a falling melody, **(b)** a rising melody, and **(c)** a stable melody

rising melodies are set to higher values as compared to the stable melodies to support these sub-objectives. Notice that the stable melodies are not ignored completely, since they might still generate a pleasant sound.

The beginning of a musical piece is important. It is like an introduction or a starting step for the song or music. This objective function returns 1.0 for a candidate melody, if it starts with the root note of the scale, since it is always harmonically correct and never disturbs the listener. Similarly, the ending is like a conclusion for a musical piece and the root note of the corresponding scale is again a good choice for an ending note, since, we hear all the notes in a song relative to the root. Due to this affect, the same notes in different songs raise different feelings. As a result, ending a melody with a base sound resolves the music comfortably. This objective function also returns 1.0 for a candidate melody, if it ends with the root note of the corresponding scale.

Over fifth objective is a complementary objective to the relationships between notes objective. All pairs of consecutive notes in a candidate melody are scanned. The objective function checks if the next note's scale degree is more than four steps higher or lower than the previous note's degree for each pair. Such pairs are counted and an overall scaling is performed by the number of actual notes minus one. The objective function punishes such tendencies that might generate unpleasant solo by returning a negative value of the scaled count. An example over fifth violation in C Major Scale is shown in Fig. 5 (a). A is the 6th note from C in C Major Scale.

Drastic duration changes between consecutive notes might sound disturbing. Hence, this objective function punishes such occurrences in a melody by checking the proportion of the duration of two consecutive notes. All pairs generating a proportion that is more than four are counted. After an overall scaling is performed by the number of actual notes minus one, the objective function returns a negated value. An example duration change violation is shown in Fig. 5 (b). C is a whole note and F is an eighth note in this example.

Rests can make the melodies more pleasant as if providing breaks in the melody. Rest proportion is the ratio of the total rest durations to the overall duration. User determines an expected value, denoted by *rpr* beforehand. Then, AMUSE attempts to arrange a melody carrying a rest proportion as close as possible to the *rpr*. For a

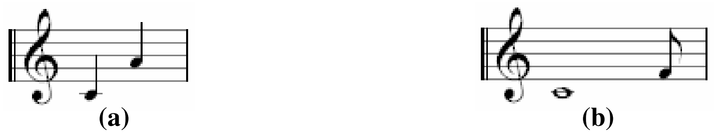


Fig. 5. Example of **(a)** an over fifth violation in C Major Scale, **(b)** a drastic duration change

candidate melody x, given that the rest proportion is denoted by $z = restPropOf(x)$, then the objective returns $f_8(x) = -200(restPropOf(x) - rpr)^2 + 1$, only if z is within ± 0.05 of the *rpr*. Otherwise, f_8 returns zero.

Every allele value receives the same rhythmic duration, determined by the user. Notes having different rhythmic durations enrich a melody. An allele value corresponding to a hold event makes the duration of a note longer. Hold event proportion is the ratio of the number of hold events to the chromosome length. Similar to the *rpr*, an expected hold event proportion can be defined by the user, denoted as *hpr*. For a candidate melody x, given that the rest proportion is denoted by $z = holdEventPropOf(x)$, then the objective returns $f_9(x) = -50(holdEventPropOf(x) - hpr)^2 + 1$, only if z is within ± 0.1 of the *hpr*. Otherwise, f_9 returns zero. The *hpr* and *rpr* provided by the user is also considered during initialization. Every initial individual is generated around the expected rest and hold event proportion.

A user might require having similar patterns in a melody. Choruses in the songs are good examples for such patterns in music. Pattern matching objective function searches for repeated patterns with 3 to 10 notes in a melody. For each note in comparison of a pattern; an award of 0.5 is given for the same notes with different durations, 1.0 for the same notes with the same durations. For the sample melodies in Fig. 6 (a) and Fig. 6 (b), the overall awards are 1.5 and 3.0, respectively. The maximum overall award occurs (Fig. 6 (c)) only if a melody repeats the same three note pattern. A scaling is performed on the overall award by this maximum possible value.

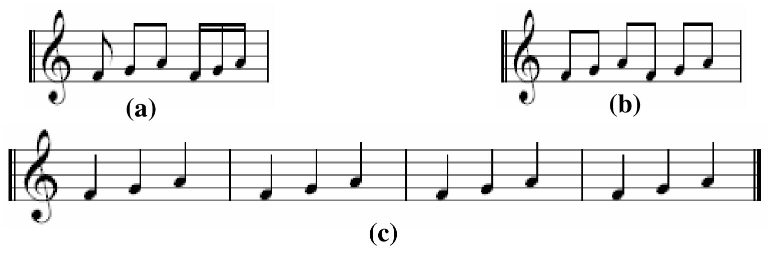


Fig. 6. Three note pattern: **(a)** same note, different duration, **(b)** same note and duration, **(c)** maximum possible overall award situation

4 Results

In this section, the results of a public evaluation are presented. This evaluation is completed by the help of 36 university students around the age of 20 from different schools and departments. It consists of three parts. The first part is a Turing Test (27 students). Two MIDI formatted music files are given to the participants. Both files share the same melodic substructure, but they carry different solos. One of the solos is generated by an amateur musician who has not listened to any solo generated by AMUSE before. The other solo is generated by the GA in AMUSE. The participants are asked to find the correct author of each solo in the musical pieces; human versus computer. The results in the first part show that, the participants cannot differentiate

Table 1. Evaluation results for the pairs of solos produced by AMUSE, where each pair uses the same substructure and one song in a pair is generated during the initial generation while the other during the 1000th generation. The letters in the song title denote the substructure.

Song Title	*V*1	*V*2	*W*1	*W*2	*X*1	*X*2	*Y*1	*Y*2	*Z*1	*Z*2
Fitness	0,92	0,30	0,29	0,94	0,94	0,27	0,94	0,24	0,17	0,90
id	*Rank of each song provided by each participant*									
1	10	7	1	2	9	8	5	6	4	3
2	10	7	8	9	5	3	6	4	1	2
3	2	1	3	4	8	7	10	9	6	5
4	2	1	3	4	10	5	8	6	7	9
5	9	7	1	6	8	3	10	2	4	5
6	3	4	9	10	7	8	1	2	5	6
7	10	9	6	7	8	5	2	1	4	3
8	9	10	3	4	5	6	7	8	1	2
9	3	1	2	4	9	8	10	7	5	6
10	10	4	3	6	9	2	8	1	7	5
11	6	5	3	7	4	2	8	4	1	3
12	3	4	6	5	8	9	2	1	10	7
13	9	3	6	7	8	2	10	4	1	5
14	10	6	3	4	9	5	8	1	2	8
15	4	3	6	5	8	7	9	10	1	2
16	2	1	4	3	10	9	6	5	8	7
17	7	2	5	8	10	1	9	3	4	6
18	4	1	7	8	10	2	9	3	5	6
19	9	6	8	7	10	3	5	4	1	2
20	9	8	1	3	7	2	10	5	4	6
Avr. Rank	6,55	4,50	4,40	5,65	8,10	4,85	7,15	4,30	4,05	4,90
Std.	3,27	2,89	2,48	2,18	1,77	2,72	2,87	2,72	2,67	2,13

the human work from the computer's work, because 52% of them provided wrong answers. This is a satisfying result, since our goal is not for AMUSE to beat an amateur musician, but to provide a matching performance.

The second part of the evaluation is arranged to observe whether the GA in AMUSE improves the initial population of melodies or not (20 students). At first, five different MIDI formatted music files are generated by using Band-in-a-Box software by PG Music Inc. (http://www.pgmusic.com). Then a pair of solos are obtained for each of them using the AMUSE. One of the solos is obtained from the initial population with a lower fitness, while the other one is obtained from the 1000th generation with a higher fitness. The rest of the parameters are kept the same during the runs. In this part, the participants listened to these ten solos in the given music files and ranked them as they like. 10 indicate the best solo, whereas 1 indicates the worst solo. The results are presented in Table 1. A solo with the higher fitness ranks as the first.

Notice the order of the average ranks of each pair of solos based on the same substructure having different fitness values in Table 1. Each solo with a higher fitness has a better rank than the other solo using the same substructure with a lower fitness. The results between the files that share the same substructure are compared to remove the subjectivity of the participants, because they all evaluate the songs according to their musical taste. Still, all solos having higher fitness rank better than the ones having lower fitness.

Table 2. Comparison of solos evaluated by different fitness functions, - indicates the excluded objective

Fitness Function	f	$-f_1$	$-f_2$	$-f_3$	$-f_6$	$-f_7$
id	*Rank of each solo provided by each participant*					
1	6	1	4	5	3	2
2	4	3	5	2	1	6
3	4	3	6	5	1	2
4	6	4	3	5	1	2
5	1	3	2	6	4	5
6	4	6	2	5	1	3
7	4	5	2	6	1	3
8	3	6	4	5	2	1
9	6	1	2	5	3	4
10	5	1	2	3	4	6
11	6	1	5	3	2	4
12	5	6	3	2	1	4
13	6	1	2	3	4	5
Avr. Rank	4.62	3.15	3.23	4.23	2.15	3.62
Std.	1.50	2.08	1.42	1.42	1.28	1.61

Furthermore, two-tailed sign test is used to investigate whether these rankings are statistically significant or not. Two related groups are compared according to the distribution similarity. The result of the sign test (‘p’ value) gives us the probability that those two groups are from the same underlying distribution. It is applied to the ranks obtained from the subjects for the song pairs. According to the results; almost all pairs (4 out of 5) have statistically significant performance variances within different confidence intervals. Among 20 subjects, V1:V2 and X1:X2 gave a result of 3 versus 17; W1:W2 and Y1:Y2 gave 4 versus 16; Z1:Z2 gave 6 versus 14 for the number of correct evaluations along with fitness values versus the number of wrong evaluations. As 3 versus 17 is a significant difference with significant level of 1% ($p<0.01$) and 4 versus 16 is also significantly different with a level of %5 ($p<0.05$). However, 6 versus 14 is not significantly different enough ($p\approx0.1$) according to the minimum accepted significant level of %5. On average, 'Z' substructure is the least liked one, hence, that might be the reason why 'Z' pair received less significance. Moreover, Z1 among all initially generated melodies and Z2 among all melodies that AMUSE improved in 1000 generations have the least fitness values.

In the third part of the evaluation, the relative importance of each objective composing the fitness function is investigated (13 students). Six different music files are generated by AMUSE using the same musical substructure. One of the files is generated using the fitness function that evaluates all objectives, while the rest of them exclude one of the objectives. The participants ranked each musical piece from one to six, similar to the previous part. As summarized in Table 2, the best solo having the highest average rank is the one that is generated by using the fitness function evaluating all the objectives as discussed in Section 3. Over fifth objective seems to be the most important one. Chord note, relationships between notes and duration change objectives follow it according to the order of importance from the highest to the lowest. The effect of the direction change objective on the generated melody might not be perceived by all the listeners and in some cases; the direction of the notes can be well-arranged even without using this objective. From the results of this part, it can be concluded that it is the least important objective.

5 Conclusions and Future Work

An automated evolutionary approach for generating melodies is discussed and evaluated in this paper. The proposed approach aims to fulfill two important requirements; generating melodies that are harmonically correct and sound pleasant to an ordinary listener. The first goal is achieved by making use of an effective representation mechanism based on the theory of the relationships between chords and scales. The second goal is achieved using an evaluation function that includes some fundamental objectives to push the search towards pleasing or at least non-disturbing melodies. The objectives assembled within the evaluation function are not mandatory rules in music. They might restrict the variety of melodies, but at the same time they reduce the risks of unpleasant melody generation.

GAs have already been used successfully in art. This study also showed that they can yield satisfying results in the field of music. The GA in AMUSE generates pleasant solos for a given substructure, that is comparable with an amateur human musician. An ordinary listener who has never listened to the solos generated by AMUSE before, cannot easily differentiate between an improvisation written by an amateur human composer and AMUSE. But, at this point, it is still not sufficient for an experienced composer to use AMUSE for practical purposes. In this form, it is more suitable for research and provides a basis to develop more sophisticated designs. To achieve better results and make the tool to be a practical one for users rather than a research tool, the fitness function may be rearranged. New fitness objectives can be added and new genetic operators can be embedded to cooperate with the fitness objectives. Current chord and scale mappings can be rearranged and new chord types can be added in order to expand the harmonic vocabulary.

References

1. Bentley, P.J., Corne, D.W.: Creative Evolutionary Systems. Morgan Kaufmann Publishers, San Francisco (2002)
2. Gartland-Jones, A., Copley, P.: The Suitability of Genetic Algorithms for Musical Composition. Contemporary Music Review 22(3), 43–55 (2003)
3. Brown, A.R.: Opportunities for Evolutionary Music Composition. In: Australasian Computer Music Conference, pp. 27–34. ACMA, Melbourne (2002)
4. Biles, J.A.: GenJam: A Genetic Algorithm for Generating Jazz Solos. In: Int. Computer Music Conf (ICMC 1994), Aarhus, Denmark, pp. 131–137 (1994)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (MA) (1989)
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. Mich. Press (1975)
7. Jacob, B.L.: Algorithmic Composition as a Model of Creativity, Organised Sound, vol. 1(3), pp. 157–165. Cambridge University Press, Cambridge (1996)
8. Jacob, B.L.: Composing With Genetic Algorithms. In: Proc. of the 1994 International Computer Music Conference, pp. 452–455 (1995)
9. Johnson, M., Tauritz, D.R., Wilkerson, R.: Evolutionary Computation Applied to Melody Generation. In: Proc. of the ANNIE 2004 (2004)
10. Ozcan, E.: An Empirical Investigation on Memes, Self-generation and Nurse Rostering. In: Proc. of the 6th International Conference on the Practice and Theory of Automated Timetabling, pp. 246–263 (2006)
11. Ozcan, E., Mohan, C.K.: Partial Shape Matching using Genetic Algorithms. Pattern Recognition Letters 18, 987–992 (1997)
12. Ozcan, E., Onbasioglu, E.: Memetic Algorithms for Parallel Code Optimization. International Journal of Parallel Programming 35(1), 33–61 (2007)
13. Papadopoulos, G., Wiggins, G.: A Genetic Algorithm for the Generation of Jazz Melodies. In: STeP 1998, Jyväskylä, Finland (1998), http://citeseer.ist.psu.edu/papadopoulos98genetic.html
14. Phon-Amnuaisuk, S., Tuson, A., Wiggins, G.: Evolving Musical Harmonization, The University of Edinburgh, Division of Informatics, Research Paper #904 (1998)
15. Towsey, M., Brown, A., Wright, S., Diederich, J.: Towards Melodic Extension Using Genetic Algorithms. Technology & Society 4(2), 54–65 (2001)
16. Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S., Tuson, A.: Evolutionary Methods for Musical Composition. In: Proc. of the CASYS 1998 Workshop on Anticipation, Music & Cognition (1998), http://citeseer.ist.psu.edu/13486.html

Unsupervised Learning of Echo State Networks: A Case Study in Artificial Embryogeny

Alexandre Devert, Nicolas Bredeche, and Marc Schoenauer

TAO team - INRIA Futurs - LRI/CNRS, Bat 490 - Université Paris-Sud - France

Abstract. Echo State Networks (ESN) have demonstrated their efficiency in supervised learning of time series: a ”reservoir” of neurons provide a set of dynamical systems that can be linearly combined to match the target dynamics, using a simple quadratic optimisation algorithm to tune the few free parameters. In an unsupervised learning context, however, another optimiser is needed. In this paper, an adaptive (1+1)-Evolution Strategy is used to optimise an ESN to tackle the ”flag” problem, a classical benchmark from multi-cellular artificial embryogeny: the genotype is the cell controller of a Continuous Cellular Automata, and the phenotype, the image that corresponds to the fixed-point of the resulting dynamical system, must match a given 2D pattern. This approach is able to provide excellent results with few evaluations, and favourably compares to that using the NEAT algorithm (a state-of-the-art neuro-evolution method) to evolve the cell controllers. Some characteristics of the fitness landscape of the ESN-based method are also investigated.

1 Introduction

Neural Networks (NNs) can be used to tackle a variety of problems. In classification or regression problems, some examples of inputs/outputs of the network are available during the learning phase: the training is *supervised*, and the fitness function is generally some Mean Square Error (MSE) between the network outputs and the actual outputs over the know examples. On the other hand, *unsupervised learning* regards cases where no such examples are available. When the network is used as a component of some computational model for a physical process, an explicit optimisation criterion (or *oracle*) is nevertheless available: the optimal network is the one for which the model reaches a target behaviour. Typical examples of such situation include control problems (e.g. robotics), and engineering inverse problems. Finally, the optimisation criterion can be implicit, and rely on some internal stabilisation of the network under external stimulation, as for instance in the case of Kohonen maps [24], where the task is to cluster pattern examples.

Two crucial programmer's decisions impact the choice of a learning method to train the network: the type of topology – feedforward or recurrent, i.e. without or with loops in the connection graph – and the choice of what will be learnt – the weights of a fixed topology, or both the topology and the weights.

For static problems, feedforward NNs will more likely be used, because the resulting learning problem is generally easier, while dynamical problems (where data k depends on data $k-1, k-2, \ldots$) will bias the choice toward recurrent NNs, that include some memory of the past in the activations of their internal neurons. However, feedforward NN fed with multiple past inputs can also capture the essence of dynamical systems, sometimes better than recurrent NNs [8].

Regarding the choice of the free parameters, very powerful methods are available to learn the weights of a fixed topology (e.g. for supervised learning, the back-propagation algorithm for feedforward [32] or recurrent [30] NNs), that would favour such approach. However, the choice of the topology itself then only relies on the programmer's expertise, and a poor guess will hinder the whole process. On the other hand, whereas learning both the weights and the topology opens up a much larger search space, the best topology can stay out of reach of the chosen learning method. The versatility and robustness of Evolutionary Algorithms make them perfect candidates for this latter task (see Section 2 for a brief survey).

Recently, however, an alternative to topology learning for recurrent NNs was proposed in the context of supervised learning of dynamical systems, namely the prediction of times series: the Echo State Network (ESN) approach [19] turns the search for the best topology into the search for the best combination of many randomly connected neurons: if sufficiently many different dynamics are present in the *reservoir* of neurons, then any dynamical system can be approximated by a linear combinations of those dynamics [29]. The learning problem is thus quadratic, and can be solved rapidly and efficiently by any standard optimisation method (more details in Section 3). However, this straightforward approach is limited to supervised learning, and very few attempts, if any, have tried to use ESNs in other contexts.

This paper proposes to use Evolutionary Computation to train an ESN in the context of unsupervised learning with explicit optimisation criterion, more precisely, to find the best controller of the cells of a multi-cellular approach to embryogenic design. As said above, training an ESN amounts to learning a vector of real parameters. However, because the context is unsupervised, no gradient-based approach applies. Moreover, because of the huge number of different dynamics that exist in the reservoir, it is expected that the fitness landscape will be quite rough. Hence Evolutionary Algorithms seem to be a good choice here. Unfortunately, an additional difficulty arises from the number of weights to learn: even though only the output weights are to be learnt (see again Section 3), the need for a large reservoir usually requires dozens or hundreds of internal neurons, resulting in as many weights per output of the network to be adjusted. Hence special care must be taken when choosing the optimisation method that will adjust those weights.

The paper is organised as follows : section 2 gives an overview on the state of the art of NN learning, focusing more precisely on evolutionary methods, and detailing in particular the NEAT algorithm [33], that constructively evolves the topology of a recurrent NN. Section 3 briefly introduces the Echo State

Networks and their (supervised) training before detailing the proposed approach for unsupervised optimisation of the weights of the ESN using an adaptive (1+1) Evolution Strategy. Section 4 describes the multi-cellular artificial embryogeny benchmark problem. Such problem has been addressed by the authors in a recent work using NEAT [9]. Those latter results are the baseline for the experimental validation of the proposed ESN-based approach, in section 5, where the fitness landscape is also studied in more detail. Finally, conclusions and further directions of research are given in section 6.

2 Artificial Evolution of Neural Network

The start of the Golden Age for Neural Networks was the invention of the back-propagation algorithm for supervised learning of feed-forward networks in the early 80's [32]. Several improvements have been proposed since then (e.g. variants of gradient methods for accelerated convergence, modification of the method for supervised learning of recurrent networks [30]). However, while theoretical results proved the representation power of such networks even with a single hidden layer [17], the issue of setting the correct number of hidden neurons for a given task remains open.

Some works addressed this issue by modifying the structure of the neural network prior to, or during, the learning process. Proposed approaches rely on hand-crafting the NN topology [25], pruning arcs or nodes from fully connected NN [26] or even growing NN by adding new nodes during the course of learning [11,2]. In this context, Evolutionary Algorithms (EAs) quickly appeared as a relevant approach towards NN learning (see [35] for a detailed survey). And though they can also be useful for the optimisation of the weights of a feed-forward NN, because backpropagation, as a gradient-based method, can easily get stuck in some local optima, EAs have been mainly used for their flexibility to handle complex search spaces: variation operators acting on the topology can easily be designed.

But evolutionary learning of NNs can be applied as soon as some performance measure for a given network is available, i.e. in both supervised or unsupervised (with an explicit optimisation criterion) context, or for feed-forward as well as for recurrent network topologies. Indeed, evolutionary learning of both the topology and the weights of recurrent NNs has been widely adopted in domains that could benefit from the wide variety of rich dynamical behaviours they offer, e.g. for control problems, in evolutionary robotics . . .

Notable works in this field include: the GNARL approach [1] which uses direct encoding of a neural network for building a robot controller; SANE [28] (Symbiotic Adaptive Neuro-Evolution), ESP [13] (Enforced Sub-population) evolve a population of neurons (rather than network) and combine these neurons to form effective neural networks; GASNET [18] combines optimising the position of neurons in an euclidean space with diffusion of chemicals; Gruau's Cellular Encoding [15,34] and his followers use indirect encoding, evolving a set of instructions that creates the network.

More recently, NEAT (Neuro Evolution of Augmenting Topologies) [33] and AGE (Analog Genetic Encoding) [10] have both been able to provide some relevant results, both in pure performance and speed of convergence, on classical benchmarks such as the double pole balancing. While AGE relies on an approach inspired from Genetic Regulatory Networks [5] where (regulatory) part of the genome encodes information on how to interpret (coding) parts, NEAT uses a direct encoding, as detailed in next section.

2.1 NEAT: Revolution of Augmenting Topologies

The *NEAT* algorithm [33], is an evolutionary neural network optimisation algorithm. It evolves both the topology and the weights of a neural network, either feedforward or recurrent. It relies on a direct encoding of neural network topologies and is based on a specific evolutionary scheme, using different subpopulations to preserver diversity along evolution. The main feature of NEAT is that it explores the topologies from the bottom-up: it starts from an empty network (direct connections from inputs to outputs), and proceeds constructively, using several mutation operators (and no crossover operator) to stochastically add neurons and connections to the networks while preserving as much as possible the behaviour of the network (e.g. new connections have very small weights, new neurons have no connections to start with, ...). Some Gaussian mutation operator modifies the weights so as to fine tune the network.

NEAT has been applied successfully to a wide range of problem, from the classical two pole balancing problem to particle systems rendering [16]. However, it should be noted that direct encoding methods poorly scale up, and sometimes have trouble to catch problem regularities (such as symmetries). Extensions of the original NEAT implementation have been proposed to tackle this issue, such as HyperNEAT [7], which has been successfully applied in an autonomous robotic context. Nevertheless, NEAT is currently one of the (if not "the") state-of-the-art NN optimisation algorithm and will be considered in this paper as the reference algorithm.

3 Echo State Networks

Echo state networks (ESN) have been proposed by Jaeger in 2001 [19,20] with the objective to endow a neural network with rich dynamics behavioural patterns while keeping learning complexity at a low level. An ESN is a discrete time, continuous state, recurrent neural network using a sigmoidal activation function for all neurons. A typical ESN is shown in figure 1, and will be used in this paper: the input layer is totally connected to the hidden layer, both the hidden and input layers are totally connected to the output layer. Moreover, the output layer is connected backward to the hidden layer. In this setup, the hidden layer, or *reservoir*, is randomly generated: N neurons are randomly connected up to a user-defined density of connections ρ. The weights of those connections are randomly set uniformly in $[-1, 1]$, and are scaled so that the spectral radius of

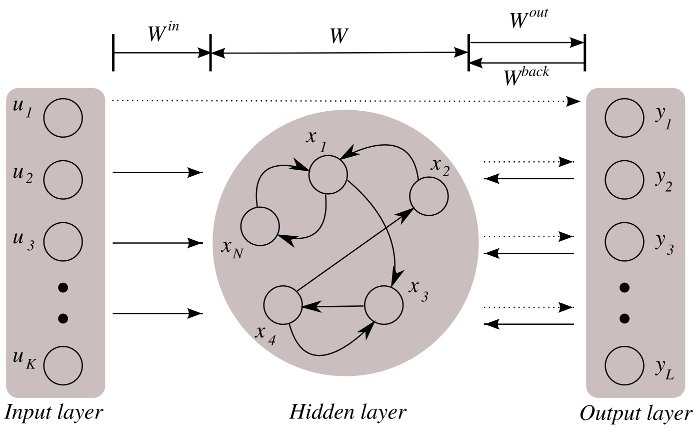


Fig. 1. Schematic view of an Echo State Network. Plain arrows stand for weights that are randomly chosen and remain fixed, while dashed arrows represent the weights to be optimised ($(K + N) \times N$) if N is the number of neurons in the reservoir.

the connection matrix is less than a given value $\alpha < 1$, ensuring that the network exhibits the "echo state property", i.e. stays out of the chaotic behaviour zone whatever the input sequence (see e.g. [21]). The random construction of an ESN is thus determined by the 3 parameters N, ρ and α.

The main point in ESN is that only the weights going from the input and hidden nodes to the output nodes are to be learnt. If the problem has K inputs and L outputs and a reservoir of size N, this amounts to $(K + N) \times L$ free parameters. Moreover, the learning problem is reduced to a quadratic optimisation problem that can be efficiently and quickly solved by any deterministic optimisation procedure, even for very large values of N. In some sense, an ESN can be seen as a universal dynamical system approximator, which linearly combines the elementary dynamics contained in the reservoir [29]. ESN have been shown to perform surprisingly well in the context of supervised learning, in particular for problems of prediction of times series, though it has also been successfully used in the context of (supervised) robot control learning (see [22] for an overview of ESN applications).

3.1 Unsupervised Learning of ESN

It is rather straightforward to replace the default learning algorithm of ESN with any derivative-free optimisers, such as Evolution Strategies or Simulated Annealing, to train an Echo State Network for supervised learning [4]. Yet, to date, and despite its intrinsic properties, ESN has not been applied to unsupervised problem with explicit criteria.

In order to address this class of problems, this unsupervised learning task is turned into an optimisation task: optimising an ESN amounts to optimising a real-valued vector representing the plastic weights of the network (from inputs and reservoir to outputs, see Figure 1). In such situation, Evolution Strategies

(ES) [31,6] provide an efficient and well-grounded framework. However, although only a limited amount of weights have to be optimised, the size of the reservoir may quickly lead to a high dimensional search space depending on the number of outputs, which impacts on the type of ES to be chosen.

This is why an adaptive $(1+1)-ES$ has been chosen here: this simple yet efficient ES scales up well with the dimension of the search space, in term of memory and computation time needed. In this algorithm, the "population" contains only a single individual X_t, that generates a single offspring Y_t using Gaussian mutation as follow :

$$Y_t = X_t + \sigma_t \mathcal{N}(0,1)$$

$\mathcal{N}(0,1)$ is a realisation of the normal probability distribution, and σ_t is the *mutation strength* at time t. The selection is deterministic: X_{t+1} is the best performing individual among $\{X_t, Y_t\}$

The idea behind this $(1+1)-ES$ is to adapt the σ value during the course of evolution with regards to the success of creating better offspring. Following [23], the σ value is updated according to the so-called *one-fifth rule*:

$$\sigma_{t+1} = \begin{cases} 2\sigma_t & \text{if } f(Y_t) < f(X_t) \\ 2^{-\frac{1}{4}}\sigma_t & \text{otherwise} \end{cases}$$

4 Multi-cellular Artificial Embryogeny

The model for Artificial Embryogeny considered here was originally proposed in [9]. It can be viewed as a continuous state discrete space and time cellular automata. *Cells* are placed on a two dimensional regular square grid (the whole grid is filled with cells, no cell division or migration is used). The state of each cell is a continuous value, representing here a grey level. The whole grid, or *organism*, can hence be interpreted as a grey image. Each cell communicates with its 4 neighbours by exchanging some "chemicals": each cell has an internal *controller* (a neural network) that determines its state as well as the amount of chemical it emits at time t toward its neighbours, according to the amount of chemical it received from its neighbours at previous time step $t-1$ (cells on the boundary of the grid don't receive anything from the external environment). Starting from a given state for all cells at time 0, this developmental process is repeated until some stopping condition is reached. The goal is here to reach a target 2D image when the development stops.

The original feature of the proposed model lies in the stopping criterion for the development: whereas previous works used a fixed number of development iterations, this model waits for the organism to stabilise (and penalises individuals whose organism doesn't stabilise within a prescribed number of iterations). The controller used in [9] was a neural network, evolved using the NEAT approach (described in section 2.1). Thanks to the stopping criterion, the evolved organisms exhibited very strong robustness to perturbation: the target image seemed to be the only fixed point of the best organisms considered as dynamical systems,

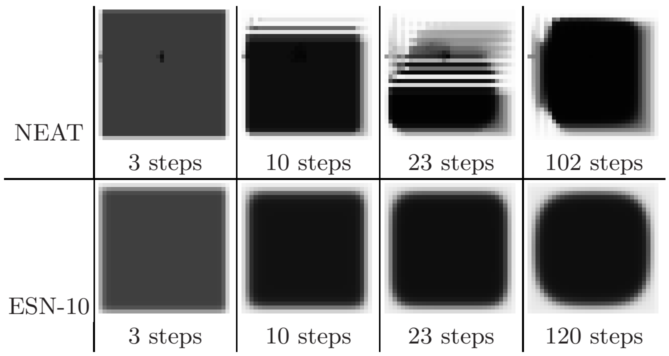


Fig. 2. Developmental stages for the *disc* problem. The right-most plots show the fixed-point image.

even though a single starting point was used during evolution. Figure 2 shows an example of a complete development of such result toward the fixed-point shape (the target shape was a black disc on white background). In the following, a maximum number of iterations of 1024 steps is fixed. If the organism has not reached a stable pattern before this limit, its fitness is set to the worst value.

4.1 The Flag Problems

In order to evaluate multi-cellular approaches, it is common to consider matching with simple geometric 2-dimensional images, like "flags" (French and Norwegian flags are quite popular in the literature [27,12]) or other regular patterns [14]). Figure 3 shows the two 32×32 target grey-level images used in this work, respectively called the disc and the half-disc.

The fitness function (to be minimised) is based on the MSE between two images. It takes value in $[0, 1]$, the optimal value is 0 when both images are identical:

$$s(A, B) = \frac{1}{wh} \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} (A(i,j) - B(i,j))^2$$

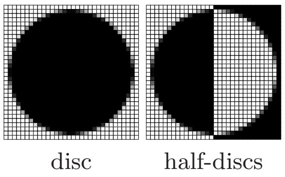


Fig. 3. The two target pictures (with grid lines)

5 Experimental Results

Three algorithms are compared on the two target flags for the embryogenic approach described in previous section: NEAT (results from [9]), and two Echo State Networks with different reservoir sizes.

5.1 Settings

The NEAT implementation used here includes the latest features from the literature. Our implementation has been validated with regards to the original results presented in [33]. NEAT explores recurrent topologies without constraints using a population of 500 individuals while all other parameters are set to default values (see [9] for a detailed information).

Two variants are tested for the Echo State Networks: reservoirs of sizes 10 and 50, with connection factor of respectively 50% and 10%. In both cases, the damping factor (spectral radius) was set to 0.9. These are refered to as 10- and 5-ESN respectively.

The settings for the $(1+1)-ES$ optimiser are as follows : σ_0 is set to 10^{-1}, and the starting point x_0 has all weights set to 0. The algorithm is stopped and restarted (with the same reservoir) whenever $\sigma_t < 10^{-8}$. In any case, the run is stopped when the total number of evaluations reaches 250000. Figure 4 displays the evolution of the fitness for a typical run: the restarts are clearly visible (note that it is a coincidence that the best fitness is reached after the final restart).

Note that the CPU cost of a single evaluation cannot be estimated alone, whatever the algorithm: it of course depends on the reservoir size for ESN, and on the (dynamic) number of neurons for NEAT, but also heavily on the number of developmental steps before stabilisation. Globally, the 16 ESN runs lasted around 2 days for reservoir size 10 and 5 days for reservoir size 50. In contrast, NEAT needed 7 days for the same experimental conditions.

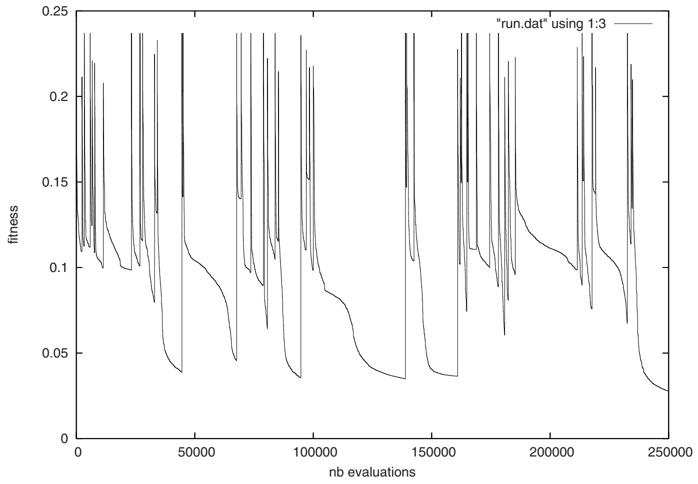


Fig. 4. A typical run for ESN with 10 neurons on the disc problem

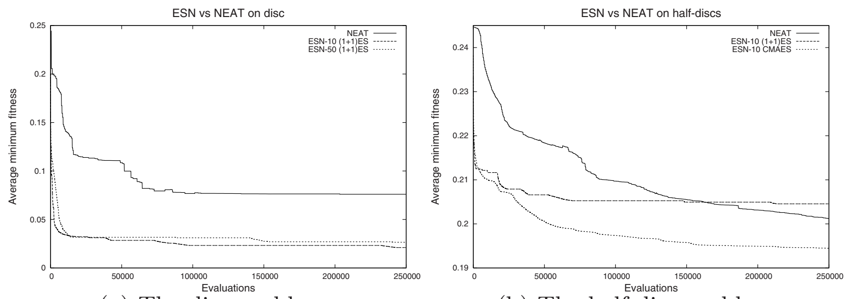
5.2 Results

On-line results of best-so-far fitness averaged over 16 independent runs are displayed on figures 5. Note that NEAT plot starts after the evaluation of the initial population (of size 500). The corresponding off-line results (i.e. after 250 000 evaluations) are detailed in Table 1.

It is clear that the ESNs outperform NEAT on the *disc* problem, and confirmed by a two-tailed t-test at 99% confidence level. Furthermore, though a bigger reservoir gives more parameters to optimise, it also makes the problem easier: the performances of ESN-10 and ESN-50 are not statistically distinguishable. An important remark is that the results of ESN are much more stable, as witnessed by the standard deviations in Table 1, one order of magnitude smaller for ESN (whatever the reservoir size) than for NEAT.

Table 1. Off-line results out of 16 runs: minimum – average(std. deviation)

	NEAT	10-ESN (1+1)ES	50-ESN (1+1)ES	10-ESN CMAES
Disc	0.076 – 0.105(0.135)	**0.021** – **0.030**(0.009)	0.027 – 0.033(0.008)	
Half-disc	**0.135** – 0.201(0.171)	0.205 – 0.207(0.002)	0.206 – 0.209(0.002)	0.184 – **0.194**(0.004)



(a) The disc problem (b) The half-disc problem

Fig. 5. On-line average minimal fitness reached by NEAT and both 10- and 50-ESN in 250000 evaluations, on the disc and half-disc problems with one chemical. Y-axes have different scales for both problems.

The picture is somewhat different for the *half-discs* problem. The first remark is that the best fitness is much worse for all algorithms than for the disc problem. Moreover, there is no statistically significant difference (whatever the confidence level in a 2-tailed t-test) among the 3 results. However, here again, the standard deviation among the NEAT runs is much larger than among the ESN runs . . . and this does make a difference here when considering the best fitness reached among the 16 runs: NEAT reaches 0.135 while no ESN run can find a better fitness than 0.206. Additional experiments are needed to give this some statistical significance. Nevertheless, this is a typical "design" situation where a large variance is a better indicator of possible good performance for equivalent averages.

5.3 The Fitness Landscape

In order to investigate the characteristics of the fitness landscape, projections on random directions of $\mathbb{R}^{(N+K)\times L}$ (see Figure 1) have been plot, both around the initial point of all optimisation, i.e. with all weights set to 0 (Figure 6) and the best point reached by one of the ESN-10 runs on the disc problem (Figure 7). Whereas the landscape around the initial point seems very smooth and almost convex, that around the final solution looks much more rough. In particular, there exist points very close to the solution that have the worst possible fitness value of 1, i.e. whose development never reaches a fixed point. This suggests that some gradient-based algorithm could be used at the beginning of evolution, but will rapidly stop being efficient when reaching lower-fitness regions, with rougher landscapes.

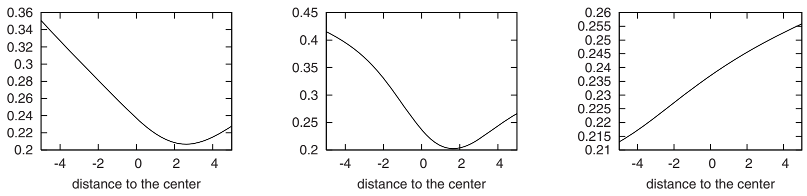


Fig. 6. Typical sections of fitness landscape for ESN-10 on the disc problem around the initial null point (centre of x-axis)

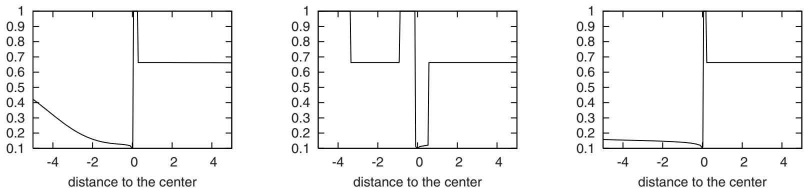


Fig. 7. Typical sections of fitness landscape for ESN-10 on the disc problem around the best individual of one of the successful runs (centre of the x-axis)

6 Conclusion

Echo State Networks are able to perform rich dynamic behavioural patterns with only few real-valued parameters to optimise. This makes ESN a very good choice for classification, regression and time-series prediction, even compared to more complex approaches such as NN weight and topology optimisation algorithms. Yet, to date, applications of ESN have been limited to supervised learning tasks.

This paper has introduced ESN in the context of an unsupervised learning task. The proposed approach combines ESN with a simple yet efficient Evolution Strategy algorithm (a $(1+1)-ES$ implementing the $1/5$ rule). Experiments conducted on two benchmark problems from Multi-Cellular Artificial Embryogeny have shown that the proposed approach is competitive with that using NEAT, a state-of-the-art Neural Network topology optimisation algorithm.

ESN clearly outperform NEAT in one of the two problems, have similar performance on the other, and converge much faster in both cases: this confirms both their ability to model complex dynamics and the possible gain due to optimising in a smaller search space. Furthermore, NEAT results have a much larger variance. Whereas this can be thought as a defect demonstrating some lack of robustness, it can also turn out to be an advantage when the average values are comparable, as in the second experiment, as it witnesses the ability for the algorithm to find some very good solution, though very rarely. A deeper statistical study is required to assess (or not) this property.

Further analysis showed that the fitness landscape is very smooth around the initial solution, which might explain the good results in terms of speed of minimisation obtained with such a simple optimiser. This also suggests to try other optimisation strategies, using, or at least starting with, gradient based method in the first steps. At the other end of the process, it seems that the landscape is rather rough close to the (local) optima reached on the flag problems, in part due to the non-stabilisation of the developmental dynamical systems very close to the solution. This in turn suggests to use an adaptive stopping criterion rather than a fixed user-defined number of iterations.

Finally, all experiments have been performed with default parameters, both for generating ESN and tuning the $(1+1)-ES$ optimiser. Results might possibly be improved with a fine tuning of these parameters. Future directions include optimising the meta-parameters implied in the generation of the ESN reservoir, as well as relying on more powerful Evolution Strategy algorithms, such as the state-of-the-art CMA-ES algorithm [3], whenever the size of the reservoir is small enough to make possible.

References

1. Angeline, P.J., Saunders, G.M., Pollack, J.P.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks 5(1), 54–65 (1994)
2. Ash, T.: Dynamic node creation in backpropagation networks. Connection Science 1(4), 365–375 (1989)
3. Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), pp. 1769–1776 (2005)
4. Babinec, S.: Evolutionary optimization methods in echo state networks. In: 6th Czech-Slovak Workshop on Cognition and Artificial Life (2006)
5. Banzhaf, W.: On the dynamics of an artificial regulatory network. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 217–227. Springer, Heidelberg (2003)
6. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Natural Computing: an international journal 1(1), 3–52 (2002)
7. D'Ambrosio, D.B., Stanley, K.O.: A novel generative encoding for exploiting neural network sensor and output geometry. In: Thierens, D., et al. (eds.) GECCO 2007, ACM Press, New York (2007)

8. Dematos, G., Boyd, M., Kermanshahi, B., Kohzadi, N., Kaastra, I.: Feedforward versus recurrent neural networks for forecasting monthly japanese yen exchange rates. Asia-Pacific Financial Markets 3(1), 59–75 (1996)
9. Devert, A., Bredeche, N., Schoenauer, M.: Robust multi-cellular developmental design. In: Thierens, D., et al. (eds.) GECCO 2007, ACM Press, New York (2007), http://hal.inria.fr/inria-00145336/en/
10. Durr, P., Mattiussi, C., Floreano, D.: Neuroevolution with analog genetic encoding. In: PPSN IX, pp. 671–680 (2006)
11. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems, Denver, vol. 2, pp. 524–532. Morgan Kaufmann, San Mateo (1990)
12. Federici, D., Downing, K.: Evolution and development of a multicellular organism: scalability, resilience, and neutral complexification. Artificial Life 12(3), 381–409 (2006)
13. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Technical Report AI96-248, 1 (1996)
14. Gordon, T.G.W., Bentley, P.J.: Bias and scalability in evolutionary development. In: GECCO 2005, pp. 83–90. ACM Press, New York (2005)
15. Gruau, F.: Genetic synthesis of modular neural networks. In: ICGA 1993, pp. 318–325. Morgan Kaufmann, San Francisco (1993)
16. Hastings, E., Guha, R., Stanley, K.O.: Neat particles: Design, representation, and animation of particle system effects. In: IEEE CIG 2007 (2007)
17. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366 (1989)
18. Husbands, P., Smith, T., Jakobi, N., O'Shea, M.: Better living through chemistry: Evolving gasnets for robot control. Connection Science 10(3-4), 185–210 (1998)
19. Jaeger, H.: The Echo State approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)
20. Jaeger, H.: Short term memory in echo state network. Technical Report GMD Report 152, German National Research Center for Information Technology (2001)
21. Jaeger, H.: Tutorial on training recurrent neural networks. Technical report, GMD Report 159, Fraunhofer Institute AIS (2002)
22. Jaeger, H., Haas, H., Principe, J.C. (eds.): NIPS 2006 Workshop on Echo State Networks and Liquid State Machines (2006)
23. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms: a comparative review. Natural Computing 3(1), 77–112 (2004)
24. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences, vol. 30. Springer, Berlin, Heidelberg (1997) (Second Extended Edition, 1997)
25. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541–551 (1989)
26. LeCun, Y., Denker, J.S., Solla, S., Howard, R.E., Jackel, L.D.: Optimal brain damage. In: Touretzky, D. (ed.) NIPS 1989, Morgan Kaufman, San Francisco (1990)
27. Miller, J.F.: Evolving a self-repairing, self-regulating, french flag organism. In: Deb, K., al., e. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 129–139. Springer, Heidelberg (2004)
28. Moriarty, D.E.: Symbiotic evolution of neural networks in sequential decision tasks. Technical Report AI97-257, 1 (1997)

29. Ozturk, M.C., Xu, D., Principe, J.C.: Analysis and design of echo state networks. Neural Computation 19(1), 111–138 (2007)
30. Pearlmutter, B.A.: Gradient calculations for dynamic recurrent neural networks: A survey. IEEE Transactions on Neural Networks 6, 1212–1228 (1995)
31. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
32. Rumelhart, D.E., Hinton, G.E., McClelland, J.L.: Exploration in Parallel Distributed Processing. MIT Press, Cambridge (1988)
33. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
34. Whitley, D., Gruau, F., Pyeatt, L.: Cellular encoding applied to neurocontrol. In: ICGA 1995, pp. 460–467. Morgan Kaufman, San Francisco (1995)
35. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)

Enhanced Genetic Algorithm with Guarantee of Feasibility for the Unit Commitment Problem

Guillaume Sandou[1], Stéphane Font[1], Sihem Tebbani[1],
Arnaud Hiret[2], and Christian Mondon[2]

[1] Supelec, Automatic Control Department, 3 rue Joliot Curie
F-91192 Gif-sur-Yvette, France
{Guillaume.Sandou,Stephane.Font,Sihem.Tebbani}@supelec.fr
[2] EDF Recherche et Développement, 6 quai Watier, F-78401 Chatou, France
{Arnaud.Hiret,Christian.Mondon}@edf.fr

Abstract. In this paper, an enhanced genetic algorithm for the Unit Commitment problem is presented. This problem is known to be a large scale, mixed integer programming problem for which exact solution is highly intractable. Thus, a metaheuristic based method has to be used to compute a very often suitable solution. The main idea of the proposed enhanced genetic algorithm is to use a priori knowledge of the system to design new genetic operators so as to increase the convergence rate. Further, a suitable penalty criterion is defined to explicitly deal with numerous constraints of the problem and to guarantee the feasibility of the solution. The method is also hybridized with an exact solution algorithm, which aims to compute real variables from integer variables. Finally, results show that the enhanced genetic algorithm leads to the tractable computation of a satisfying solution for large scale Unit Commitment problems.

1 Introduction

The Unit Commitment problem is a classical mixed integer optimisation problem for Power Systems. It refers to the computation of the optimal scheduling of several production units while satisfying consumer's demand and technical constraints of production units. Integer variables are the on/off status of production units, while real variables are the amount of energy they produce. Because of temporal coupling implied by constraints such as time up or time down constraints, a large temporal horizon has to be considered, leading to a large number of binary variables. Numerous methods have already been applied to get a suitable solution. They are for example listed in [1].

An exact solution method can be used: exhaustive enumeration, "Branch and Bound" [2] or dynamic programming [3] have been applied, but these methods suffer from combinatorial complexity. Furthermore, some of temporal constraints may be difficult to express in a suitable frame for those methods. As a result, approximated methods are required to compute near optimal solutions with low computation times, especially for large scale systems.

Deterministic approximated methods such as priority lists can be used [4]. However, these methods can be strongly suboptimal, as once again constraints can sometimes be hardly taken into account. Constraints are explicitly considered by Lagrangian relaxation [5]. Coupling constraints such as consumers' demand fulfilling are first relaxed. Thus, the unit Commitment problem is divided into several smaller optimisation problems (one per production unit), each of them being easier solved by considering dual problems. However, because of the non convexity of objective function, a duality gap may occur. Furthermore, no guarantee can be given on the actual optimality of the solution. An iterative organisation of the algorithm has to be used: solution of the optimisation problems considering fixed Lagrange multipliers, updating of these multipliers, and so on. This updating can be performed with genetic algorithms [6] or by subgradient methods [7].

Stochastic approximated methods are a class of interesting methods which have been intensively used for Unit Commitment. For example, a simulated annealing approach is used in [8], tabu search is used in [9], genetic algorithms are used in [10] and ant colony is used in [11]. With such methods, there is no guarantee on the actual optimality of the solution, but, one can often get a very suitable solution with low computation times. In this paper, an enhanced genetic algorithm is developed for the Unit Commitment solution. By defining new knowledge based genetic algorithm, the convergence time can be decreased. Furthermore, by choosing a suitable criterion and an elitist strategy, the feasibility of the final solution can be guaranteed. The algorithm is also hybridized with an exact solution method which aims to compute real variables from integer variables.

In section 2, the Unit Commitment problem is briefly called up. The enhanced genetic algorithm is depicted in section 3. Exact real solution and new a priori knowledge based operators are presented, together with the feasibility criterion. Numerical results are given in section 4 for academic cases and large scale cases, up to 100 unit cases. Finally, conclusions and forthcoming works are drawn in section 5.

2 Unit Commitment Problem Statement

Unit Commitment is a very classical optimisation problem, stated as follows:

$$\min_{\{u_n^k, Q_n^k\}} \sum_{n=1}^{N} \left(\sum_{k=1}^{K} (c_{prod}^k(Q_n^k, u_n^k) + c_{on/off}^k(u_n^k, u_{n-1}^k)) \right) \tag{1}$$

With :

- N: length of time horizon,
- K: number of production units,
- u_n^k : on/off status of unit k during time interval n (binary variable),
- Q_n^k : energy produced by unit k during time interval n (real variable).

Production costs of unit k during time interval n are:

$$c_{prod}^k(Q_n^k, u_n^k) = (\alpha_1^k Q_n^k + \alpha_0^k) u_n^k = \alpha_1^k Q_n^k u_n^k + \alpha_0^k u_n^k \tag{2}$$

Start-up and start-down costs can be expressed by:

$$c_{on/off}^{k}(u_n^k, u_{n-1}^k) = c_{on}^k\, u_n^k (1 - u_{n-1}^k) + c_{off}^k\, u_{n-1}^k (1 - u_n^k) \tag{3}$$

The following constraints have to be satisfied:

- Capacity constraints:

$$Q_{\min}^k u_n^k \le Q_n^k \le Q_{\max}^k u_n^k \tag{4}$$

- Satisfaction of consumer's demand Q_n^{dem}:

$$\sum_{k=1}^{K} Q_n^k \ge Q_n^{dem} \tag{5}$$

- Spinning reserves:

$$\sum_{k=1}^{K} Q_{\max}^k u_n^k \ge Q_n^{dem} + R_n \tag{6}$$

- Time-up constraints:

$$\begin{cases} u_{n-1}^k = 0 \\ u_n^k = 1 \end{cases} \Rightarrow \left(u_{n+1}^k = 1,\, u_{n+2}^k = 1, \cdots, u_{n+T_{up}^k-1}^k = 1 \right) \tag{7}$$

- Time-down constraints:

$$\begin{cases} u_{n-1}^k = 1 \\ u_n^k = 0 \end{cases} \Rightarrow \left(u_{n+1}^k = 0,\, u_{n+2}^k = 0, \cdots, u_{n+T_{down}^k-1}^k = 0 \right) \tag{8}$$

3 Enhanced Genetic Algorithm

3.1 Real Variable Computation

The problem is firstly reformulated in a fully integer programming problem framework. Consider that binary variables u_n^k, $\forall n \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,K\}$ are given and refer to a feasible solution: time-up and time-down constraints are fulfilled and spinning reserves (equation 6) and demand constraints are possible to satisfy. Then, real variables Q_n^k are computed from the following optimisation problem:

$$\arg\min_{\{Q_n^k\}} \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \left(c_{prod}^k (Q_n^k, u_n^k) + c_{on/off}^k (u_n^k, u_{n-1}^k) \right) \right) \tag{9}$$

This problem can be successively simplified:

$$\begin{aligned}
&\underset{\{Q_n^k\}}{\arg\min} \sum_{n=1}^{N}\left(\sum_{k=1}^{K}(c_{prod}^k(Q_n^k,u_n^k)+c_{on/off}^k(u_n^k,u_{n-1}^k))\right)\\
&=\underset{\{Q_n^k\}}{\arg\min}\left(\sum_{n=1}^{N}\sum_{k=1}^{K}(c_{prod}^k(Q_n^k,u_n^k))\right)\\
&=\underset{\{Q_n^k\}}{\arg\min}\left(\sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_1^k Q_n^k u_n^k+\alpha_0^k u_n^k\right)=\underset{\{Q_n^k\}}{\arg\min}\left(\sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_1^k Q_n^k u_n^k\right)
\end{aligned} \tag{10}$$

The solution is supposed to be feasible. So there are no temporally coupling constraints anymore, and the problem is divided into N optimisation problems:

$$\underset{\{Q_n^k,k=1,\ldots,K\}}{\min}\left(\sum_{k=1}^{K}\alpha_1^k Q_n^k u_n^k\right) \text{ subject to } \begin{cases} \sum_{k=1}^{K} Q_n^k \geq Q_n^{dem} \\ Q_{\min}^k u_n^k \leq Q_n^k \leq Q_{\max}^k u_n^k \end{cases} \tag{11}$$

Consider that production units are sorted, $\alpha_1^1 \leq \alpha_1^2 \leq \ldots \leq \alpha_1^K$, then the optimal solution of problem (10) is to produce as much as possible with low-cost units, while satisfying capacity constraints. It leads to the following recursive computation of real variables:

$$\begin{cases}
Q_n^1 = \min\left(\max\left(Q_n^{dem}-\sum_{i=2}^{K}Q_{\min}^i u_n^i, Q_{\min}^1\right), Q_{\max}^1\right)u_n^1\\
\vdots\\
Q_n^k = \min\left(\max\left(Q_n^{dem}-\sum_{i=1}^{k-1}Q_n^i-\sum_{i=k+1}^{K}Q_{\min}^i u_n^i, Q_{\min}^k\right), Q_{\max}^k\right)u_n^k\\
\vdots\\
Q_n^K = \min\left(\max\left(Q_n^{dem}-\sum_{i=1}^{K-1}Q_n^i, Q_{\min}^K\right), Q_{\max}^K\right)u_n^K
\end{cases} \tag{12}$$

The Unit Commitment problem (1) is thus a full integer programming problem, for which genetic algorithm is well suited.

3.2 Classical Genetic Algorithm

Genetic algorithm has emerged as a well-known and efficient metaheuristic method to solve integer optimisation problems. The general flow chart of this algorithm is given in fig. 1. The main idea is to make a population of potential solutions evolve so as to create new potential population by using stochastic (or "genetic") operators. Classical operators (crossing-over and mutation operators) are depicted in fig. 2.

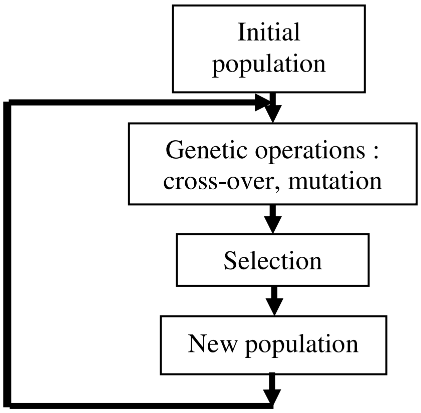


Fig. 1. General flow chart of a genetic algorithm

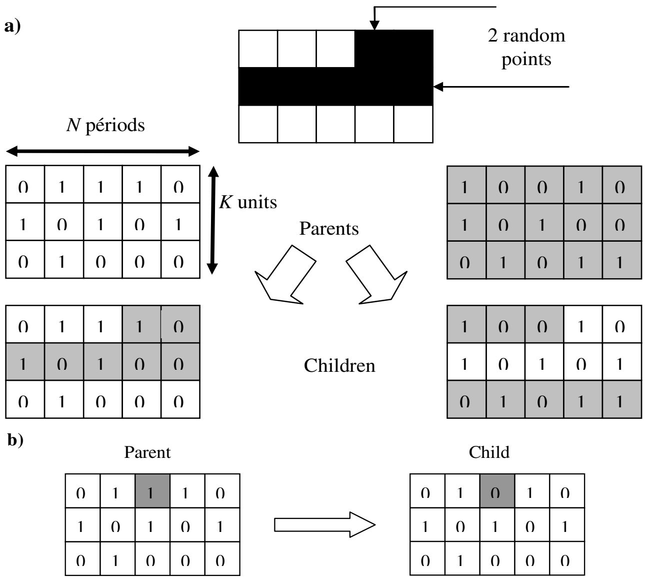


Fig. 2. Classical genetic operators a) Crossing over; b) mutation

For the genetic algorithm, individuals are represented as matrices of binary variables, which are $u_n^k,\ \forall n \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,K\}$ variables (see figure 2). Note that they are *NK* optimisation variables. For the crossing over operation, two potential solutions (labeled "parents") are randomly chosen in the population. They stochastically exchange their optimisation variables (or "genes") to create two new solutions ("children"). In fig. 2, the crossing over operation is a two point one. The mutation operation deals with the random selection of a solution and of one of its genes. This gene is changed to another. The aim is to keep the population genetic diversity by making new genes appear in the population.

Finally, the selection operator (see fig. 1) is a genetic operator which aims to choose a new population from parents and children. This operation is performed by the classical roulette wheel selection. After having computed the fitness value of each individual, the probabilities of selection are proportional to the quality of individuals.

3.3 Feasibility Criterion

To guarantee the feasibility of the solution, a suitable criterion is now defined. The idea is to compute penalty functions and to define an elitist strategy. Penalty functions have to be quickly computed as they will be calculated for all potential solutions in successive populations. As in [12], the following integer variables can be defined:

$$\begin{aligned} \delta_n^k &= u_n^k(1-u_{n-1}^k) \\ \varepsilon_n^k &= u_{n-1}^k(1-u_n^k) \end{aligned} \tag{13}$$

With the help of these variables, time-up and time-down constraints can be expressed by the following set of linear constraints:

$$\begin{gathered} \left(\delta_n^k = 1 \Rightarrow \left(u_{n+1}^k = 1, u_{n+2}^k = 1, \ldots, u_{n+T_{up}^k-1}^k = 1 \right)\right) \Leftrightarrow \sum_{j=0}^{T_{up}^k-1} u_{n+j}^k \geq T_{up}^k \delta_n^k \\ \left(\varepsilon_n^k = 1 \Rightarrow \left(u_{n+1}^k = 0, u_{n+2}^k = 0, \ldots, u_{n+T_{down}^k-1}^k = 0 \right)\right) \\ \Leftrightarrow \sum_{j=0}^{T_{down}^k-1} (1-u_{n+j}^k) \geq T_{down}^k \varepsilon_n^k \end{gathered} \tag{14}$$

Capacity constraints (4), consumers' demands satisfaction (5) and spinning reserves (6) have also linear expressions. Finally, constraints can be expressed by a linear global matrix equation, A_c and B_c collapsing all constraints:

$$\begin{gathered} \{u_n^k, Q_n^k; n = 1,\ldots,N; k = 1,\ldots,K\} \text{ is feasible} \\ \Updownarrow \\ \begin{cases} A_c x \leq B_c \\ x = (u_n^k, Q_n^k, \delta_n^k, \varepsilon_n^k; n = 1,\ldots,N; k = 1,\ldots,K)^T \end{cases} \end{gathered} \tag{15}$$

From this expression, a penalty function is quickly computed. To guarantee the feasibility of the final solution, the following criterion is defined for the algorithm.

$$\min_{\substack{\{u_n^k, Q_n^k\} \\ n=1,\cdots,N \\ k=1,\ldots,K}} \left(\begin{aligned} &\sum_{n=1}^{N}\sum_{k=1}^{K}\left(c_{prod}^k(Q_n^k, u_n^k) + c_{on/off}^k(u_n^k, u_{n-1}^k)\right) \\ &+ ((1+\varepsilon)\, c^f + h(\{u_n^k, Q_n^k\})).B(\{u_n^k, Q_n^k\}) \end{aligned} \right) \tag{16}$$

- ε is a small positive real.
- $h(\{u_n^k, Q_n^k\})$ is a penalty function for non feasible solutions $\{u_n^k, Q_n^k\}$.
- $B(\{u_n^k, Q_n^k\})$ is a boolean function (1 for non feasible solutions, 0 otherwise).
- c^f is the cost of a known feasible solution.

With such a criterion, any unfeasible solution will have a higher cost than the best feasible solution already known. Thus, the optimisation problem (17) can be solved with an unconstrained optimisation algorithm, and constraints will be implicitly taken into account. In this study, the chosen optimisation procedure is genetic algorithm. If the selection strategy is an elitist one (the best solution is always kept in the population by the selection operator), then the final solution necessary satisfies all the constraints of the Unit Commitment problem.

This feasible solution can be quickly computed from a basic priority list. Note that the high quality of this solution is not necessary for the algorithm as it is possible to change the criterion during the algorithm when better feasible solutions are known.

3.4 Knowledge Based Genetic Operators

Classical genetic operators may not be well suited to Unit Commitment problem. The a priori knowledge can be used to design new genetic operators. Such adapted operators will imply a faster convergence of the algorithm. Selective mutation, exchange and "all-on" and "all-off" operators are defined.

Selective mutation operator. Consider for instance the situation of fig. 3. Because of time down and time up constraints, a random mutation will very often lead to an infeasible solution. Switching times are particular points of the solution where a mutation has a higher probability to create a new feasible solution.

Thus, a selective mutation operator has been designed. The idea is to detect switching times, and to allow mutations only on these particular points. Note that there is no guarantee that the feasibility is achieved, but the "probability of feasibility" is higher.

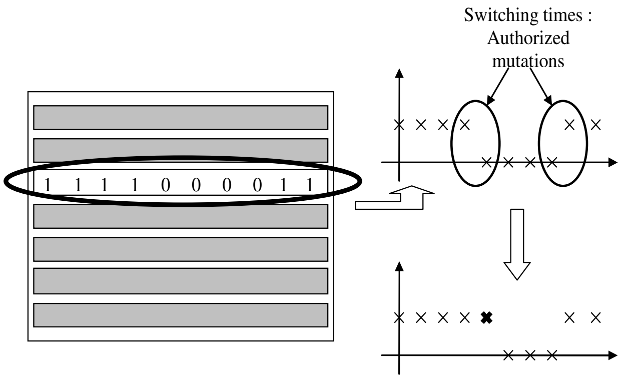


Fig. 3. Selective mutation operator

Exchange operator. As some production units are more profitable than others, it may be interesting to exchange their planning. The exchange operator randomly chooses a potential solution, two production units and two instants and exchanges the corresponding planning (see figure 4).

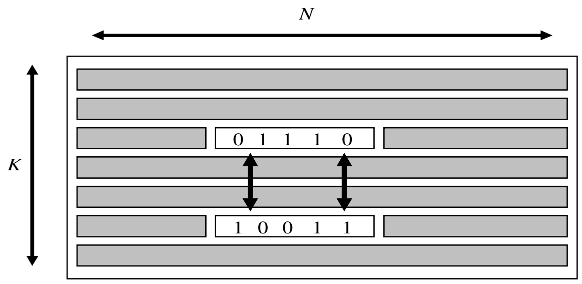


Fig. 4. Exchange operator

"All-on" and "all-off" operators. Finally, very simple but very important stochastic operators have been developed. Consider the situation of fig. 5. Because of time down constraints, mutation or even selective mutation will lead to infeasibility. The only way to reach the solution b) is to make two successive mutations, or to perform a lucky crossing-over, which is very improbable. That is why an "all-on" (and an "all-off") operator has been designed. The operator randomly chooses a solution, a production unit and two instants, and switch on or switch off the unit on the corresponding time interval. The goal of this operator is to increase the probability of going from a feasible solution to another by crossing the infeasible set.

4 Numerical Results

4.1 Academic Cases

The proposed algorithm has been tested with Matlab 6.5 (Pentium IV 2 GHz) for an academic case (4 unit example). As stochastic algorithms are considered, 100 tests are performed, and statistical data about the results are given. Optimisation horizon is 24 hours with a sampling time of one hour. The characteristics are those of table I.

It is assumed that at time 0, all units are switched off and can be switched on (time down constraints are satisfied). Consumer's demand Q_n^{dem} is depicted in figure 6a. This demand can be fulfilled by 2 production units (see 2 units limit in figure 6), except for hour number 9, for which a third unit has to be switched on. Because of time up constraints this unit will be switched on at least for 3 hours.

To be able to guess the optimal solution, null spinning reserves have been considered. As very low start up and start down costs have been chosen, the third unit will be switched off as soon as possible. From these physical arguments, one can guess the optimal solution, represented in figure 6b.

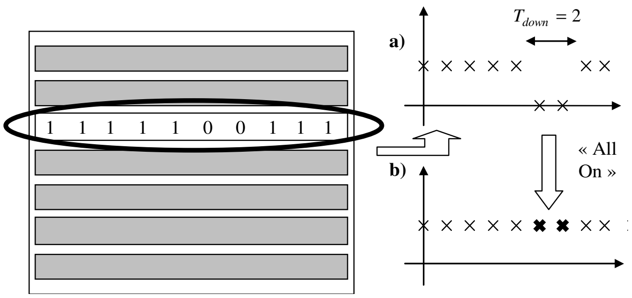


Fig. 5. "All on" operator

Table 1. Characteristics for the "4 units" case

Unit	$Q_{\min}$ MW	$Q_{\max}$ MW	α_0 (€)	α_1 (€/MWh)	c_{on} (€)	c_{off} (€)	T_{down} (h)	T_{up} (h)
1	10	40	25	2.6	10	2	2	4
2	10	40	25	7.9	10	2	2	4
3	10	40	25	13.1	10	2	3	3
4	10	40	25	18.3	10	2	3	3

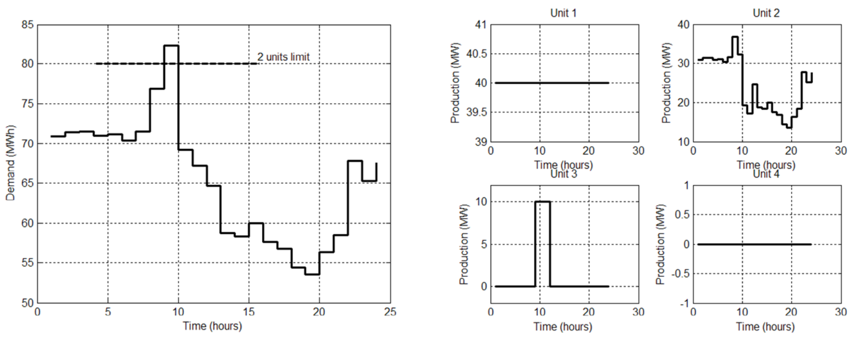


Fig. 6. Consumers' demand for the "4 unit" case and "4 unit" case optimal solution

The corresponding optimisation problem is made of 96 binary variables (24 hours and 4 units). Table 2 shows comparative results of optimisation. Statistical results are given: best case, mean, standard deviation σ, number of success (a test is successful if the known best solution is found by the algorithm) and computation times. Genetic algorithm is tested with a population of 50 or 100 individuals, and for 100, 200 and 300 generations.

Operator probabilities are set to:

- 70% for the crossing-over operator,
- 5% for the mutation operator,
- 10% for the selective mutation, the exchange, the all-on and the all-off operators.

Table 2. Optimisation results for the "4 unit" case with knowledge based operators

Case	Best	Mean	σ	Nb of success	Time
50 ind. 100 iter.	8778 € (+0%)	9050 € (+3.1%)	236 €	20	13 s
100 ind. 100 iter.	8778 € (+0%)	9006 € (+2.6%)	265 €	31	26 s
50 ind. 200 iter.	8778 € (+0%)	8824 € (+0.5%)	73 €	77	24 s
50 ind. 300 iter.	8778 € (+0%)	8788 € (+0.1%)	37 €	94	40 s

The choice of parents for genetic operations is always a fully random one, and does not depend on parent performances. Individual performances only influence the selection operator to create the new population. Results show that the enhanced algorithm is a very efficient algorithm as a very satisfying solution is found in a few seconds: for 50 individuals and 300 iterations, the best solution is found 94 times out of 100, in just 40 seconds. The mean result is just 0.1% higher than the best solution. It has been observed that the algorithm is not sensible to the tuning of operator probabilities. In order to study the influence of knowledge based operators, the same tests have been performed for a classical genetic algorithm, without these operators. Statistical results are given in table 3.

Table 3. Optimisation results for the "4 units" case without knowledge based operators

Case	Best	Mean	σ	Nb of success	Time
50 ind. 100 iter.	12 221 €	13272 €	466 €	0	12 s
100 ind. 100 iter.	10 798 €	12 838€	559 €	0	24 s

Results are much less satisfying with the classical algorithm than with the enhanced algorithm. To achieve as good results, the number of iterations has to be multiplied by 5 or 10, compared with the enhanced algorithm. It can be observed that the selective mutation is the most interesting operator. The knowledge based operators are not time consuming as it can be deduced from table 2 and 3. Indeed, the introduction of these operators only leads to a very slight increase in computation times.

4.2 Large Scale Cases

In this section, the proposed algorithm is applied to large scale cases with up to 100 units, with a time horizon of 24 hours. Statistical results are given for 10, 20, 40, 60, 80 and 100 unit cases. Small populations of 50 individuals have been considered. It is possible to choose particular characteristics for the problem so as to guess the optimal solution. Table 4 shows optimisation results for these large scale cases.

Table 4. Optimisation results for large scale cases

Nb units	Nb iter.	Best	Mean	σ	Time
10	500	29815 € (+0.07%)	30016 € (+0.74%)	185 €	188 s
20	500	$1.50\ 10^5$ € (+ 0.1%)	$1.51\ 10^5$ € (+ 0.9%)	710 €	466 s
40	1000	$5.56\ 10^5$ € (+ 0.07%)	$5.59\ 10^5$ € (+ 0.7%)	$1.76\ 10^3$ €	3100 s
60	1500	$1.21\ 10^6$ € (+ 0.21%)	$1.22\ 10^6$ € (+ 0.37%)	$3.45\ 10^3$ €	12000 s
80	2000	$2.14\ 10^6$ € (+ 0.32%)	$2.15\ 10^6$ € (+ 0.61%)	$6.83\ 10^3$ €	21000 s
100	2000	$3.34\ 10^6$ € (+ 0.85%)	$3.36\ 10^6$ € (+ 1.51%)	$1.50\ 10^4$ €	34000 s

Results of table 4 show that the algorithm leads to very satisfying solutions, with a relatively small number of iterations compared with the size of the optimisation problem. Computation times are not particularly good because the optimisation algorithm has been tested with Matlab which is not the best tool for this kind of applications. However, it can be seen that the required number of iterations is much smaller than classical genetic algorithm procedures: the number of iterations is divided by 3 to 5, compared with literature benchmarks. Thus, the enhanced algorithm with knowledge based operators has a great potential for large scale applications.

5 Conclusion

In this paper, an enhanced genetic algorithm has been presented. The main idea of the method is to use a priori knowledge of the problem to design new stochastic operators to increase the probability of creating new interesting solutions. The selective mutation, the all-on, the all-off and the exchange operators have been designed. Thus, the probability of reaching a feasible point in the search space is increased. A feasibility criterion has also been defined, leading to the guarantee of solution feasibility. Due to the use of relevant operators and criterion, the algorithm is very easy to tune, as it is not sensible to parameter modifications. Finally, it appears that the use of the operators leads to a decrease in the iteration numbers, with only a slight increase in iteration computation times. This could lead to a huge decrease in global computation times in the use of genetic algorithm, as shown for large scale cases: for a population of 50 individuals, only 2000 iterations are required to find suitable solutions (less than 1% from the optimal solution) for the 100 unit case. Forthcoming works deals with the generalization of the approach to non linear costs, the hybridization of genetic algorithm with local search or other stochastic algorithms such as ant colony, and the use of such algorithms for predictive control of hybrid systems.

Acknowledgments

This study is a part of the project *€nergie* of Supélec dealing with the optimisation of energy systems. This part is carried out with 'Electricité de France'.

References

[1] Sen, S., Kothari, D.P.: Optimal Thermal Generating Unit Commitment: a Review. Elec-tri-cal Power & Energy Systems 20, n° 7, 443–451 (1998)
[2] Chen, C.-L., Wang, S.-C.: Branch and Bound scheduling for thermal generating units. IEEE Trans. on Energy Conversion 8, n°2, 184–189 (1993)
[3] Ouyang, Z., Shahidehpour, S.M.: An intelligent dynamic programming for unit commitment application. IEEE Trans. on Power Systems 6, n°3, 1203–1209 (1991)
[4] Senjyu, T., Shimabukuro, K., Uezato, K., Funabashi, T.: A fast technique for Unit Commitment problem by extended priority list. IEEE Trans. on Power Systems 19, n° 4, 2119–2120 (2004)
[5] Zhai, Q., Guan, X.: Unit Commitment with identical units: successive subproblems solving method based on Lagrangian relaxation. IEEE Trans. on Power Systems 17, n°4, 1250–1257 (2002)
[6] Orero, S.O., Irving, M.R.: A combination of the genetic algorithm and Lagrangian relaxation decomposition techniques for the generating unit commitment problem. Electric Power Systems Research 43, 149–156 (1997)
[7] Dotzauer, E., Holmström, K., Ravn, H.F.: Optimal Unit Commitment and Economic Dispatch of Cogeneration Systems with a Storage. In: 13th PSCC 1999, pp. 738–744 (1999)
[8] Wong, Y.W.: An Enhanced Simulated Annealing Approach to Unit Commitment. Electrical Power & Energy Systems 20, n° 5, 359–368 (1998)
[9] Rajan, C.C.A., Mohan, M.R.: An evolutionary programming-based tabu search method for solving the unit commitment problem. IEEE Trans. on Power Systems 19, n°1, 577–585 (2004)
[10] Kasarlis, S.A., Bakirtzis, A.G., Petridis, V.: A genetic algorithm solution to the unit commitment problem. IEEE Trans. on Power Systems 11, n°1, 83–92 (1996)
[11] Sandou, G., Font, S., Tebbani, S., Hiret, A., Mondon, C.: Optimisation par colonie de fourmis d'un site de generation d'énergie. Journal Européen des Systèmes Automatisés 38, n°9/10, 1097–1119 (2004)
[12] Sandou, G., Font, S., Tebbani, S., Hiret, A., Mondon, C.: Short term optimisation of cogeneration systems considering heat and electricity demands. In: 15th PSCC 2005 (2005)

On the Design of Adaptive Control Strategies for Evolutionary Algorithms

Jorge Maturana and Frédéric Saubion

LERIA, Université d'Angers
2, Bd Lavoisier 49045 Angers (France)
{maturana,saubion}@info.univ-angers.fr

Abstract. This paper focuses on the design of control strategies for Evolutionary Algorithms. We propose a method to encapsulate multiple parameters, reducing control to only one criterion. This method allows to define generic control strategies independently from both the algorithm's operators and the problem to be solved. Three strategies are proposed and compared on a classical optimization problem, considering their generality and performance.

1 Introduction

Evolutionary Algorithms (EAs) [1] are metaheuristics inspired by natural evolution that are used to find sufficiently acceptable solutions to complex optimization problems. A set of candidate solutions, known as *population*, evolves by means of genetic operators. The two main operators are *mutation*, that modifies randomly an individual from the population, and *crossover*, that combines two of them. A *selection* process chooses the individuals that will survive in the next generation population. The whole process is repeated until a termination condition is satisfied. One of the strongest advantages of EAs over traditional optimization methods is their ability to escape from local optima. They have been successfully applied to various application domains.

Most of the time, the performance of these algorithms are strongly related to a suitable setting of several parameters such as population size and operator's application rate. The tuning of these parameters is difficult to achieve and often depends on empirical experiments or intuition. From the problem's resolution point of view, these parameters can be used to control the exploration of the search space and the exploitation of its interesting areas. If exploitation (also known as intensification of the search) is excessive, premature convergence may occur, while if exploration is too excessive (i.e., diversification), the algorithm becomes inefficient. The management of these two search strategies is indeed the central preoccupation of search (meta)heuristics.

Another important difficulty when using EAs to solve specific problems is the limited efficiency of generic evolution operators. Generally, the performance of an EA is also strongly related to the definition of efficient dedicated operators that take into account the structural properties of the considered problem (without neglecting the importance of encoding). These operators are often controlled by parameters, even in its most elemental way of application rate. The influence of those parameters over the Balance between Exploration and Exploitation (EEB) is a priori unknown, and knowledge about it is usually acquired across computationally expensive sets of experiments.

Control strategies often rely on specific rules to control a particular parameter. This makes it impossible to apply the acquired knowledge to algorithms with different operators: *knowledge is not exportable because it is not expressed in general terms*. It would be then interesting to generate control strategies w.r.t. EEB, which would allow us to encapsulate the complexity of handling specific parameters and define simpler and more general control schemes.

In this paper, we present a study about general control strategies based on a more global view of EA behavior. We first use a method to encapsulate the complexity of handling multiple parameters, even if they are associated to unknown-behavior operators [2]. This scheme focuses on a particular criterion: the population diversity. The population diversity and quality (i.e. mean fitness), produced by different combination of parameter's values, are measured during a training phase. Then, the combinations that provide maximal quality for different levels of diversity are identified and used later during the control phase. Genotypic diversity[1] is highly correlated with EEB: if exploitation is intensive, individuals will tend to concentrate in the higher fitness zones, so diversity will be low. On the other hand, if exploration is sparse, individuals will be dispersed in the search space, and diversity will increase.

Then, we propose several control strategies for managing diversity along the search process. By using diversity as the main controlling parameter, strategies can be expressed in more general terms of exploration and exploitation, common to all EAs. These control strategies are experimented and compared on a well-known combinatorial optimization problem: the quadratic assignment problem.

This paper is organized as follows. Sect. 2 summarizes relevant work, Sect. 3 provides an overview of our approach, while Sect. 4 analyzes several criteria to define general control strategies. Sect. 5 discusses experimental results, to finally draw conclusions in Sect. 6.

2 Relevant Related Work

2.1 Parameter Control

In [3], a broad number of approaches to control EA parameters have been reviewed and classified according to the taxonomy of Fig. 1.

Parameter setting strategies are divided in two main sets: those that fix parameters for the whole search *before the run*, and those that change their values *during the run*. In the first group the central task consists in finding fixed recommended values. In the second group parameter's values change during the run,

[1] Measured as the difference between individuals in the population. Since this approach can be applied to different problems, diversity must be defined accordingly.

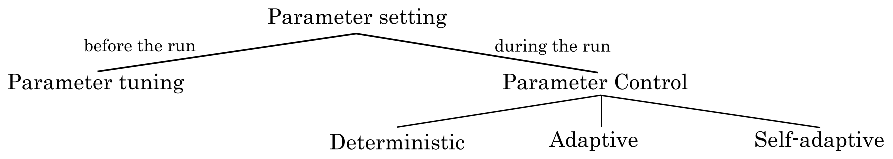


Fig. 1. Taxonomy of parameter control proposed by Eiben et al. [3]

those are divided according to how the adjustment is achieved: *Deterministic* control changes parameter's values by using deterministic rules, usually in relation with the number of elapsed generations. *Adaptive* control modifies parameters according to the current state of the search. Finally, parameter modification in *self-adaptive* control is performed by coding parameters inside individuals and make them evolve together.

Most studies focus on specific parameters control, with just a few exceptions. An adaptive genetic algorithm is presented in [4], where the relationship between state measures and parameters are encoded in control rules. In [5] a statistical method is used to measure relevance and to tune the parameters of an EA thanks to a second one. Two dynamic control strategies are compared in [6], where parameters are awarded according to their past performance.

Moreover, the integration of different fields of Artificial Intelligence has led to new kinds of control approaches. One of these approaches involves Fuzzy Logic (FL), where fuzzy rules are used to set parameter's values based on performance measures [7]. Our approach to handle parameters is based on this mixture, but applies FL not to control but to modeling behavior, while control is based on adaptive heuristics.

2.2 Fuzzy Logic Controllers

FL is an extension of classic boolean logic where levels of truth are expressed by a membership function with values ranging from 0 (false) to 1 (true). One of the most useful applications of FL are Fuzzy Logic Controllers (FLC) [8,9]. FLCs allows to infer answers from rules such as "IF car_speed is high AND road is dry, THEN risk is medium". Since FLCs are universal approximators of continuous functions [10], they act as modeling tools that express the output w.r.t. inputs. An early application was proposed by Wang and Mendel [11], in which many of the subsequent methods have been based.

3 Handling Multiple Parameters in EAs

3.1 Overview of the Approach

When faced to an EA, the user needs to understand its behavior in order to correctly tune its parameters and benefit from acceptable performances. Most of the time, a learning process (usually a long set of experiments using the algorithm with different parameters values) is not included in the algorithm but relies on the user's expertise and intuition. Then, she/he may apply her/his personal, and often empirical, control rules.

In a similar way, our approach is divided in two phases: *Learning* and *Control*. During *Learning*, the algorithm produces examples (EA's generations) using different parameter's values, to capture the mapping of these combinations with genotypic diversity. Since populations with similar levels of diversity may vary in terms of quality, another model is built, in order to link parameters and quality, measured in terms of mean fitness.

Both models are used to find the combinations of parameter's values that produce the higher quality populations corresponding to different values of diversity, which are obtained from a fine partition of reachable diversity. With this approach, the only parameter to modify thereafter is diversity.

During the *Learning* phase, three main problems arise: *dimensionality*, *inertia* and *noise*. Dimensionality is related to the fact that the amount of examples to be generated depends exponentially on the number of controlled parameters. Inertia is related to the resistance to the change of diversity and mean fitness values between consecutive generations. Here, we understand noise as the short-term variation product of random operators that induce inaccuracy in modeling.

During the *Control* phase, the controller changes diversity (and therefore parameters) in order to correctly exploit the search space and try to escape from local optima. It allows the user to express a generic strategy that can be applied to algorithms with different operators and solving different problems.

To provide an easy integration with any EA, the controller algorithm has been designed to be as independent as possible. Communication occurs as follows: the EA informs about current diversity and quality, and the controller assigns new values to controlled parameters, decides the reinitialization of population and the end of the search.

3.2 Learning Phase

There are 4 subphases in *Learning*: 1.*Example production*, in which examples are generated for every fuzzy partition combination; 2.*Modeling*, where diversity and quality FLCs are built, based on earlier collected examples; 3.*Refinement*, in which new examples, focused on the most promising areas, are generated to achieve a fine tuning of the model; and 4.*Releasing*, where all examples are used to build the definitive model, which will be used during *Control* phase.

The effects of noise and inertia are specially disturbing during *Example production*. Several techniques have been used to mitigate those effects. The first one aims to eliminate the influence of initial population by ignoring a number of generations at the beginning of the search.

Inertia has the undesirable effect of flattening measures of diversity and quality, as shown in Fig. 2.a (the surface should be continuous), where the examples of a each training grid cell (shown in the base of the plot), has been generated before passing to the next cell. Note that flattening occurs even in a close area, so it is advisable to use a training grid fine enough. We have defined a grid in function of fuzzy partitions of FLC's input variables. The intersection of

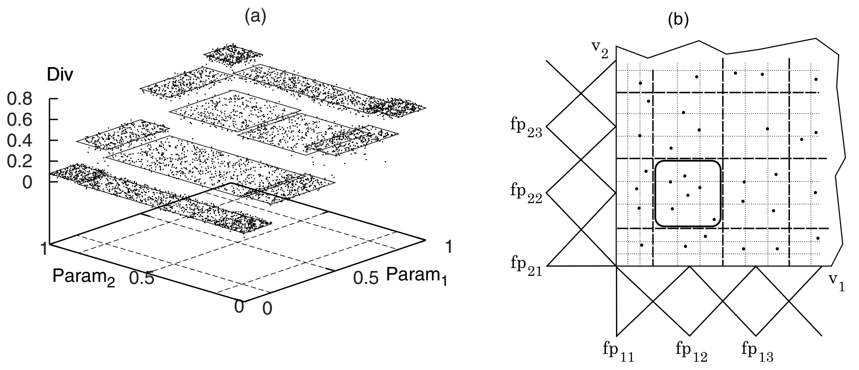


Fig. 2. (a) Formation of platforms (emphasized by squares) in a 4x4 coarse training grid, (b) influence area (fp_{12},fp_{22}) for two dimensions, in a partition with fineness of 3

all parameter's partitions define what we have called *influence areas*, that are subdivided by a factor of fineness (Fig. 2.b).

In order to avoid abrupt changes in parameters that would increase the undesirable effects of inertia, we have defined a visiting order called *smooth*, in which just one parameter value is modified in a minimal amount each time. Fig. 3 shows examples for 2 and 3 parameters in contrast with classical "nested loop" visiting order.

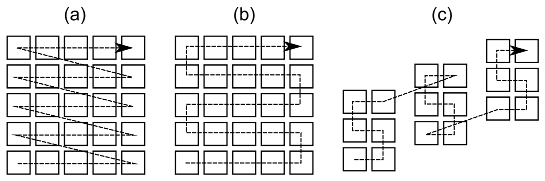


Fig. 3. Visiting orders: (a) classical nested loop, (b) smooth in 2D, (c) smooth in 3D

Once examples are collected, the *Modeling* subphase takes place. A Takagi-Sugeno FLC with polynomials of order 1 is used. To obtain the coefficients of the polynomials of the rule corresponding to an influence area, the algorithm performs a multiple linear regression of the examples collected to build FLCs in the sense *parameters* $\rightarrow$ *diversity/fitness*.

Fitness FLC is built similarly to diversity FLC, with the difference that an exponentially descending weighted average correction is done to consider the effects of long-term operators, like mutation. This method has also the advantage of reducing the noise. To cancel the bias produced by this correction, an even number of visiting runs are performed, shifting the direction every time.

Since controller requires the inverse, i.e. which parameters produce a given level of diversity with maximum quality, a dense grid of parameter combinations must be created to first find –using Diversity FLC– which ones have the required diversity, and then –using Fitness FLC– the one with the higher quality. Values of diversity and corresponding optimal parameters are stored in the so-called *cachedDiv* table.

In order to refine the model, a kind of "beta testing" is performed during the *Refinement* subphase, generating examples with the optimal parameter's values found, including a normal-distributed error. After that, during *Releasing* subphase, all generated examples are used to build the definitive model.

4 Control Strategies for EAs

By modeling diversity and quality w.r.t. parameter values, control strategies can be expressed closer to EEB than existing control methods. Therefore, the strategies could be applied to EA's that solve different problems with other operators. The issue is then to manage diversity in order to escape from local optima and to better exploit promising areas. This section discusses several criteria to design such strategy.

If an excessive diversity is allowed, it is likely that an excessive exploration will occur, without concentrating in the most interesting zones. On the other hand, if diversity falls to a small value, all individuals will tend to concentrate and will be trapped in a local optima. Additionally, if the latter happens, it would be difficult to reconstruct a population both diverse and of good quality, since all secondary optima must be found again. Therefore, an intermediate "correct" value of diversity should be maintained to have a good balance between exploration and exploitation, and mainly avoid the loss of diversity. Of course, all problems have different levels of "correct" diversity, so the algorithm must be able to identify it. A possible approach is to observe the fitness value of the better individual during the last generations. If the same value is often repeated, it is likely that the population is converging to the point corresponding to that fitness.

Another consideration is to alternate between stages of exploration and exploitation. Actually, during preliminary experiments, we have noted that there are some problems that were solved very easily with a simple zigzag between minimal and maximal diversities. It seems that it is sometimes necessary to "forget" what has been found to have the chance to explore a totally different zone of the search space and –perhaps– find a better solution. The question of how long, in terms of generations, should last this forgetting period is also another issue to consider: if it is too short the population will return to its initial position. On the other hand, if it is too long, the algorithm will loose computation time, although, since the parameters corresponding to maximal diversity are set to be quality-optimal, the risk of loosing the entire wealth of the population is much smaller than those when population is regenerated.

We have also experimented strategies with a small oscillation around the nominal level of diversity, that both performs a local exploration/exploitation and helps to stabilize the value of observed diversity in relation to commanded diversity. Another well-known consideration is to first explore and then exploit, in such way that the algorithm concentrates progressively in the most promising areas of the search space.

Tested Control Strategies. In order to compare control strategies, we have tested three different approaches that emphasize some of the aspects discussed earlier. Those strategies varies from maintaining diversity in a rather stable level to moving it abruptly during the search.

- **MX (Mixed):** it integrates first-explore-then-exploit, forgetting and small oscillation. A series of intermediate descending diversity levels: 0.7, 0.6, 0.5, 0.4, 0.3 and 0.2, in the range of achievable diversity, belonging to $[0, 1]$, are commanded to the EA, with an oscillation of 10% of this range, both above and below the nominal level. A period of 300 generations are executed at each level, which are extended in case of finding an historical improvement. After the algorithm has performed the descending steps of diversity, this one is raised to its maximum value, to escape from local optima, during 200 generations. After this, decreasing starts again.
- **CD (Correct Diversity):** it tries to reach an adequate diversity level. Every 10 generations, the fitness of the best individuals of the last 100 generations are considered. If more than 46 of them have the same value, commanded diversity is risen by 0.003, while if there are less than 17, diversity is lowered by 0.001. Those values where obtained from preliminary experiments. If the repeated value is not the higher one, or if the higher one has been obtained recently, the rising rule is relaxed a bit. Note that it is faster to rise diversity than to decrease it, what reflects the importance of avoiding premature convergence.
- **ZZ (ZigZag):** it implements a wide oscillation around a central value of diversity. This value is given by the mean of commanded diversities corresponding to the last five historic improvements. The oscillation, centered at this point, grows until the limits of possible diversity. If an historic improvement is reached, the amplitude of the oscillation is reset to zero, to start growing again.

Some control parameters were tuned to obtain reasonable results, even if our goal here is not to define the best heuristic for solving QAP but to compare several control approaches. However, it must be noted that different instances presented considerable differences among them, as we will see in the following sections. I order to provide more general and reliable control strategies over a wider set of benchmarks, a learning component could be studied in future research.

5 Experimentation

In this section, we describe the experiments we have carried out in order to compare the different control strategies described in section 4 [2]. We use an

[2] It is also possible to obtain fixed settings by taking parameters corresponding to a given diversity level in CachedDiv. Those settings normally produce worse performances than adaptive strategies, as shown in [2].

EA to solve the Quadratic Assignment Problem (QAP) with three operators, whose application rate parameters are controlled according to our method. Our purpose is not to be competitive w.r.t. state of the art solvers on this particular problem, but rather to compare performances of the previously presented control strategies.

5.1 Quadratic Assignment Problem

QAP is a well-known combinatorial optimization problem that can be stated as follows. Let us consider two matrices $A = (a_{ij})_{n\times n}$, $B = (b_{kl})_{n\times n}$, and a mapping function Π. The goal is to find a permutation $p_i = (\pi(1), \pi(2), \ldots, \pi(n))$ that minimizes:

$$f(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}$$

This problem was formulated by Koopmans and Beckmann [12] for a facility allocation problem, in which a set of n facilities with physical flows between them (matrix A) must be placed in n locations separated by known distances (matrix B). The goal is to minimize the cost $(flow \times distance)$ of overall system.

A set of 38 medium-size instances, obtained from the QAPLIB repository [3], was selected to test the algorithm, covering instances from all families.

5.2 Evolutionary Algorithm

The individuals are encoded as permutations. Population size is set to 100 individuals and three operators are applied: standard exchange mutation, that simply interchange two allocations randomly, cycle crossover [13], that preserves the absolute position of allocations from parents to descendants, and a specialized operator called *remake* that randomly erases four allocations, try the 4! possible reconstructions and chooses the best one. In order to focus on the general abstraction and control methodology, the selection process is not considered here as a mechanism to control diversity. Therefore, we choose a very basic selection scheme (roulette wheel) that correspond to a rather naive genetic algorithm implemented by a non specialist user. A set of 15 runs of 10.000 generations (learning not considered) have been performed for each instance and strategy. This great amount of generations was defined to observe how definitive is premature convergence in every case.

Diversity is defined as the sum of the differences of encoded variables between all population's individuals, and scaled linearly in [0,1] between minimal an maximal possible values, given the number of variables and the population size.

5.3 Learning Phase Parameters

During *Learning* phase, 2.000 generations were ignored at the beginning of *Example production* and *Refining* phases. Parameter's value range were divided

[3] http://www.seas.upenn.edu/qaplib/

into 4 fuzzy partitions and subdivided with fineness of 3. Within each partition of fineness, 5 generations were executed. During *Refining*, diversity descends and mounts linearly for 800 generations each one. In order to eliminate the effects of the modeling in the strategy comparison, 15 preliminary runs were made for each instance and only one cachedDiv has been chosen for each instance. The chosen cachedDiv was the one that presented the smaller deviation of observed diversity from commanded diversity during test runs.

5.4 Results and Discussion

Table 5.4 presents the mean values of cost and the standard deviation (in parenthesis) for each strategy and instance. An additional column shows the best known solution published in QAPLIB (April 2007). At the bottom of the table we have included the average number of runs in which the best known value was reached, and the number of instances where each strategy significatively[4] outperformed the others.

MX and CD have obtained the best results across different instance's families, with a slight advantage of CD. On the other hand, ZZ seems to be the least efficient strategy, with a couple of exceptions. However, the mean number of times in which the best known value have been reached is not much different for ZZ, but all successful results are concentrated on a few instances, while other control strategies seem to have a more regular behavior. Therefore MX and CD appear as more generic and could work properly on other problems. Some instances are notably easier that others (considering the operators used), since they were optimally solved by all strategies in every run, while others by none.

In order to analyze the characteristics of each strategy, we will concentrate on three representative instances that show the behavior of each one of the three strategies (Fig. 4). Relative ranking of the control scheme is indicated in each figure.

Considering CD on tai64c, we can see that, after an initial confusion due to the effect of initial population, the algorithm is able to rise diversity up to the level required by this particular instance. However, since the convergence-escaping heuristic of CD considers only one value of equally fitness-valued generations, it fails to detect excessive convergence when there are several values with fitness close to the local optima, as in ste36a and els19.

ZZ, nevertheless, solves els19 without any difficulty. This is partially accidental, since the starting diversity level agrees with the level required for this instance. Actually, placing the center of oscillation in the mean value of last successful improvements does not guarantee that this will be the right diversity to command. This appears clearly on tai64c. While CD lacks of means to detect sub-optimal multimodality, ZZ lacks of a criterion to well focusing on the right diversity level.

Note that CD and ZZ have opposite behavior w.r.t. diversity change : ZZ moves it continuously during the search, while CD stands quietly at a convenient value. Why CD solves better tai64c and ZZ does with els19 ? One explanation

[4] Using a t-Student test with significance level of 0.05.

Table 1. Mean and standard deviation of experimental results

Instance	MX	CD	ZZ	Best known
bur26a	5430603(2813)	5429382(2801)	5431717(1797)	5426670
bur26b	3820099(2645)	3820701(3237)	3819815(2497)	3817852
bur26g	10117735(637)	10118015(1805)	10118437(734)	10117172
bur26h	7098797(227)	7101812(11143)	7099036(314)	7098658
chr12a	9552(0)	9552(0)	9552(0)	9552
chr18b	1534(0)	1534(0)	1534(0)	1534
chr20c	14980(676)	15564(967)	14462(415)	14142
chr25a	4282(145)	4160(193)	4738(189)	3796
els19	17255411(64685)	17403382(428255)	17212548(0)	17212548
esc32a	133(1)	137(3)	146(4)	130
esc32b	174(8)	183(7)	190(3)	168
esc64a	116(0)	116(0)	116(0)	116
had12	1652(0)	1652(0)	1652(0)	1652
had20	6922(0)	6922(0)	6922(0)	6922
kra30a	90618(536)	90688(671)	91805(556)	88900
lipa20a	3696(21)	3690(12)	3695(21)	3683
lipa40b	504178(40403)	509671(41952)	558748(18977)	476581
lipa60a	108419(55)	108263(46)	108649(35)	107218
lipa60b	3001663(133444)	3005495(6122)	3068971(5125)	2520135
nug15	1150(0)	1150(0)	1150(0)	1150
nug20	2573(3)	2571(2)	2574(6)	2570
nug30	6177(26)	6156(21)	6261(25)	6124
rou20	729645(1575)	728832(1764)	729987(3362)	725522
scr20	110375(428)	110129(178)	110178(255)	110030
sko42	15982(62)	15962(48)	16341(61)	15812
sko64	49193(120)	49051(175)	50934(209)	48498
ste36a	9744(102)	9704(98)	10268(148)	9526
ste36b	16209(293)	16341(526)	16973(254)	15852
ste36c	8402215(76340)	8360860(84502)	8525723(69745)	8239110
tai20a	710633(2832)	709743(1618)	712817(2402)	703482
tai20b	122562707(222601)	122824782(270418)	122667094(266302)	122455319
tai40a	3249903(12230)	3231014(13179)	3304776(11237)	3139370
tai40b	647477626(8092672)	650707492(12312552)	654471314(8168843)	637250948
tai60a	7615573(33603)	7468530(19378)	7722640(22912)	7205962
tai60b	617635298(4993344)	616454128(4540470)	630041081(4689169)	608215054
tai64c	1857916(2066)	1856511(736)	1857830(2128)	1855928
tho40	244248(1349)	244027(1418)	251943(1598)	240516
wil50	49029(56)	48994(88)	49728(80)	48816
avg. optima	4.82	4.60	4.29	
outperf. MX	—	6	2	
outperf. CD	3	—	1	
outperf. ZZ	20	21	—	

could be related to the shape of fitness landscape of both instances: if it is smooth, a quiet search, that walks across the "plains" and the "valleys" could be appropriate, while if it is rugged, a "messy search", that first jumps between "peaks" to then concentrate of them could be best suited. The interest in finding

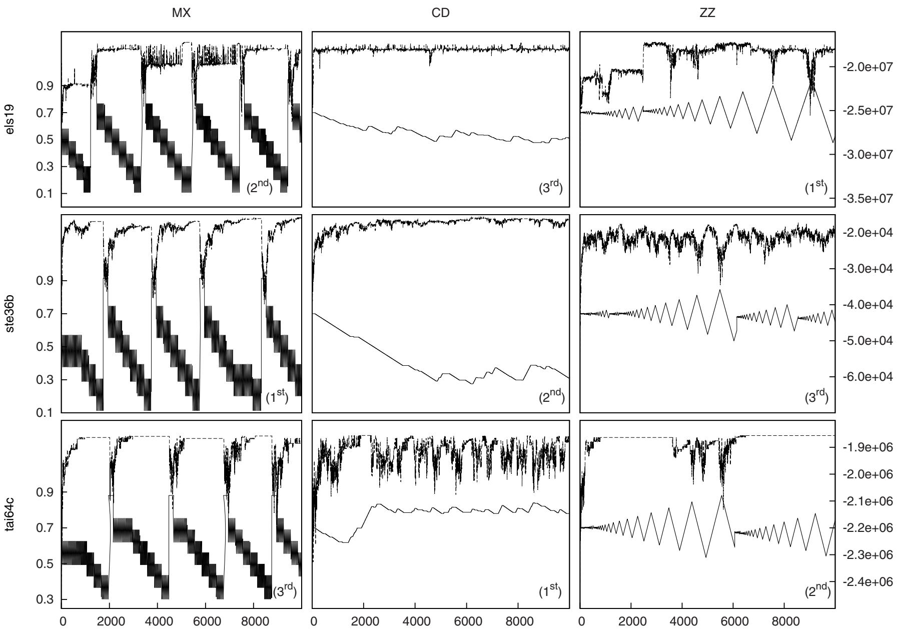


Fig. 4. Plot of commanded diversity (below) and fitness of best individual of the population (above) obtained with proposed strategies for representative instances els19, ste36b and tai64c. In parenthesis appears the relative order according to mean result.

out a relationship between fitness landscape and the best suited control strategy lies in the simplicity to know the former, thus it would be possible to automatically select the most performing strategy by measuring ruggedness at the beginning of the search.

In order to check this hypothesis, we have calculated the random walk correlation function, proposed by Weinberger [14]. This function takes a sequence of fitness values from a solution that is randomly modified by an operator, and calculates correlation between fitness values separated by s iterations. We have calculated the correlation for all treated instances with values of s ranging form 1 to 10, and with two operators; exchange mutation and remake, for series 50.000 iterations long. We have found that tai64c has, as we expected, a high level of correlation, revealing a smooth fitness landscape. Most of the instances with a similar correlation structure were best solved by CD (lipa60a, tai60a) and in some of them ZZ's performance was particularly inefficient (lipa60b, sko64, tai60b, wil50). On the other hand, els19 has one of the lowest measures of correlation, pointing out a rugged fitness landscape. The same happens with another instance well solved by ZZ, chr20c. However, the mapping between strategies and fitness landscape is not that clear in this case, since there are some rugged instances that are slightly better solved by CD (rou20, tai20a), and most them are solved similarly by all strategies (chr12a, chr18b, had12, had20, nug15, nug20, scr20). That could be caused by the inappropriate placing of center in ZZ. Further work is needed before concluding a definitive relationship.

Analyzing MX, we found that it worked exactly as expected when solving ste36b: Diversity descends progressively as fitness rises, and forgetting periods allows to escape from local optima to reach a better one. Even though these general-purpose diversity levels worked relatively well with most instances, they were not high enough to deal with tai64c. Another drawback of this strategy is that it does not consider the converging rate, i.e. even if fitness is rising firmly, when the time to forget is come, the algorithm starts to explore, losing the opportunity to improve the current best solution. The number of generations before entering a forgetting period is a sensible parameter, whose value depends on the problem. Something similar happens with the frequency of diversity change in ZZ.

6 Conclusion

In this paper we have presented a method to control multiple EA's parameters. Our purpose was to create an abstraction of algorithmic details in order to allow the definition of high-level control strategies, applicable to a wide range of EAs, regardless of the operators used and the problems being solved.

We have discussed several criteria that should be considered when defining general strategies, and three different schemes, all of them absolutely independent from EA's operators, have been studied. We have considered strategies: that keep a somewhat stable value of EEB, and others that continuously move this value.

An interesting relation between the performance of these strategies and the shape of the fitness landscape has been suggested. This could be used to automatically choose which strategy to apply when faced to a particular problem. A learning component could analyze performance of control strategies over different problems and parameters.

We have considered application rate parameters. It would be interesting to apply this method to other kind of parameters, such as selection pressure or population size.

Future work would also extend to other problems in order to assess the generality of our approach.

References

1. Michalewicz, Z.: Genetics Algorithms + Data Structures = Evolution Programs (1996)
2. Maturana, J., Saubion, F.: Towards a generic control strategy for EAs: an adaptive fuzzy-learning approach. In: IEEE Congress on Evolutionary Computation (CEC 2007), IEEE, Los Alamitos (2007)
3. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in EAs. In: Parameter Setting in EAs, pp. 19–46. Springer, Heidelberg (2007)

4. Kee, E., Airey, S., Cyre, W.: An adaptive genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 391–397 (2001)
5. Nannen, V., Eiben, A.: Relevance estimation and value calibration of EA parameters. In: Proc. of 20th Int.Joint Conf. on Artificial Intelligence IJCAI-2007, pp. 975–980 (2007)
6. Thierens, D.: Adaptive Strategies for Operator Allocation. In: Parameter Setting in EAs, Springer, Heidelberg (2007)
7. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: Current framework and new trends. Fuzzy Sets and Systems 141(1) (2004)
8. Kulkarni, A.: 3. In: Fuzzy Logic Fundamentals, pp. 61–103. Prentice-Hall, PTR (2001)
9. Piegat, A.: Fuzzy Modeling and Control. Springer, Heidelberg (2001)
10. Buckley, J.J.: Sugeno type controllers are universal controllers. Fuzzy Sets and Systems 53, 299–303 (1993)
11. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man and Cybernetics 22(6), 1414–1427 (1992)
12. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica 25, 53–76 (1957)
13. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the TSP. In: Proceedings of the 2nd Int.Conf.on GAs and their application, USA (1987)
14. Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics 63, 325–336 (1990)

Improvement of Intelligent Optimization by an Experience Feedback Approach

Paul Pitiot[1], Thierry Coudert[1], Laurent Geneste[1], and Claude Baron[2]

[1] Laboratoire Génie de Production, Ecole Nationale d'Ingénieurs de Tarbes, 47, av. d'Azereix BP 1629 - 65016 Tarbes, France

[2] Laboratoire d'Etude des Systèmes Informatiques et Automatique, INSA de Toulouse, 135, av. de Rangueil - 31077 Toulouse, France

{paul.pitiot,thierry.coudert,laurent.geneste}@enit.fr, claude.baron@insa-toulouse.fr

Abstract. Intelligent optimization is a domain of evolutionary computation that emerges since a few years. All the methods within this discipline are based on mechanisms for maintaining a set of individuals and, separately, a space of knowledge linked to the individuals. The aim is to make the individuals evolve to reach better solutions generation after generation using the knowledge linked to them. The idea proposed in this paper consists in using previous experiences in order to build the knowledge referential and then accelerate the search process. A method which allows reusing knowledge gained from experience feedback is proposed. This approach has been applied to the problem of selection of project scenario in a multi-objective context. An evolutionary algorithm has been modified in order to allow the reuse of capitalized knowledge. This knowledge is gathered in an influence diagram allowing its reuse by the algorithm.

Keywords: Project management, evolutionary algorithm, knowledge management, experience feedback, influence diagrams.

1 Introduction

The management of industrial projects is a more and more complex activity. The constraints to take into account are: multi-domain projects, uncertain and dynamic environment, innovative systems and/or components, multi-criteria optimization, etc. In this context, optimal methods are not suitable because of the complex and large search space. Several studies (see [1] for example) highlight the fact that combinatorial optimization techniques are in general relatively blind methods (i.e. they are not a priori guided). It is usual to launch an optimal search algorithm considering that the search space is uniformly interesting. This hypothesis is, in practice, often proved to be inappropriate. However, some techniques based on meta-models propose to gain knowledge (learning) during the resolution processes. They build a model of objective function (response surfaces, Kriging, etc.). Another alternative is to propose, during the exploration process, some interesting hypotheses of configuration and to use them as guidelines [2].

Nevertheless, being able to reuse the knowledge capitalized during previous optimization processes can be an interesting way to improve future explorations of the search space. The capitalized knowledge can provide interesting information for initial conditions and during exploration of search space. It is however necessary to adapt the knowledge capitalized during previous explorations to the current one.

In the domain of project management, the problem of scenario selection is very difficult, particularly when projects concern the design of new products. This context provides an interesting framework for knowledge reuse. Indeed, the reuse of components or knowledge when designing new systems is an increasingly important and strategic issue for companies. So a lot of information is already available, but not used to accelerate the optimization process. Moreover, the reuse process must take into account variations of environment to improve information used.

So, in this paper a framework for integration of an experience feedback process during optimization processes is proposed. The study is adapted to scenario selection in project management. In the next section, a state of the art about knowledge utilization to guide search methods is presented as well as the objectives of the proposed method based on experience feedback process. In section 3, knowledge acquisition process and model used by experience feedback process are described. Section 4 presents different ways to use knowledge during resolution before concluding and presenting prospects.

2 Knowledge as Guidelines for Optimization Methods: State of Art

The goal of combinatorial optimization techniques is to find a good solution, if possible optimal, according to a set of criteria, in the state space of input parameters (domains of combinatorial parameters). The method searches in this space combinations of parameters leading to interesting areas with respect to evaluation criteria. When the studied problem is too complex, two kinds of search methods are usually used: the meta-modeling of the objective function or the meta-heuristic methods.

The meta-modeling methods, such as neural networks, consider that the objective function, even complex, is coherent. Those methods try to build a regression model used to guide exploration procedure. One major problem with this type of method is the lack of explanations about obtained solutions. Knowledge is learned by the program and then stored in a model (e.g. weights on arcs between neurons) but it remains inaccessible for the user. Their advantage is the capacity of generalization which ensures a good reuse of knowledge on new cases.

Most heuristics and meta-heuristics methods assume that exploration process (local or global) will make it possible to find a good solution in a reasonable time and this by formulating assumptions on the data structure [3]. Heuristics methods are rules improving search of solutions for certain types of problems or for a particular aspect of a complex problem. Thus, they carry out a partial knowledge about a part of the problem or its structure. The search procedure is then accelerated but it is not enough to find a complete solution.

Meta-heuristics are more general models which must be adapted according to the problem to solve. For most of these methods and especially for Evolutionary Algorithms (EA), the method does not consist in spending time to capitalize knowledge about the problem which is too complex or changing. It rather consists in testing (quickly) a great number of possible solutions and to make sure that exploration process converges towards increasingly interesting solutions. This kind of method indirectly reuses knowledge associated with the problem via the evaluation of generated solutions. The assumption is made that carried out knowledge is linked to the quality of generated solutions. The method tries to improve these solutions. But knowledge, used during exploration, is not preserved from one execution to the other. Moreover, it is impossible to reuse only a part of it.

Recently, new methods, called "intelligent optimization methods" [2][4] suggest a coupling between a Model of Knowledge (MoK) about the problem to be solved and a traditional search method. The MoK must guide search towards promising zones while a traditional search method provides a "virtually contextualized information" to the MoK. For the majority of methods listed above, the use of knowledge is achieved indirectly. It is represented by means of classes of operators [5], intervals [6], assumptions on the parameters values [7] or by attributes about good solutions [8]. Works carried out by Chebel-Morello in [9] or Huyet in [4] propose to model the knowledge directly using parameters classes. Each class of parameters is more or less favorable to the different objectives. The problem is that it is very difficult to directly handle this knowledge with the employed methods (Knowledge Discovery in Databases - e.g. decision trees or neural network).

Finally, other methods [10] [11] use different kinds of Bayesian network as MoK. The MoK is learned from a database containing selected individuals from previous generation and it is used to generate directly the new population of individuals by sampling. The step of induction on the probability model constitutes the hardest task to perform and this task had to be performed for each generation.

Moreover, no study proposes a reuse mechanism of knowledge obtained during previous resolution processes to better solve new problems. Such reuse allows building and improving a complex MoK of the problem before optimization process and only using it during search (no knowledge actualization).

Objectives of our study. The proposed framework suggests to use a hybrid method including a meta-heuristic for search and a MoK to provide heuristics adapted to the current case. The MoK should not provide all information precisely but has to give some orientations with respect to a given situation. Michalski in [2] shows that fixing some interesting solutions properties is enough so that search method generates very quickly some solutions close to the optimal one. The system has to ensure the following properties:

1) The search process has to be efficient and to provide optimization even with an incomplete MoK or a failing one. In this study an Evolutionary Algorithm for the search process and an Influence Diagram as MoK are used;
2) Reuse and continuous improvement of operational knowledge has to be performed in an interactive manner (achieved projects) within an experience feedback process;

3) Reuse of the knowledge resulting from the simulations produced by the genetic algorithm. This is possible by means of a Knowledge Discovery in Databases (KDD) process for example;
4) Capacities of knowledge generalization that allow adapting knowledge according to new current cases. This adaptation has also to be performed in interaction with the meta-heuristic.

Application problem. In the domain of project management, the problem of scenario selection is considered. The aim of this application is to solve simultaneously the problem of selecting design alternatives for a system and the project planning problem to achieve this system. The constraints to be taken into account during the project planning are modeled by a project graph proposed in [12] and shown on figure 1. This model allows considering simultaneously the planning constraints and the design constraints. The project graph includes the tasks to be carried out, the AND nodes (parallel tasks) and the OR nodes representing the possible decisions during the design process called "design alternatives". Tasks are represented by means of rectangles with a task number, AND nodes by means of vertical double-bars, OR nodes by means of circles. The gray rectangles inside the tasks represent the different possibilities to carry out a task, called "task options". The selection of a path in the graph represents a potential scenario for the project as show on figure 1.

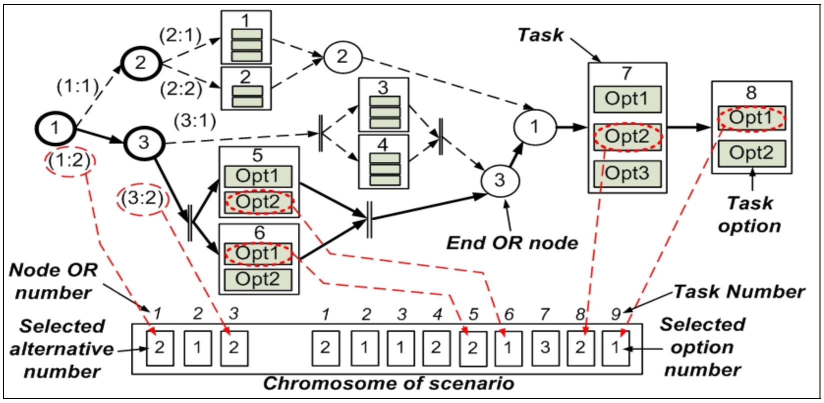


Fig. 1. Graphs of a project and scenario encoding; for example, this one concerns the realization of a pen with a scenario highlighted

Search method used in previous study. The search method is an evolutionary algorithm (EA) proposed by C. Baron in [12] for scenario selection. An EA is a meta-heuristic for stochastic optimization, used for global exploration. The SPEA method used in this study (Strength Pareto Evolutionary Algorithm) [13] is a traditional EA with classical steps (initialization, evaluation, selection) and genetics operators cross-over and mutation. It ensures the multi-objective evaluation of individuals according to the two following steps: i) the Taguchi approach is used in order to evaluate cost of a scenario for each criterion [14]; ii) multi-criteria evaluation is then achieved by means of Pareto front in order to compare and classify the scenarios (concept of dominance between solutions). The probability of selection for an individual is proportional to its fitness. The fitness depends on the position of the individual compared to the Pareto front (maximum fitness for the Pareto-optimal solutions). The fitness of an individual is calculated by formula (1) according to the strength (***S****ᵢ*) of individual which it dominates (Pareto-dominance). The strength of an individual is given by the formula (2) where ***n*** is the number of dominated solutions and ***Pop*** is the population size.

$$f_j = \frac{1}{1 + \sum_{i,\ i<j} s_i} \ . \tag{1}$$

$$s_i = \frac{n}{|Pop| + 1} \ . \tag{2}$$

After selection, cross-over operator is applied (selection of two "parents" and then crossing of their "genes"). Finally, the mutation operator is applied (selection of an individual and change of one or more genes). The criterion used for stopping this process is a strict limit of the number of generations.

EA requires an encoding of individuals. In this model, an individual represents one scenario for the project, i.e. an instantiation of the graph as shown on figure 1. The chromosome of an individual gathers on first part (the left one) the design alternatives choice. Each gene corresponds to a path choice in the project graph and it is represented in chromosome by a number corresponding to the selected alternative. Then on the second part of the chromosome (right one), each gene corresponds to selection of a task option (a number corresponding to selected option).

All possible choices are always represented whereas majority of them are inactive since they are inhibited by alternatives choices. For example, in figure 1, the first design choice (*OR node* ① *- choice of arc (2) noted (1:2)*) inhibits node ② and then, tasks 1 and 2. The second possible design choice for this scenario (*OR node* ③ *- choice of arc (2) noted (3:2)*) inhibits the tasks 3 and 4. Consequently, the tasks 1, 2, 3 and 4 are present in the chromosome but their genes are inhibited. This encoding ensures a constant viability of solutions but requires an additional control on genetics operators. A check has to be performed to avoid inefficient mutation or cross-over.

3 Knowledge Acquisition and Modeling

3.1 Knowledge and Experience Feedback

Knowledge management relies on expert's knowledge extraction and direct use of this knowledge through a modeling. Methods using this process (MKSM, Kads) encounter problems such as difficulty of data extraction, expert's availability or knowledge actualization. Experience feedback proposes acquisition and knowledge reuse through the experiences (spontaneous declaration of knowledge during their application). It rests mainly on two cycles of information management: capitalization and generalization. Capitalization is carried out by memorizing behavior of the expert.

Experiences are used as vectors to build knowledge reference frame. Each time that an event occurs, actors formalize their judgment. Indeed, to generalize a model of knowledge starting from lived experiments is easier than to clarify knowledge apart from its context.

In our study, knowledge corresponds to the identification of system and environment of system to be realized. It consists in probabilistic or determinist links between three state spaces: i) input parameters of the problem (i.e. genome where each gene corresponds to a design or planning decision); ii) the evaluation criteria and objectives (discrete values in our study); and iii) parameters characterizing the environment of the project. Indeed, context modeling is necessary to adapt knowledge to current situation. Knowledge about criteria and objectives is related to functionalities of the designed system (e.g. customer's requirement) or on requirements for the project management (e.g. minimize cost, delay, etc). Among knowledge about genome, two different kind of knowledge can be distinguished. The first one is a structural knowledge related to the problem to solve (e.g. the constraints of precedence and inhibition between decisions). Note that this part of knowledge referential is specific to the graph routing problem. The second kind of knowledge used is the set of preferences between genes according to criteria and objectives. Knowledge is extracted from: a) the project graph shape; b) information and analyses of operational experience feedback and finally; c) intermediate simulation results (individuals evaluated by EA).

3.2 The Model of Knowledge (MoK)

Paradigm used to model required knowledge is Influence Diagrams (ID) [15], a stepwise-solvable bayesian network. Firstly, they represent an interesting way for representation and use of knowledge because they were conceived for conceptual representation in decision support. So, they allow the representation of expertise and an interactive management of knowledge coming from experience feedback. Moreover, they allow an automated learning process on simulation results. This double source of knowledge (expertise and learning process) allows an interesting way for knowledge extraction [15]. Indeed, an expert can easily provide a structure of problem (or parts of it) but with uncertain parameters values. Formulating this expert knowledge by means of an ID, some rules, based on probabilities, allow calculating estimated data. These data can be compared with the data resulting from simulation or from previous carried out projects. Considering the process of KDD, statistical extraction of structures is a much more complex problem than the statistical extraction of parameters values. If we already have a structure (resulting from expertise), calculations for parameters acquisition are simplified. This cycle of information extraction improves and facilitates the construction of the model by using the two sources of knowledge (i.e. expertise and results of simulation).

An Influence Diagram (ID) is a regular acyclic and no-forgetting graph of probabilistic relations between decisions, objectives, decision criteria and environment (see left part of figure 2), based on a net. Three kinds of nodes are used: "utility nodes", "decision nodes" and "chance nodes". "Utility nodes" do not have "children nodes". To each of them, an exact value is associated for each combination of "parent nodes". They represent in our case the objectives of the project. "Decision nodes" represent the possible decisions, i.e. the genes of EA. Decision rules enable to associate to each possible configuration of "parent nodes" a single decision. Lastly, the "chance nodes" are used in order to represent context and decision criteria. A table of conditional probabilities is associated to each node. It contains all the probabilities depending on states of "parent nodes". The example on left part of figure 2 is linked to the scenario shown in figure 1. Decisions d5 and d6 are related to the options of realization (respectively Task T5 and T6) for a pen built in two parts. An expert determines that for these two decisions the principal criterion is the mode of realization (internal or external). This criterion is conditioned by the external supply (E2) of the required resources (for example a subcontracting supplier). Based on its experience, the expert estimates that these criteria can be aggregated in "Mode of realization" (C4). Once this analysis is carried out, we have a structure which could be completed by an estimated distribution of probabilities provided by the expert.

Nevertheless, a better way consists in setting these probabilities by confronting the ID with the real data coming from previous similar scenarios (e.g. a similar decision chain). This ID constitutes an "experience" in our system. This diagram ensures a conceptual classification of the properties of the decisions. The interest of conceptual classification is that it enables to define and classify the objects according to their descriptions (concepts used) without any considerations about raw data. The search for new projects or similar ones in different contexts can be done at conceptual level. This largely facilitates exchange with user and knowledge reusing.

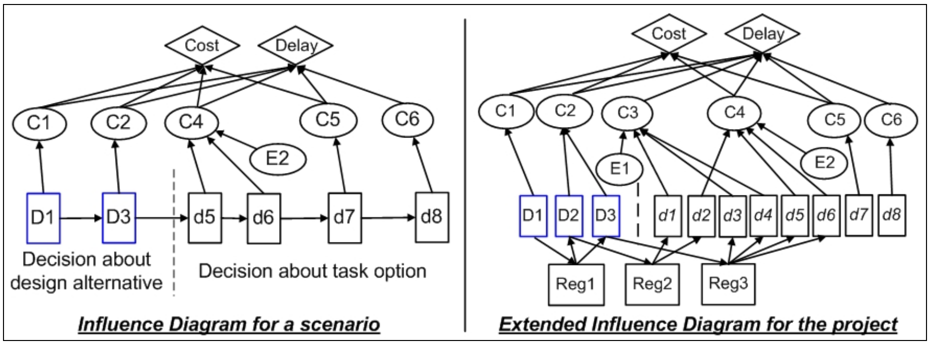

Cost
Delay
C1
C2
C4
C5
C6
E2
D1
D3
d5
d6
d7
d8
Decision about design alternative
Decision about task option
Influence Diagram for a scenario
Cost
Delay
C1
C2
C3
C4
C5
C6
E1
E2
D1
D2
D3
d1
d2
d3
d4
d5
d6
d7
d8
Reg1
Reg2
Reg3
Extended Influence Diagram for the project

Fig. 2. Influence Diagram for a scenario and for the project

The ID represented on left part of figure 2 results from a learning process, but for only one scenario. To obtain a global MoK able to guide the EA for all the possible scenarios, it is necessary to carry out a second cycle of abstraction by confronting the various ID. This global MoK is an extended influence diagram (on right part of figure 2) representing the set of possible decisions, criteria and sub-criteria linking decisions to objectives and environment. It is carried out by differential analysis of the contexts of each concept (objective, criterion or decision) used in the IDs. The context of a concept is defined as the set of concepts on which it is linked in the ID. For example, the context of a decision of realization associated with a task in a particular project (for example D1 on figure 2) is the set of the other decisions concerning the scenario, the criteria, functions and objectives related to this task. Mechanisms of context changing can be inferred each time that this task is used within a project.

These mechanisms activate or inhibit concepts associated with the task according to its context of use. The global ID is composed of three parts: in the higher part, a net of chance and utility nodes gathers the decision criteria and the objectives. It is the result of generalization of the selected decision criteria. It enables to represent all the combinations of relevant concepts. Then, the set of decisions to be evaluated are represented (D1,…, d8). It corresponds to the project's genome. Another set of decision nodes (Reg1 to Reg3) related to design alternatives is also represented. It corresponds to the structural aspect of knowledge resulting from project graph. This knowledge can be interpreted as activation constraints [16] and illustrates the fact that selection of an alternative activates corresponding tasks and inhibits tasks associated with other possible alternatives. For example, decision D3 activates d3 and d4 or d5 and d6.

4 Different Way to Use knowledge in Evolutionary Algorithm

Within our model of knowledge, some information links the search space (parameters to be selected), the objectives of project and the space of global context. So, in order to accelerate the exploration process within the EA, we use the capitalized knowledge (gathered in the MoK) adapting it to the current case. Considering that the MoK can be unsuitable or incomplete in certain cases and that it is not revised during search process, it is thus necessary to preserve independence of search method when predictions of MoK are not appropriated. For this reason, evaluation and selection steps of classic EA are preserved. Moreover, a mechanism is envisaged allowing to partially come back to traditional genetics operator when insufficient progress is observed.

The main problem for direct interaction between ID and EA is the step of inference in probability model especially because of multi-criteria fitness in ID consumes too much computing time. As a consequence, knowledge should be clusterised [2] [11] with respect to objectives and provided to EA during initialization. This way developed in next section allows general orientation to reach interesting zones of search space. The main difference between this study and the others using a MoK to guide search [10] [11] to guide search is that the MoK is more complex here (multi objectives, interdependencies between variables) but it is made before optimization process, as an off-line process.

Interaction based on knowledge clustering. This kind of interaction is based on two classifications: i) the classification of individuals with respect to objectives by EA and ii) the conceptual classification of scenarios by project's global ID. During initialization step, the global I.D. provides classes of favorable genes for each class of objectives. The classes of objectives are distributed uniformly on search space and their number is fixed by the user. A Class of genes gathers probabilities of selection for each alternative of each gene of the genome.

In a traditional EA, initial population is generated randomly. Here, the initial population is build according to the classes of genes, in order to start search procedure with a priori good orientations. The individuals are divided through the various classes of genes. Then for each individual, the probabilities of his class are used to fix the value of genes as shown on figure 3. In the class of gene, each table corresponds to the selection's probabilities for each state of a gene. For example, the values (1; 0) of the first table indicate that, to reach the matched class of objectives, it is necessary to choose with a probability of 100% the first alternative. This choice implies that genes 3, 6, 7, 8 and 9 (respectively linked to decision D3, d3, d4, d5 and d6 on figure 2) are inhibited according to structural knowledge (their probability on the class of genes is fixed to -1). When a gene is inhibited by previous genes already instantiated (genes encoding OR node's options choice), its value is fixed to 1.

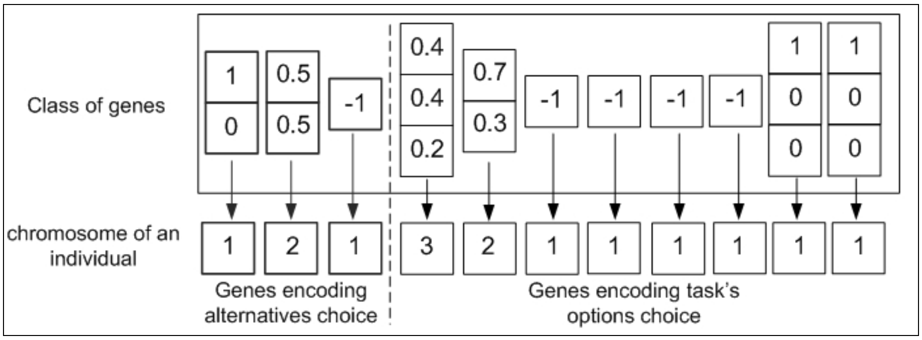

Class of genes
chromosome of an individual
Genes encoding alternatives choice
Genes encoding task's options choice

Fig. 3. Utilization of gene's class for individual building during initialization

During EA process, classes of genes are matched to current cluster of Pareto-optimal individuals (see left part of figure 4). The centered solution of cluster (i.e. which minimizes distance with other solutions) will be used as reference point for the class of genes to which it is matched. That makes it possible affecting to each individual the class of genes to which the center is closest.

The mutation operator, shown on the right part of figure 4, selects an individual randomly among the population and secondly, the probabilities of its class are used to fix the value of genes similarly as during initialization except that a gene mutation occurs according to the mutation probability and mutations on inhibited genes are skipped. This method allows to preserve good properties of an individual, to avoid useless changes and thus to concentrate the changes on the remaining genes.

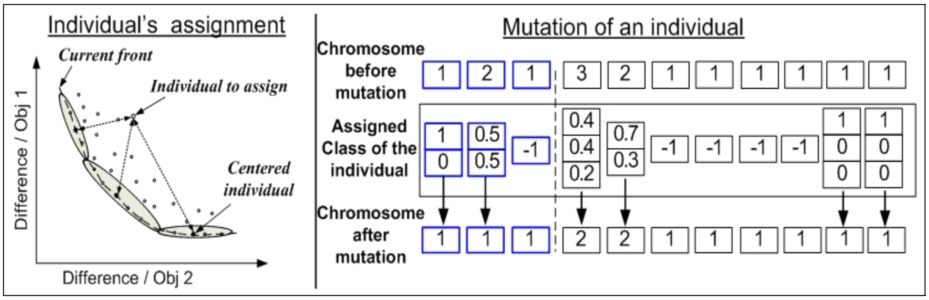


Fig. 4. Assignment of individual to classes and mutation operator

The crossover operator, illustrated on figure 5, enables exploration or intensification of search space. It corresponds then to an “inter-class” exchange by crossing individuals belonging to different classes or an “intra-class” exchange by crossing on the same class, according to the strategy of selection of parents. Once parent selection is done, probabilities of their classes are used to determine points of crossing. For each gene, if one of its parents has an inhibited gene or if the two parents have a probability of 1 for two different alternatives, the crossing does not take place for this gene. If only one of the two parents has a probability of 100% for an alternative, a unilateral crossing is done for this gene (from the parent with 100% towards the other as for first gene on figure 1). Lastly, if none of the preceding case is applicable, the crossing will be carried out in a traditional way according to the probability of crossing as for second gene on figure 1. This method makes it possible to preserve and, if possible, to exchange favorable genes of each individual.

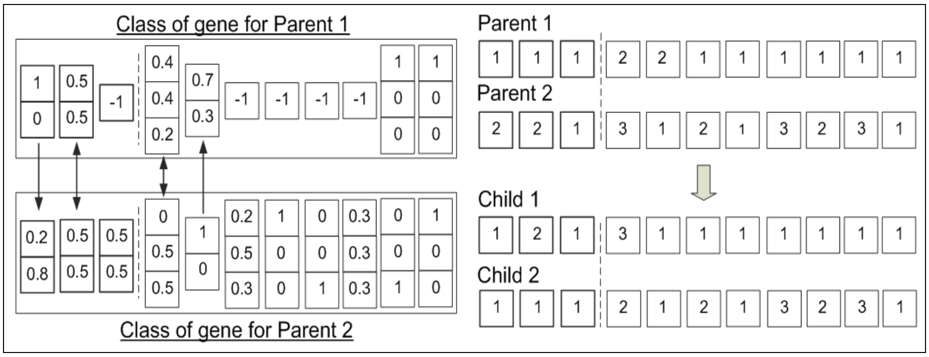


Fig. 5. Crossover operator

First results. We use a platform coded in C++ which allows testing our method. Table 1 presents preliminary results obtained for a graph with sixty tasks (three options per task) and fifteen OR node (three alternatives per OR node). The first row is the mean of best solution’s evaluation obtained after ten runs. The evaluation of an individual is calculated by adding the square of difference to each objective multiplies by a coefficient define by user (the optimal solution’s evaluation for this problem is 3051.28). The second row is the mean of generation’s number in which the best solution has been found and the last one is the time needed to achieved the ten runs on a AMD Sempron 3400+ with 512Mb of RAM.

For a first evaluation of this approach, the intelligent evolutionary algorithm (IEA) has been tested with a perfect learned MoK (class of genes learned with every best solutions generated separately) and without MoK (MoK with class equiprobable for every gene, equivalent to a traditional EA). Tests were realized for ten generations, with a population of fifty or ten individuals and with a genetic strategy (GS: P_{Cross}=0.7, P_{Mut}=0.3) or an evolutionary strategy (ES: P_{Cross}=0.3, P_{Mut}=0.7). The best strategy, here the ES one, is linked to problem instance. These first encouraging results show that the learned MoK guide very quickly the algorithm to interesting individuals as shown by runs done with ES strategy and fifty individuals where the

IEA find optimal solution at every execution in only five seconds. The learned MoK achieved best results (average of eight percents for all runs) to find the best solution and done as well for every class of objectives (the entire Pareto front is improved).

Table 1. Mean results after ten executions with and without learned MoK. The Eval. and Gen. columns represent respectively the best value obtained and the generation in which it added.

Gen/Pop/ Strat.	I.E.A. using learned MoK		I.E.A. without MoK		time
	Eval.	Gen.	Eval.	Gen.	
10/50/ GS	3074.27	1.66	3316.96	3.73	6s
10/50/ ES	3051.28 (optimal)	3.26	3274.85	4.9	5s
100/10/ ES	3051.28 (optimal)	6.4	3212.72	61.1	8s

5 Conclusion / Perspectives

An original approach based on coupling between a graphic model of knowledge built starting from experiments in an interactive and iterative way and a multi-objective evolutionary algorithm has been proposed in this paper. The first tests show promising results. We currently perform more experiments to test and validate the approach when MoK is incomplete or failing. Then some perspectives have to be carrying out. First of all, we investigate different way to use a direct interaction between MoK and EA without classification matching in order to use knowledge more specifically. Two possibilities are now investigated. The first one consists in compiling all information with an automaton as proposed by J. Amilastre in [17]. This model has very short response times but must be built before launching scenario's search process. The second possibility is to simplify the global ID. This operation can be carried out by means of an adaptive meta-model representing the relevance of a node as proposed M. Crampes in [18]. Secondly, another perspective concerns the update of the global ID starting from the past experiments (i.e. starting from the set of all the MoK corresponding to all previous instantiated scenarios). It has to be specified as well as the interaction between models and decision makers.

References

1. Talbi, D.: Application of Optimization Techniques to Parameter Set-up of Industrial Scheduling Software. Computers In Industry 55, n°2, 105–124 (2005)
2. Michalski, R.S., Wojtusiak, J., Kaufman, K.A.: Intelligent Optimization via Learnable Evolution Model. In: 18th IEEE Conf. on Tools with Artificial Intelligence, pp. 332–335 (2006)
3. Russel, S., Norvig, P.: Artificial Intelligence: A modern approach, 2nd edn. Prentice Hall/Pearson education international, London (2003)
4. Huyet, A.-L., Paris, J.-L.: Synergy between Evolutionary Optimization and Induction Graphs Learning for Simulated Manufacturing Systems. International Journal of Production Research 42(20), 4295–4313 (2004)
5. Sebag, M., Schoenauer, M.: A rule based similarity measure, vol. 837, pp. 119–130. Springer, Heidelberg (1994)

6. Cervone, K.: Experimental Validations of the Learnable Evolution Model. Congress on Evolutionary Computation, San Diego CA (2000)
7. Alami, J., El imrani, A.: A multipopulation cultural algorithm using fuzzy clustering. Applied Soft Computing 7(2), 506–519 (2007)
8. Chung, C.J.: Knowledge based approaches to self adaptation in cultural algorithms, PhD thesis, Wayne State University, Detroit, USA (1997)
9. Chebel-Morello, B., Lereno, E., Baptiste, P.: A New Algorithm to Select Learning Examples from Learning Data. In: Leung, K.-S., Chan, L., Meng, H. (eds.) IDEAL 2000. LNCS, vol. 1983, pp. 13–15. Springer, Heidelberg (2000)
10. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A new Tool for Evolutionary Computation. Kluwer, Dordrecht (2001)
11. Miquélez, M., Bengoetxea, E., Larrañaga, P.: Evolutionary computation based on Bayesian classifiers. Int. Journal AMCS 14(3), 335–349 (2004)
12. Baron, C., Rochet, S., Esteve, D.: GESOS: a multi-objective genetic tool for project management considering technical and non-technical constraints, Art. Intel. Applications and Innovations (AIAI), IFIP World Computer Congress, Toulouse (2004)
13. Zitzler, E., Thiele, L.: Multi objective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. on evolutionary computation 3, n°4, 257–271 (1999)
14. Watthayu, W., Peng, Y.: A Bayesian network based framework for multi-criteria decision making. In: MCDM 2004, Whistler, Canada (2004)
15. Becker, A., Naim, P.: Les Réseaux Bayésien: modèles graphique de connaissance, Eyrolles (1999)
16. Vareilles, E., Aldanondo, M.: Evaluation of a Solution in Interactive Aiding Design Process. In: INCOM 2006, Saint Etienne, France (2006)
17. Crampes, M.: Méta modèle adaptatif de la pertinence d'un modèle de connaissance. In: RFIA 2004, Toulouse (2004)
18. Amilastre, J., Fargier, H., Marquis, P.: Consistency restoration and explanations in dynamic CSPs – Application to configuration, vol. 135(1-2), pp. 199–234. Elsevier A.I, Amsterdam (2001)

Author Index