[12] S. Jung and B.-R. Moon, "The Natural Crossover for the 2D Euclidean TSP," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, (D. Whitley *et al.*, eds.), pp. 1003–1010, Morgan Kaufmann, 10-12 July 2000.

[13] K. Katayama and H. Narihisa, "Iterated Local Search Approach using Genetic Transformation to the Traveling Salesman Problem," in *GECCO-1999: Proceedings of the Genetic and Evolutionary Computation Conference*, (W. Banzhaf *et al.*, ed.), pp. 321–328, Morgan Kauffman, 1999.

[14] N. Krasnogor and J. Smith, "A Memetic Algorithm With Self-Adaptive Local Search: TSP as a Case Study," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, (D. Whitley *et al.*, eds.), pp. 987–994, Morgan Kaufmann, 2000.

[15] S. Lin and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

[16] K. Mathias and D. Whitley, "Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem," in *Parallel Problem Solving from Nature - Proceedings of 2nd Workshop, PPSN 2*, (R. Männer and B. Manderick, eds.), pp. 219–228, Elsevier Science Publishers, 1992.

[17] P. Merz, *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.

[18] P. Merz and B. Freisleben, "Genetic Local Search for the TSP: New Results," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, (T. Bäck, Z. Michalewicz, and X. Yao, eds.), (Piscataway, NJ), pp. 159–164, IEEE Press, 1997.

[19] P. Merz and B. Freisleben, "Genetic Algorithms for Binary Quadratic Programming," in *GECCO-1999: Proceedings of the Genetic and Evolutionary Computation Conference*, (W. Banzhaf *et al.*, eds.), pp. 417–424, Morgan Kauffman, 1999.

[20] P. Merz and B. Freisleben, "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[21] P. Merz and B. Freisleben, "Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-Partitioning," *Evolutionary Computation*, vol. 8, no. 1, pp. 61–91, 2000.

[22] P. Merz and B. Freisleben, "Memetic Algorithms for the Traveling Salesman Problem," Tech. Rep., Department of Computer Science, University of Siegen, Germany, 2001. Accepted for publication in *Complex Systems*.

[23] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," Tech. Rep. C3P Report 826, Caltech Concurrent Computation Program, California Institue of Technology, 1989.

[24] P. Moscato, "Memetic Algorithms: A Short Introduction," in *New Ideas in Optimization*, (D. Corne, M. Dorigo, and F. Glover, eds.), ch. 14, pp. 219–234, McGraw-Hill, London, 1999.

[25] H. Mühlenbein, "Evolution in Time and Space – The Parallel Genetic Algorithm," in *Foundations of Genetic Algorithms*, (G. J. E. Rawlins, ed.), Morgan Kaufmann Publishers, 1991.

[26] Y. Nagata and S. Kobayashi, "Edge Assembly Crossover: A High–power Genetic Algorithm for the Traveling Salesman Problem," in *Proceedings of the 7th International Conference on Genetic Algorithms*, (T. Bäck, ed.), pp. 450–457, Morgan Kaufmann, 1997.

[27] N. Radcliffe and P. Surry, "Fitness Variance of Formae and Performance Prediction," in *Proceedings of the Third Workshop on Foundations of Genetic Algorithms*, (L. Whitley and M. Vose, eds.), (San Francisco), pp. 51–72, Morgan Kaufmann, 1994.

[28] G. Reinelt, "TSPLIB— A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[29] A. Sinha and D. E. Goldberg, "Verificiation and Extension of the Theory of Global-Local Hybrids," in *Proceedings of the Genetic and Evolutionary Computation Conference*, (L. Spector *et al.*, ed.), pp. 592–598, Morgan Kaufmann, 2001.

[30] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A Comparison of Genetic Sequencing Operators," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 69–76, Morgan Kaufmann, 1991.

How Random Generator Quality Impacts Genetic Algorithm Performance

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

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

Abstract

It has been shown that pseudo-random number generator (PRNG) choice can affect simple genetic algorithm (GA) performance. However, these performance impacts are non-intuitive; PRNGs of poor quality can drive GAs to superior performance, for certain problems. The same PRNGs cause worse performance for other problems. In this paper we present a plausible explanation for this phenomenon: PRNGs of poor quality cause higher Vose discrepancy values than do higher quality PRNGs. Higher Vose discrepancy values could then be manifest as GA performance differences, as GA populations move toward fixed points of the Vose heuristic far away from the expectation.

1 INTRODUCTION

Several researchers have examined the impact of pseudo-random number generator (PRNG) choice on genetic algorithm (GA) performance. Meysenburg and Foster [Meysenburg, 1997, Meysenburg and Foster, 1997] examined several PRNGs, using the Knuth [Knuth, 1997] and Marsaglia's Diehard [Marsaglia, 1993] empirical test suites. They used the PRNGs to drive a simple GA, applied to a collection of several well-known GA test functions. Using a relatively coarse-grained statistical measure, they found no statistical evidence that PRNG quality affected GA performance.

In a second study Meysenburg and Foster [Meysenburg and Foster, 1999b] developed a set of specific, empirical PRNG quality tests tailored to the way a simple GA uses randomness. They used a similar set of PRNGs and the same set of GA test functions as in the previous work. They found, however, that there was no correlation between good performance on the PRNG tests and good performance by the GA. In the second study, however, a finer statistical measure was used that did reveal an interesting phenomenon.

One of the PRNGs used was a version of the Java language Random generator, limited to a period of 1000 numbers. With such a limited period, this PRNG (rand1k) failed the PRNG tests miserably. However, there was evidence that rand1k affected GA performance. It would be reasonable to assume that worse PRNG quality would cause worse GA performance, but this was not the case.

On several of the GA test functions, rand1k caused the GA to perform better than other, much better, PRNGs. On other functions, rand1k caused the GA to perform worse than the other PRNGs. In summary, Meysenburg and Foster's second study found that there was evidence that PRNG choice could impact GA performance, although in non-intuitive ways. Similar results have been noted for genetic programming (GP) systems [Meysenburg and Foster, 1999a, Daida et al., 1997, Daida et al., 1999].

In summary, the research to date on this subject shows that PRNG choice can impact GA (or GP) performance. However, the research shows no direct correlation between improved PRNG quality and improved GA performance; in fact, better PRNGs can in some cases cause worse GA performance. No one has yet been able to explain why PRNG choice can alter GA performance in this manner.

2 GA THEORY

Vose [Vose, 1999] has developed a general mathematical theory describing the behavior of simple GAs. Vose

calls the search space explored by the GA Ω. If the size of Ω is n, then GA populations can be represented as vectors in n-space. These population vectors are elements of a set that Vose terms the simplex:

$$\Lambda = \left\{ \langle x_0, \ldots, x_{n-1} \rangle : \mathbf{1}^T x = 1, x_j \geq 0 \right\}. \quad (1)$$

Elements of the simplex are column vectors of size n, where each component of the vector is non-negative, and all components of the vector sum to one. A vector $p \in \Lambda$ represents a population as follows: component p_j is the percentage of the whole of the j^{th} element of Ω in the GA population.

A GA is defined in terms of a transition rule $\tau : \Lambda \rightarrow \Lambda$, describing how a GA population evolves over time. Given an initial population vector p, the next generation would be $\tau(p)$; the following generation would be $\tau(\tau(p)) = \tau^2(p)$; and so on. Unfortunately, we are unable to say with certainty what $\tau(p)$ would be, because GAs are stochastic algorithms.

To deal with the stochastic nature of GAs, Vose introduces another function $\mathcal{G} : \Lambda \rightarrow \Lambda$, called the heuristic function. For a population vector p, the result of $\mathcal{G}(p)$ is another vector $q \in \Lambda$. q is then used as a sampling distribution to produce the next generation. The j^{th} component of q is the probability that the j^{th} element of Ω is selected to be a member of the next generation. The various operators of the GA (selection, crossover, and mutation, for example) are implemented in the particular heuristic $\mathcal{G}$ chosen. The GA population is moved forward by applying $\mathcal{G}$ to the initial population p, and using the resulting sampling distribution to create the next population. The process repeats until termination criteria are met.

Given an initial population vector p, repeated applications of the heuristic $\mathcal{G}$ produce a path through n-space. This is the expected path the GA population should follow during a run. Fixed points of $\mathcal{G}$ correspond to situations where the GA converges. The actual path followed by a GA, of course, will vary to a certain degree from the expectation, due to the stochastic nature of the process.

Vose has developed a formula for determining how far away from the expected path a particular GA population vector is.

For population vector p, the probability that the next population vector is q is shown in Figure 1. In the formula, the summations are only done for indexes where $q_j > 0$, and r is the number of individuals in the GA population.

In Figure 1, the term

$$\sum q_j \log \frac{q_j}{\mathcal{G}(p)_j} \quad (2)$$

is called the discrepancy of q with respect to the expectation $\mathcal{G}(p)$. The discrepancy is a measure of how far the actual next population vector, q, is from the expected next population vector, $\mathcal{G}(p)$. It is a measure of the distance between expectation and reality.

Our current research has shown that Vose's theory can be used to explain the non-intuitive GA behavior observed in previous studies [Meysenburg, 1997, Meysenburg and Foster, 1997, Meysenburg and Foster, 1999b]. Our hypothesis is that a PRNG of quality poor enough to drive the GA population far from the path predicted by Vose theory, would cause the GA to perform differently than a GA driven by a PRNG of higher quality. We hypothesized that a PRNG like rand1k would cause higher Vose discrepancy values for successive GA populations than a high quality PRNG like the Mersenne Twister [Matsumoto and Nishimura, 1998] would. Then rand1k might drive the GA populations into the basins of attraction of different Vose heuristic fixed points than the Mersenne Twister would; this would account for GA performance differences.

3 EXPERIMENT DESIGN

In order to test our hypothesis, we first collected 42 GA test problems suitable for Vose discrepancy statistic calculation. Since the complexity of the discrepancy measure is $O(3^l)$, for chromosome length l, the statistic can only be efficiently computed for chromosomes of approximate length 20 or less. Our test functions were created as part of an undergraduate research project. The functions are based on several different classes of problems drawn from the literature, adapted to our chromosome length restrictions. The functions have chromosome lengths ranging from eight to 20. Our GA test problems are briefly summarized in Table 1. More detailed descriptions of each of the problems may be found on the World Wide Web at the following URL: `http://ist.doane.edu/meysenburg/cooperstuff/index.html` . This page describes each test problem, as well as the parameters (crossover and mutation rates, population size, etc.) used for each run.

Next, we ran a simple GA (of the type described by Vose [Vose, 1999]) on each of the 42 GA test problems. We repeated the runs for each of 14 different PRNGs, ranging in quality from rand1k to the Mersenne Twister. Finally, to reduce the likelihood of anomalies caused by poor seed value selection, we

repeated each of our runs for 32 different PRNG seed values. For each problem / seed value combination, we initialized the GA population identically, and then used the PRNG under test for the rest of the GA run. In this way, each of the runs for a problem / seed value pair started at the same point in Ω. The seed values and initial populations were constructed using the truly random source at `www.random.org` .

We then used the Mann-Whitney non-parametric statistical test to determine if PRNG choice caused performance differences in our GA runs. We compared average population fitness on a generation by generation basis in a manner similar to Meysenburg and Foster's second study [Meysenburg and Foster, 1999b].

Finally, we calculated the Vose discrepancy statistic between each generation of each GA run. These calculations are complete for every GA test function where $l < 20$, and are still under way for the problems where $l = 20$. We used the Wilcoxson non-parametric statistical test to determine if discrepancy values caused by the rand1k PRNG were greater than those caused by the other PRNGs.

4 RESULTS

In our experiments, we again found that PRNG choice impacts GA performance. Our statistical measures here did not indicate if a PRNG caused better or worse GA performance than the other PRNGs; the measures only detected that a difference (in either direction) existed. Of all our GA runs, we found that the rand1k PRNG caused performance differences in 68% of the cases. None of our other PRNGs caused consistent performance differences across the 42 GA test functions.

Having confirmed that rand1k causes unexpected GA performance, we next tried to determine if the poor quality of rand1k caused higher Vose discrepancy values than our other PRNGs. For the GA test functions we have had time to calculate Vose discrepancy statistics for, this is indeed the case. Representative results for three of our shorter-length GA test functions are shown in Tables 2, 3, and 4.

The DC_19 GA test function has chromosome length $l = 12$. The function is an instance of CNF-SAT, for 12 variables, 300 clauses, and five variables per clause. The bits of the chromosome determine the values of each variable.

The DC_37 and DC_41 GA test functions have chromosome length $l = 8$. These functions are a modified version of the emergency-unit placement problem described by Haupt and Haupt [Haupt and Haupt, 1998]. In this case, an emergency response building must be placed on a city map, represented as a 16 by 16 grid, with a river cutting across the map at row seven. A bridge is placed over the river to allow vehicles to cross the river. For the DC_37 function, the bridge is in column one of row seven, while in the DC_41 function, the bridge is in column seven of row seven.

In the figures, the letter 'W' represents a case where the row-label PRNG caused statistically higher Vose discrepancy values compared to the column-label PRNG. The figures show that, for these GA test functions, rand1k causes higher discrepancy values than any of our other PRNGs. Other PRNGs cause sporadic Vose discrepancy differences, but rand1k causes higher Vose discrepancies compared to all other PRNGs, in all of the GA test functions we have computed the statistics on so far. We speculate that the sporadic Vose discrepancy differences of other PRNGs are caused by the small population size of our GA runs; Vose theory says that higher discrepancy values are likely in small population GAs.

It is interesting that the infamous RANDU PRNG [Knuth, 1997], which scores as badly as rand1k in the Diehard suite of PRNG quality tests, does not impact the GA in the same way rand1k does. In particular, RANDU never caused GA performance differences in our runs (while rand1k did 68% of the time), and neither did RANDU cause consistently higher discrepancy values than the other PRNGs (while rand1k did). Therefore, it seems that the Diehard suite is not predictive for GA use. We have developed a GA-specific empirical test of PRNG quality (described in a poster presented at this conference [Meysenburg et al., 2002]) which eliminates this false positive problem. Our new test, tailored to the specific GA parameters of our test functions, gives poor scores to rand1k but normal scores for RANDU.

In summary, for the GA functions we have been able to examine to date, rand1k does cause higher Vose discrepancy values than other, higher quality PRNGs.

5 CONCLUSIONS AND FURTHER WORK

We have shown that poor PRNG quality does correlate with abnormally high Vose discrepancy values. We feel that this correlation explains why a poor quality PRNG, such as rand1k, can cause improved or degraded GA performance, compared to other PRNGs. High enough discrepancy values could cause the GA

to enter the basins of attraction of unexpected fixed points of the Vose heuristic; this would be manifest as GA performance differences.

In order to further bolster our confidence in our hypothesis, we are continuing Vose discrepancy calculations on our larger GA test functions. As the results become available, we will determine if the correlation between poor PRNG quality and high Vose discrepancy values continues. In addition, we would like to determine the fixed points of the Vose heuristic for our GA test functions, in order to confirm that rand1k drives GA populations to fixed points different than other PRNGs do.

Acknowledgments

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

$$\begin{aligned}
&\Pr\{\tau(p) = q\} \\
&\quad = r! \prod \frac{\left(\mathcal{G}(p)_j\right)^{rq_j}}{(rq_j)!} \\
&\quad = exp\left(-r \sum q_j \log \frac{q_j}{\mathcal{G}(p)_j} - \sum \left(\log \sqrt{2\pi r q_j} + \frac{1}{12rq_j + \Theta(rq_j)}\right) + O(\log r)\right)
\end{aligned}$$

Figure 1: Vose equation for probability that population q came from population p.

Function	**Name**	**Length**	**Function**	**Name**	**Length**
DC_01	Rastrigin's Function	20	DC_22	Ackley's Trap Function	20
DC_02	Michalewicz's Function	16	DC_23	Ackley's 1-Max Function	20
DC_03	Whitley's Function	20	DC_24	Ackley's Mix Function	20
DC_04	Rana's Function	20	DC_25	Ackley's Plateaus Function	20
DC_05	Schwefel's Function	20	DC_26	Hoelting's Projectile	16
DC_06	Griewangk's Function	20	DC_27	Koza's Cart-Pole	20
DC_07	Schaffer's Function	20	DC_28	New Light's Bug Bomb	16
DC_08	McElvain's Fibonacci	16	DC_29	Haupt's 4-letter Word Guesser	20
DC_09	Shaffer's Function	20	DC_30	Koza's Cart-Pole II	20
DC_10	Keane's Bump Function	20	DC_31	Koza's Cart-Pole III	20
DC_11	Shopping Cart Packing	18	DC_32	Koza's Cart-Pole IV	20
DC_12	Function F9	20	DC_33	6-city TSP	18
DC_13	Schubert's Function	20	DC_34	Max Clique	16
DC_14	16-200-4 CNF-SAT	16	DC_35	6-city TSP II	18
DC_15	16-50-3 CNF-SAT	16	DC_36	6-city TSP III	18
DC_16	20-80-3 CNF-SAT	20	DC_37	Haupt's ERU Location	8
DC_17	15-5-5 CNF-SAT	15	DC_38	Haupt's ERU Location II	8
DC_18	20-80-3 CNF-SAT II	20	DC_39	Real Topology Hill-Climber	9
DC_19	20-300-5 CNF-SAT	20	DC_40	Binary-to-Gray Circuit	17
DC_20	Ackley's 2-Max Function	20	DC_41	Haupt's ERU Location III	8
DC_21	Ackley's Porcupine	20	DC_42	Meysenburg's DFA	18

Table 1: Doane College GA Test Suite functions

	add	fsr	mersenne	mother	pm	rand	rand1k	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w											
fsr														
mersenne														
mother														
pm		w	w								w			w
rand		w	w											
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu		w												
shlec		w												
shpm														
shsub														
sub														
tauss		w												
tgfsr														

Table 2: Vose discrepancy results for DC_19

	add	fsr	mersenne	mother	pm	rand	rand1k	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w		w	w		w	w	w	w	w		
fsr														
mersenne									w	w				
mother		w	w		w			w	w	w	w	w		
pm									w					
rand									w					
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu														
shlec														
shpm														
shsub									w					
sub									w	w				
tauss		w	w		w	w		w	w	w	w	w		
tgfsr		w			w			w	w	w	w			

Table 3: Vose discrepancy results for DC_37

	add	fsr	mersenne	mother	pm	rand	randlk	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w											
fsr														
mersenne														
mother														
pm		w	w								w			w
rand		w	w											
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu		w												
shlec		w												
shpm														
shsub														
sub														
tauss		w												
tgfsr														

Table 4: Vose discrepancy results for DC_41

References

[Daida et al., 1997] Daida, J., Ross, S., McClain, J., Ampy, D., and Holczer, M. (1997). Challenges with verification, repeatability, and meaningful comparisons in genetic programming. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 64–69, Stanford University, CA, USA. Morgan Kaufmann.

[Daida et al., 1999] Daida, J. M., Ampy, D. S., Raatanasavetavadhana, M., Li, H., and Chaudhri, O. A. (1999). Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's conundrum. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, FL, USA. Morgan Kaufmann.

[Haupt and Haupt, 1998] Haupt, S. and Haupt, R. (1998). *Practical Genetic Algorithms*. John Wiley and Sons.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming*, volume 2. Addison Wesley, third edition.

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

[Matsumoto and Nishimura, 1998] Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3 – 30.

[Meysenburg, 1997] Meysenburg, M. M. (1997). The effect of pseudo-random number generator quality on the performance of a simple genetic algorithm. Master's thesis, University of Idaho.

[Meysenburg and Foster, 1997] Meysenburg, M. M. and Foster, J. A. (1997). The quality of pseudo-random number generators and simple genetic algorithm performance. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 276 – 281. Morgan Kaufmann.

[Meysenburg and Foster, 1999a] Meysenburg, M. M. and Foster, J. A. (1999a). Random generator quality and gp performance. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

[Meysenburg and Foster, 1999b] Meysenburg, M. M. and Foster, J. A. (1999b). Randomness and ga performance, revisited. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

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

[Vose, 1999] Vose, M. D. (1999). *The Simple Genetic Algorithm*. MIT Press.

LINKGAUGE: Tackling hard deceptive problems with a new linkage learning genetic algorithm

Miguel Nicolau
C.S.I.S. Department
University of Limerick
Ireland
Miguel.Nicolau@ul.ie

Conor Ryan
C.S.I.S. Department
University of Limerick
Ireland
Conor.Ryan@ul.ie

Abstract

A novel approach to obtaining a tight linkage between genes in a genetic algorithm is described, and a new system based on that approach, LINKGAUGE, is proposed. Experiments presented draw a comparison between the standard messy genetic algorithm and LINKGAUGE, and show that the latter avoids deceptive traps and early convergence, with minimal computational cost. The scalability potential of the new approach is illustrated with results for two hard deceptive problems.

1 INTRODUCTION

Since they were first introduced, genetic algorithms (Holland, 1975; Goldberg, 1989) have been considered good all-round general problem solvers, and have since been applied to a variety of problems, which show their flexibility and adaptability. In the standard approach, each individual consists of a sequence of values, and operators are provided to exchange and combine those values, so that building blocks (short, highly fit sequences of values) are constructed, and later combined to form correct solutions. However, there is no mechanism to ensure a tight linkage between the values of those sequences (Goldberg, Deb, Korb, 1991); when applying standard genetic operators, this leads to an easy disruption of building blocks, rather than their maintenance (Harik, 1997), and therefore to an inability to scale-up to more difficult problems. Furthermore, an individual's genes are position dependent, in that a given locus on the genome always codes for the corresponding bit position in the phenotype. This can make crossover even less likely to maintain useful building blocks, especially if they represent geographically distant positions in the phenotype.

According to (Goldberg, Deb, Thierens, 1993), a successful algorithm should not only concentrate on the production of building blocks, but also on their preservation and exchange between individuals.

In recent years, much work has been done on achieving a tighter linkage between genes, and a family of algorithms called competent GAs has emerged (Goldberg, 2001); these are mostly based on the idea of genes coding both the position and the value of each element of an individual. These algorithms have proven to be successful when applied to hard problems, such as deceptive linkage problems (Goldberg, Korb, Deb, 1989; Goldberg, Deb, Kargupta, Harik, 1993; Harik, 1997; Pelikan, Goldberg, Cantú-Paz, 1999).

In this paper, we present a new system, LINKGAUGE, which tackles the class of deceptive linkage problems by using a simple yet effective algorithm. This system is an extension of GAUGE (Genetic Algorithms Using Grammatical Evolution), a system described in (Ryan, Nicolau, O'Neill, 2002) and based on the idea of encoding a position/value couple on each gene, to create a position-independent algorithm; GAUGE, in turn, employs many of the ideas behind Grammatical Evolution (Ryan, Collins, O'Neill, 1998; O'Neill, Ryan, 2001; O'Neill, 2001). So far, GAUGE has been successfully applied to both standard and deceptive ordering problems.

Our aim when running the experiments described in this paper was to test the aptitude of LINKGAUGE to solve linkage problems, and its scalability when presented with more difficult problems; to do so, we applied the system to two hard deceptive linkage problems, and compare its performance to the standard messy Genetic Algorithm (Deb, Goldberg, 1991). By extending GAUGE's mapping mechanism, we have built a new flexible approach to this kind of problem; our results show by comparison that it finds a solution faster, scales better to harder versions of the

problem, and requires far less hardware resources than the messy GA[1].

This paper is organized as follows: we start by briefly introducing Grammatical Evolution in section 2, followed by an explanation of how GAUGE works (section 3) and its extension into LINKGAUGE (section 4). In section 5 we present the problems used for our experiments, and in section 6 we present our results. Finally an analysis of those results is made and conclusions are drawn in section 7, followed by the outline of some future directions of research in section 8.

2 GRAMMATICAL EVOLUTION

GAUGE is based upon many of the techniques implemented in Grammatical Evolution, so we start with an introduction to this system, to highlight the similarities and differences between the two systems.

Grammatical Evolution (GE) is an evolutionary algorithm approach to automatic program generation, which evolves strings of binary values, and uses a BNF (Backus-Naur Form) grammar to map the strings into programs. This mapping involves transforming the binary individual into a string of integer values, and then using those values to choose transformations from the given grammar, so that a start symbol is mapped into a syntactically correct program.

This process is based on the idea of a genotype to phenotype mapping: an individual comprised of binary values (genotype) is evolved, and, before being evaluated, is subjected to a mapping process to create a program (phenotype), which is then evaluated by the fitness function. This creates two distinct spaces, a search space and a solution space.

The degenerate genetic code employed in GE also plays a role in the performance of the system, as seen in (O'Neill, Ryan, 1999); by using the *mod* function to normalize each integer to a finite number of production rules, different integer values can be used to select the same rule. The genotype can therefore be modified without necessarily affecting the phenotype, in a process known as neutral mutations (Kimura, 1983; Banzhaf, 1994).

Finally, the functionality of the values in the integer string is dependent on the values preceding it, as those determine which non-terminal symbols remain to be mapped. This creates a linkage between each gene

[1]Due to its variable length nature, some messy GA runs required over 1GB of memory to store a population, comparing to less than 1MB for the most demanding LINKGAUGE runs.

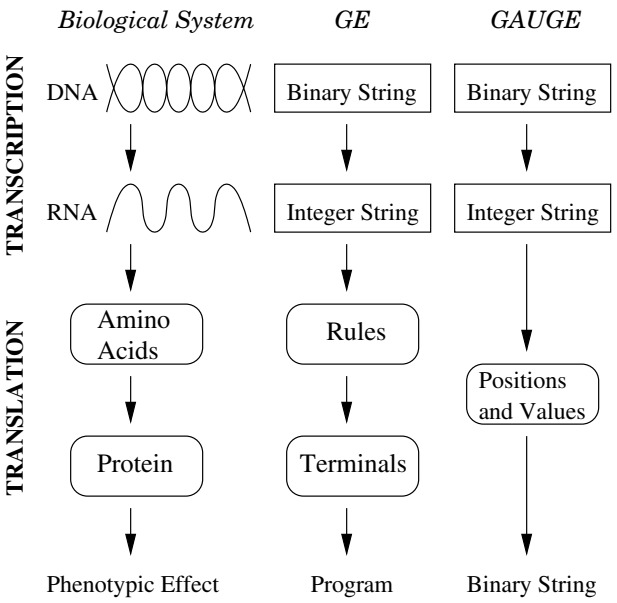


Figure 1: Genotype to Phenotype mapping

on the chromosome and all those which precede it, and helps the individual in preserving good building blocks during the evolution process, where it is subjected to the harsh effects of operators like crossover. This has been termed the "Ripple Effect" (Keijzer, Ryan, O'Neill, Cattolico, Babovic, 2001).

3 GAUGE

GAUGE is based on many of the same ideas behind the implementation of GE. It uses a genotype to phenotype mapping in much the same fashion: an individual is composed of a binary sequence (genotype) which, once ready for evaluation, is mapped onto a string of integer values, which are decoded as a collection of *(position, value)* pairs to finally build a new binary string (the phenotype), ready to be evaluated. Figure 1 illustrates this process, and compares it to GE's analogy to molecular biology.

Another feature of GE upon which GAUGE is based is that the function of a gene in an individual depends on the value of the genes preceding it; this creates a tight linkage between adjacent genes in that individual.

Since the position and value of each bit of the phenotype string are expressed on each gene, geographically disparate values of the phenotype can be grouped together on the genotype. This leads to the creation of tight building blocks at the start of the genome that can be gradually grown by the evolutionary process, in a process we call competitive building blocks.

Work by Bean (Bean, 1994) with the Random Keys Genetic Algorithm (RKGA) hinted that a tight linkage between genes would result in both a smoother transition between parents and offspring when genetic operators are applied, and an error-free mapping to a sequence of ordinal numbers.

3.1 EXAMPLE GAUGE MAPPING

In this subsection we take a look at how an individual is created and evaluated using GAUGE. Let us take as an example individual the following binary sequence:

```
0110 0111 0001 0100 0111 1001 0010 0011
```

The first step is to map it onto an integer string. For the purpose of brevity, we will use four bits to encode each integer (rather than the standard eight used in the actual GAUGE code), and therefore end up with:

```
6 7 1 4 7 9 2 3
```

This string will be evaluated as a sequence of four (*position, value*) pairs, and will be used to fill in a string of four bits. We therefore take the first position, `6`, and map it onto the number of available positions in the final string (i.e., `4`), by calculating the remainder of the division of `6` by `4` (`6 % 4`), giving the value `2` (i.e., the third position in the phenotype string). We use the same mapping process to transform the value for that position, `7`, into a binary value: `7 % 2 = 1`. This is the state of the final array after the above steps are executed:

```
?  ?  1 ?
```

By taking the next pair, (`1,4`), we again map the position onto the number of available positions, in this case 3, which gives us `1 % 3 = 1` (second free position), and normalize the value `4` onto a binary value, which gives us `4 % 2 = 0` :

```
?  0 1 ?
```

With the next pair, (`7,9`), we map the position `7` onto the number of available positions, `2`, by calculating `7 % 2 = 1` (second free position, which is the last position in the string), and the value `9` onto a binary value, `9 % 2 = 1`:

```
?  0 1 1
```

Finally, with the last pair, we map the position `2` onto the number of remaining places, in this case `1`, giving the value `2 % 1 = 0`, and place the value `3 % 2 = 1` in it. Note that the last position will always be mapped onto value `0`, since there is only one free position left in the final individual. Our phenotype, now ready for evaluation, is the string:

```
1 0 1 1
```

3.2 EARLY RESULTS

In (Ryan, Nicolau, O'Neill, 2002), GAUGE was applied to both a standard genetic algorithm problem and a deceptive ordering problem. On the former, its performance was as good as that of a simple genetic algorithm, showing that its overhead processing (namely its mapping process) does not reflect in a loss of performance in simple problems, while on the latter, its (*position,value*) specification was shown to provide the flexibility of swapping elements in a solution, helping the system to avoid local optima. The interested reader is referred to the mentioned paper.

4 LINKGAUGE

In this section the LINKGAUGE system is presented. The idea is to extend the tight linkage between the gene positions, as seen in GAUGE, to the gene values themselves. This is achieved by extending GAUGE's mapping process: every time a value is to be placed on the phenotype string, it is calculated by adding all the previous *value* fields in each (*position, value*) pair and then normalizing the result over the range of accepted values. The value each gene will provide can therefore be calculated by the formula

$$(\sum_{i=0}^{n} x_i)\%v$$

where
n = order of the gene (i.e. gene 0, gene 1, etc)
x_i = number in *value* field for gene i
v = value to normalize (for binary strings, 2 is used)

It should be noted that, theoretically, any function could be used to introduce dependency between the values; the suitability of other functions will be the subject of further research.

4.1 EXAMPLE LINKGAUGE MAPPING

Following the GAUGE mapping example, the pair (`6,7`) will generate the same string as before:

```
?  ?  1 ?
```

In the next pair, however, the value is calculated by `(7+4) % 2` (i.e., the cumulative total of the previous *value* fields normalized over the range of binary numbers), giving the value `1`. The position calculation is the same as before (`1 % 3 = 1`), so we end up with the string:

```
?  1 1 ?
```

In the next pair, the value will be calculated by `(7+4+9) % 1`, giving the value `0`, and the final value is calculated by `(7+4+9+3) % 1 = 1`. The final string will be:

```
1 1 1 0
```

The objective of this mapping is to create a tight linkage between the value of the genes. The previously mentioned "Ripple Effect" is therefore extended to the values within the genes themselves.

5 DECEPTIVE PROBLEMS

In this section we introduce the two deceptive problems which we used on our experiments. These were used to test the performance of LINKGAUGE, and to compare it to the messy GA, using the mGA code available in the IlliGAL web site and described in (Deb, Goldberg, 1991). We chose to compare our system to the messy GA as the latter is the origin of most modern competent GAs, introducing the concepts of primordial and juxtapositional phases, over- and under-specification, and competitive templates. Future work should include comparisons to other more recent competent GAs.

5.1 ORDER-THREE DECEPTIVE PROBLEM

The order-three deceptive problem was the first problem reported using the original mGA, in (Goldberg, Korb, Deb, 1989). In the original problem, ten order-three deceptive sub-functions are concatenated together to form a 30-bit length problem. We have extended the problem, and used lengths of 30, 45, 60, 75, 90 and 105 bits.

Each sub-function has a global optimum (`000`) and a deceptive local optimum (`111`). The objective is to create a series of local optima that will attempt to keep the systems from reaching the one and only global optimum; on the 105-bit problem, this means there are 2^{105} (`4.05e+31`) possible solutions, with 2^{35} (`3.44e+10`) optima (local and global optimum combinations within each of the sub-functions), of which only one is the global solution for the entire string. Table 1 shows the function values for every 3-bit combination.

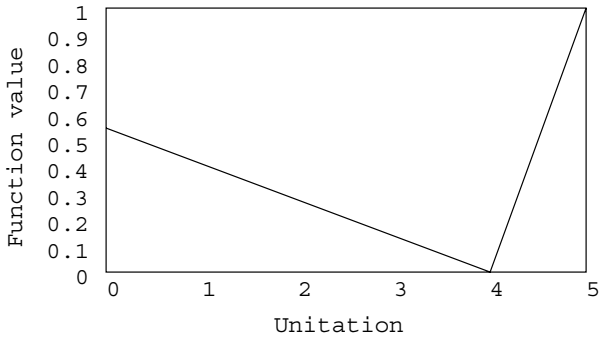


Figure 2: Order-Five Deceptive Problem Unitation Graph.

Table 1: Order-Three Sub-function Values.

String	Value	String	Value
000	28	100	14
001	26	101	0
010	22	110	0
011	0	111	30

5.2 ORDER-FIVE DECEPTIVE PROBLEM

In (Goldberg, Deb, Kargupta, Harik, 1993), a performance comparison between the original messy GA and the Fast Messy Genetic Algorithm is made, by using both the order-three sub-function and an order-five sub-function; we used the same problems in our tests.

In this problem, substrings of five bits are concatenated together, with the global optimum being (`11111`) and the local optimum (`00000`). Figure 2 shows this problem in terms of a unitation graph, i.e. the number of `1`s in a sub-function determines its fitness. This function is fully deceptive, as can be seen in (Deb, Goldberg, 1994).

6 EXPERIMENTS

In this section we present the results obtained on the two described problems, using both LINKGAUGE and the original messy GA. We start by describing the experimental setup used on each system, follow with an overview of the results obtained in our experiments, and conclude this section with a discussion of those results.

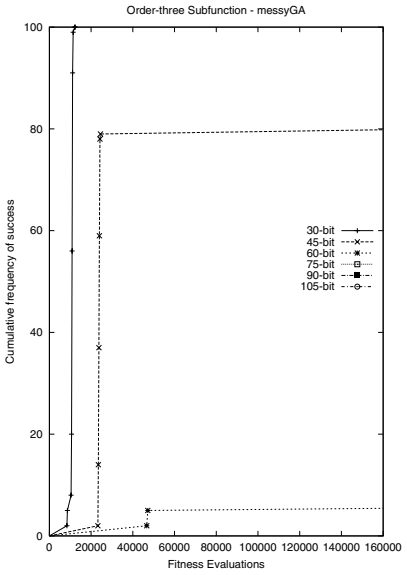


Figure 3: Order-Three Results For MessyGA; No Results Were Obtained With String Lengths Over 60 Bits.

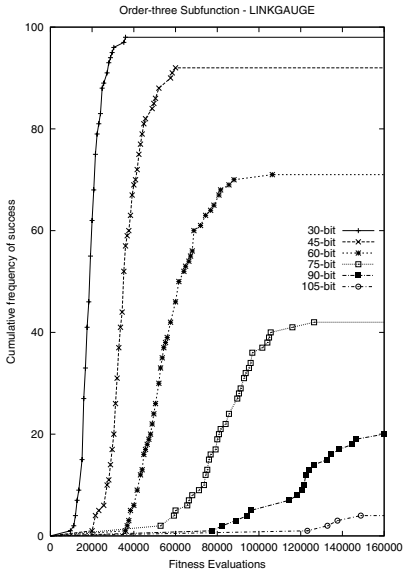


Figure 4: Order-Three Results For LINKGAUGE.

6.1 EXPERIMENTAL SETUP

We used a standard configuration with LINKGAUGE for this problem. With a population of 800 individuals, the replacement strategy used was steady-state, and the selection routine was roulette-wheel; probability of crossover was set to 0.9, and mutation was set to 0.01, which are the standard Genetic Programming (Koza, 1992) values used in GE; no attempt was made to optimize these values. The maximum number of fitness function calls was set to `1.6e+04`, on both systems.

Table 2: Tested combinations of settings for the messy GA algorithm.

Parameters	Set 1	Set 2	Set 3	Set 4
Maximum era	3	3	4	3
Prob. cut	0.02	0.02	0.02	0.02
Prob. splice	1.0	1.0	1.0	1.0
Prob. allelic mut.	0.0	0.0	0.0	0.0
Prob. genic mut.	0.0	0.0	0.0	0.0
Thresholding	no	yes	yes	yes
Tie-breaking	no	yes	yes	yes
Reduced popsize	no	yes	yes	yes
Extra members	no	no	no	no
Copies	5,1,1	5,1,1	5,1,1,1	5,1,1
Total generations	20	20	15	100
Juxtapos. popsize	250	250	250	100

In the messy GA, a range of different sets of parameters were tried as seen in Table 2; a detailed explanation of these settings can be found in (Goldberg, Korb, Deb, 1989; Goldberg, Deb, Korb, 1991). Settings 1 (the standard messy GA values) and 2 behaved well with the 30-bit string problem, but gave very poor results with longer string lengths, and were therefore discarded; settings 3 and 4 gave the improved results, with setting 3 achieving the best performance on the harder problems, and so was chosen for our comparison.

6.2 RESULTS

Both systems were applied to each of the problems over 100 runs, and the graphs presented here show the cumulative number of successful runs plotted against the number of fitness evaluations required.

6.2.1 Order-Three Problem

Results for the order-three problem, shown in Figures 3 and 4, show the messy GA achieving a superior performance with small string lengths. However, as the string length gets longer, it can be seen that LINKGAUGE scales better to the problem; with lengths of 75-bit, 90-bit and 105-bit, all messy GA runs failed to find one instance of the global optimum (a string composed of all 1s), and therefore they are not plotted in the graph presented.

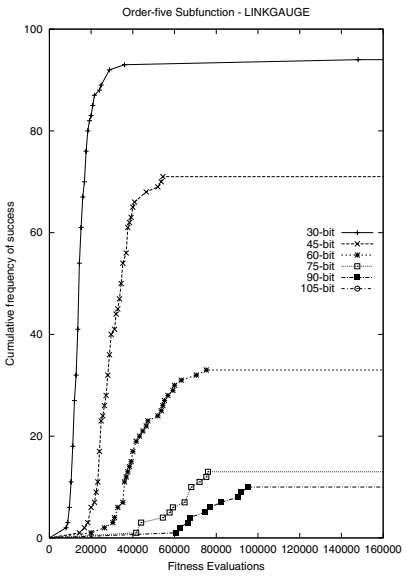


Figure 5: Order-Five Results for LINKGAUGE.

6.2.2 Order-Five Problem

According to results published in (Goldberg, Deb, Kargupta, Harik, 1993), the original messy GA with standard parameter settings would need a number close to `1e+08` function evaluations to find an instance of a global solution for the order-five problem, with a 50-bit string. It is therefore no wonder that in our tests, the messy GA failed to find any instance of the global solution over `1.6e+04` fitness function calls[2].

The results obtained with LINKGAUGE (shown in Figure 5) are, however, similar to those obtained in the order-three problem, which suggests that the extra deceptiveness of order-five problems doesn't have as strong an impact on its performance as one might expect. Over the allowed number of fitness evaluations, only in the 105-bit problem did LINKGAUGE not find any solution. It should be mentioned, however, that in this class of problem LINKGAUGE worked better with a population size of 1600 individuals using the same number of fitness function calls (and indeed found solutions for the 105-bit problem), which tends to suggest that a better trade-off between number of generations versus population size can be found; this could, however, be looked upon as parameter optimization, and therefore those results are not reported here in detail.

[2]It also increased its hardware requirements exponentially; on one specific run, to specify the contents of a 30-bit string, the average length of an individual was over 22000 genes.

6.3 ANALYSIS

On the standard order-three problem, a direct comparison with the messy GA shows LINKGAUGE's ability to adapt to an increasing problem difficulty; although with smaller strings the messy GA is faster at finding a solution (with any of the parameter sets tested), it does not present the scalability of LINKGAUGE when the problem gets harder.

On the order-five deceptive problem, the lack of results for the messy GA, and the only slight loss of performance of LINKGAUGE, underline the scale-up properties of the latter. Results obtained (but not reported, for the sake of clarity) have shown that with larger population sizes and the same number of fitness evaluations, LINKGAUGE's performance in this problem increased, which leads to some optimism as to the system's ability to avoid early convergence.

7 CONCLUSIONS

We have presented a new genetic algorithm based system, LINKGAUGE, for the purpose of solving hard deceptive linkage problems. The results reported show an interesting scale-up property for our system, which is remarkable given that it is based on a simple genetic algorithm; no specific genetic operators have been introduced, and parameters such as crossover and mutation rate have been set to standard values. This is not the case of the system compared to, messy GA, which has a specific implementation that slightly diverges from the original implementation ideas of genetic algorithms: although a good approach in itself, this does make the algorithm harder to use and understand, and parameter tuning was required to achieve a good performance. It should also be mentioned that LINKGAUGE's fixed-length, fixed-population size nature results in an algorithm that has very little hardware and over-head processing requirements, especially when compared to the original messy GA.

8 FUTURE WORK

Future lines of research include numerical and statistical analysis of the data presented, to effectively measure the performance and the degree of scalability of the system. Also, a rigorous comparison should be made between the system presented and other more recent linkage learning genetic algorithms (Goldberg, Deb, Kargupta, Harik, 1993; Harik, 1997), to highlight similarities and differences, and advantages/disadvantages.

Acknowledgments

The authors would like to thank Dr. Michael O'Neill for his advice on writing this paper, and for his driving force in the initial implementations of GAUGE.

References

Banzhaf, W. 1994. Genotype-Phenotype-Mapping and Neutral Variation - A case study in Genetic Programming. In *Parallel Problem Solving from Nature III*, Springler. (pp. 322-332)

Bean, J. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. ORSA Journal on Computing, Vol. 6, No. 2, Spring 1994. (pp. 154-160)

Deb, K., and Goldberg, D. E. 1991. mGA in C: A Messy Genetic Algorithm in C. Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 91008.

Deb, K., and Goldberg, D. E. 1994. Sufficient Conditions for Deceptive and easy Binary Functions. *Annals of Mathematics and Artificial Intelligence 10.* (pp. 385-408)

Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley.

Goldberg, D. E. 2001. The Design of Competent GAs: Toward a Computational Theory of Innovation. Tutorial presented at the Genetic and Evolutionary Computation Conference, San Francisco, July 2001.

Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. 1993. Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 93004.

Goldberg, D. E., Deb, K, and Korb, B. 1991. Don't Worry, be Messy. In *Proceedings of the Fourth International Conference on Genetic Algorithms (San Mateo, CA)*, R. Belew and L. Booker, Eds., Morgan Kaufman. (pp. 24-30)

Goldberg, D. E., Deb, K., and Thierens, D. 1993. Toward a Better Understanding of Mixing in Genetic Algorithms. Journal of the Society of Instrument and Control Engineers, Vol. 32, No. 1. (pp. 10-16)

Goldberg, D. E., Korb, B., and Deb, K. 1989. Messy genetic algorithms: Motivation, analysis, and first results. in *Complex Systems, 3.* (pp. 493-530)

Harik, G. 1997. Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms. Doctoral Dissertation, University of Michigan, Ann Arbor.

Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press.

Keijzer M., Ryan C., O'Neill M., Cattolico M., and Babovic V. 2001. Ripple Crossover in Genetic Programming. In *LNCS 2038, Proceedings of the Fourth European Conference on Genetic Programming*, Springer. (pp. 74-86)

Kimura, M. 1983. The Neutral Theory of Molecular Evolution. Cambridge University Press.

Koza, J. 1992. *Genetic Programming.* MIT Press.

O'Neill, M. 2001. Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Doctoral Dissertation, University of Limerick.

O'Neill, M., and Ryan, C. 1999. Genetic Code Degeneracy: Implications for Grammatical Evolution and Beyond. In *ECAL'99: Proceedings of the Fifth European Conference on Artificial Life.*

O'Neill, M., and Ryan, C. 2001. Grammatical Evolution. IEEE Transactions on Evolutionary Computation, Vol. 5, No. 4. (pp. 349-358)

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. 1999. BOA: The Bayesian Optimization Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Morgan Kaufman. (pp. 525-532)

Ryan, C., Collins, J.J., and O'Neill, M. 1998. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In *LNCS 1391, Proceedings of the First European Workshop on Genetic Programming*, Springer-Verlag. (pp. 83-95)

Ryan, C., Nicolau, M., and O'Neill, M. 2002. Genetic Algorithms using Grammatical Evolution. In *Proceedings of EuroGP-2002.* (to appear)

Setting the Mutation Rate: Scope and Limitations of the $1/L$ Heuristic

Gabriela Ochoa
Universidad Simon Bolivar, Computacion y TI
Sartenejas, Miranda, Apartado 89000, Caracas 1080-A, Venezuela
E-mail: gabro@ldc.usb.ve
Tel.: +58 212 906 32 63, Fax.: +58 212 906 32 43

Abstract

An important decision to make when designing a GA is how to set the evolutionary parameters. Among these parameters, the mutation rate has been acknowledged as the most sensitive one. All approaches so far for a near-optimal setting of the mutation rate have intrinsic limitations. A promising guideline is, however, the heuristic suggesting $p_m = 1/L$ where L is the string length. This paper is a first attempt to explore the scope and limitations of this heuristic on GAs with bit-string representation. Specifically, we select two real-world domains as test problems and explore (i) whether optimal mutation rates change with time; and (ii) the interactions between the mutation rate and other evolutionary parameters. Results suggest that a constant mutation rate of $1/L$ is useful for a GA with a controlled 'moderate' selection pressure. It should be, however, revised for a weak or extremely strong selection pressure, and for a small population size.

1 INTRODUCTION

It has been suggested that the most sensitive of GA parameters is the mutation rate [Schaffer et al., 1989, Bäck, 1996]. Several studies in the literature look for 'optimal' mutation rates, or optimal schemes for varying the mutation rate over a single run [Fogarty, 1989, Davis, 1989, Bäck, 1991, Mühlenbein, 1992, Julstrom, 1995, Tuson and Ross, 1998].

We believe that the most useful guideline so far for an effective and general setting of the mutation rate in GAs is the heuristic suggesting $p_m = 1/L$ (per bit), where L is the string length. This figure has appeared several times in the evolutionary computation literature. The earliest appearance we can trace back was due to [Bremerman et al., 1966] as quoted by [Bäck, 1996]. Also, in his dissertation [DeJong, 1975] suggested this value as quoted by [Hesser and Männer, 1991]. The work of [Mühlenbein, 1992] states that $p_m = 1/L$ is optimal for general unimodal functions. This setting has also produced good results for several NP-hard combinatorial optimization problems such as the multiple knapsack problem [Khuri et al., 1994], the minimum vertex cover problem [Khuri and Bäck, 1994], the maximum independent set problem [Bäck and Khuri, 1994], and others [Bäck and Khuri, 1994]. The work of [Smith and Fogarty, 1996] found $1/L$ as the best fixed setting for the mutation rate, giving results comparable to their best self-adaptive method. Other authors have found a dependence of effective mutation rates upon the string length L, although they had not explicitly suggested $p_m = 1/L$ [Schaffer et al., 1989, Hesser and Männer, 1991, Bäck, 1992, Bäck, 1993].

Thus, there may well be some true principle underlying this heuristic. In previous work, we argued that this principle is related to the notion of error threshold from molecular evolution [Ochoa et al., 1999, Ochoa et al., 2000]. The error threshold is the minimal replication accuracy that still maintains genetic information in the population.

This paper is a first attempt to explore the scope and limitations of the $1/L$ heuristic on GAs with bit-string representation. Specifically, we select two real-world domains as test problems and explore (i) whether optimal mutation rates change with time; and (ii) the interactions between the mutation rate and other evolutionary parameters (the selection pressure and the population size).

The remainder of this document is organized as follows. Section 2 describes the test problems used in

this paper: a combinatorial optimization problem — the Multiple Knapsack problem, and an engineering problem — the design of an optimal aircraft Wing-Box. Sections 3 and 4 describe our methods and results respectively, and, finally, Sections 5 and 6 summarizes our findings.

2 TEST PROBLEMS

Two real-world domains were selected for study, namely, a combinatorial optimization problem — the Multiple Knapsack problem, and an engineering problem — the design of an optimal aircraft Wing-Box. The Multiple Knapsack is a maximization problem, whereas the Wing-Box is a minimization problem. This selection is somewhat arbitrary, but is consistent with the following criteria. First, both are complex problems: the Wing-Box is an engineering design problem based on real data and constraints, and the Multiple Knapsack is a highly constrained combinatorial optimization problem known to be NP-hard. Second, both problems were available and relatively easy to implement, and third, both have a natural bit string encoding which was a requirement for the present study. Additionally, these two problems are completely unrelated, so common results have a good chance to convey some generality. It is worth noting, however, that other real-world problems may have very different characteristics from these two test problems.

2.1 THE WING-BOX PROBLEM

The Wing-Box problem was formulated as part of the Genetic Algorithms in Manufacturing Engineering (GAME) project at COGS, University of Sussex [1]. An industrial partner, British Aerospace, provided data from a real Airbus wing box.

A common problem faced in the design of aircraft structures, is to define structures of minimum weight that can withstand a given load. Figure 1 sketches the elements of a wing relevant to this problem. The wing is supported at regular intervals by slid ribs which run parallel to the aircraft's fuselage. On the upper part of the wing, thin metal panels cover the gap separating adjacent ribs. The objective is to find the number of panels and the thickness of each of these panels while minimizing the mass of the wing and ensuring that none of the panels buckle under maximum operational stresses. More details, and the equations for calculating the fitness function, can be found in [McIlhagga et al., 1996].

[1] http://www.cogs.susx.ac.uk/projects/game/

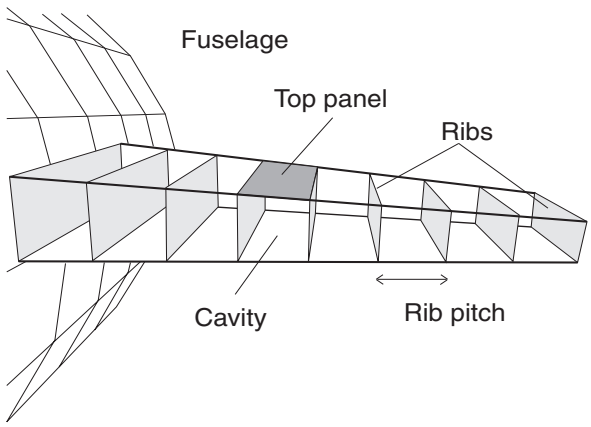


Figure 1: Relevant elements of a wing. Wing dimensions are fixed. The variable elements are the number of ribs and the thickness of the top panels.

A full description of a potential solution to the Wing-Box problem requires the definition of the number of ribs N and the thickness of the $N-1$ panels. There is a constraint on the thickness of these panels which is that adjacent panels should not differ in thickness by more than 0.25 mm. The simplest way to accomplish this, is to encode the differences in thickness between adjacent panels rather than the absolute thickness of the panels. If we know the difference in thickness $\delta th(i)$ between panels i and $i+1$ for $i \in (1, N-1)$, the absolute thickness of the first panel is enough to define everything else.

Originally, the Wing-Box parameters were encoded following the order described by Figure 2. For the experiments in this paper we fixed the number of panels in 50 (i.e $N = 51$ ribs, since the number of ribs is 1 + the number of panels), thus our genetic encoding is the same, but excluding the first gene. The thickness of the first panel was allowed to vary between 10 and 15 mm by steps of 10^{-3} mm. This requires 5×10^3 values which can be represented with a minimum of 13 bits. For all subsequent $N-2$ panels the difference in thickness with the previous panel is encoded. According to manufacturing tolerance considerations, only five values were allowed for these differences in thickness: $\{-0.25, -0.125, 0.0, 0.125, 0.25\}$. Three bits are needed to encode these five values. Notice that a change in $\delta th(i)$ leads to changes in the thickness of panel $i+1$, and of all subsequent panels up to the tip of the wing. Notice also that in both the encoding of the first section, and the remainder $N-2$ sections, there is an amount of redundancy in the genotype to phenotype mapping. To sum up, the number of bits needed for encoding an individual is 13 for the first

panel, and 3 for each of the others 49 panels, that is $13 + 3 \times 49 = 160$.

N	th(1)	Δ th(1)= th(2)-th(1)	...	Δ th(i)= th(i+1)-th(i)	...	Δ th(N-2)= th(N-1)-th(N-2)

N: Number of ribs
th(i): Thickness of ith panel

Figure 2: Genetic representation of the wing parameters.

2.2 THE MULTIPLE KNAPSACK PROBLEM

The combinatorial optimization problem described here, called the 1/0 multiple knapsack problem, follows the specifications given by [Khuri et al., 1994]. This problem is a generalization of the 0/1 simple Knapsack problem where a single knapsack of capacity C, and n objects are given. Each object has a weight w_i and a profit p_i. The objective is to fill the knapsack with objects producing the maximum profit P. In other words, to find a vector $x = (x_1, x_2, \ldots, x_n)$ where $x_i \in \{0, 1\}$, such that $\sum_{i=1}^{n} w_i x_i \leq C$ and for which $P(x) = \sum_{i=1}^{n} p_i x_i$ is maximized.

The multiple version consists of m knapsacks of capacities $c_1, c_2, \ldots, c_m$ and n objects with profits $p_1, p_2, \ldots, p_n$. Each object has m possible weights: object i weighs w_{ij} when considered for inclusion in knapsack j ($1 \leq j \leq m$). Again, the objective is to find a vector $x = (x_1, x_2, \ldots, x_n)$ that guarantees that no knapsack is over-filled: $\sum_{i=1}^{n} w_{ij} x_i \leq c_j$ for $j = 1, 2, \ldots, m$; and that yields maximum profit $P(x) = \sum_{i=1}^{n} p_i x_i$.

This problem leads naturally to a binary encoding. Each string $x_1 x_2 \ldots x_n$ represents a potential solution. If the ith position has the value 1 (i.e. $x_i = 1$) then the ith object is in all knapsacks; otherwise, it is not. Notice that a string may represent an infeasible solution. A vector $x = (x_1, x_2, \ldots, x_n)$ that over-fills at least one of the knapsacks; i.e., for which $\sum_{i=1}^{n} w_{ij} x_i > c_j$ for some $1 \leq j \leq m$, is an infeasible string. Rather than discarding infeasible strings and thus ignore infeasible regions of the search space, the approach suggested by [Khuri et al., 1994] is to allow infeasible strings to join the population. A penalty term reduces the fitness of infeasible strings. The farther away from feasibility, the higher the penalty term of a string. Thus, the following fitness function was defined (s is the number of over-filled knapsacks):

$$f(x) = \sum_{i=1}^{n} p_i x_i - s \times max(p_i) \quad (1)$$

Hence, the fitness function uses a graded penalty term $max(p_i)$. The number of times this term is subtracted from the fitness of a infeasible solution is equal to the number of over-filled knapsacks that the solution produces.

A Multiple Knapsack instance, taken from the literature (termed Weish 30), was used as test problem. It has 90 objects and 5 sacks. This (and several other) problems are available online from the OR-library by [Beasley, 1990]. Weish 30 is among the biggest and more complex Multiple Knapsack instances available in the library.

3 METHODS

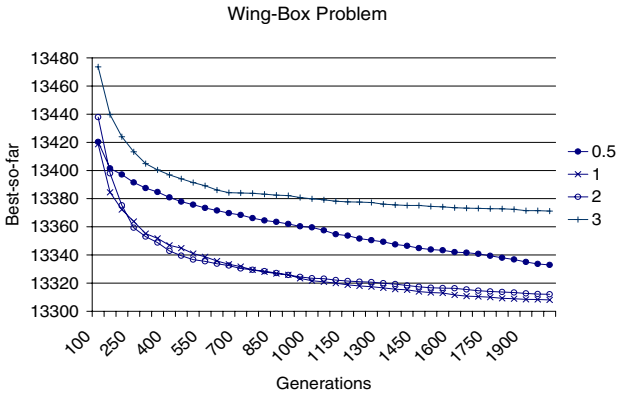
For estimating optimal mutation rates in GAs we need to define what an optimal or near-optimal mutation rate is. The working definition used here is: an optimal mutation rate is that producing optimal performance. But then, we need a good way of measuring GA performance. Given the randomized nature of GAs, conclusions can never be drawn from a single run. Instead, the common practice is to consider statistics from a sufficiently large number of independent runs. So, the standard performance measures for GAs are the average and best fitness values attained after a prefixed termination criterion, averaged over several runs. Within a given run, the best fitness could be either the current best in the population, or the best fitness attained so far. These measures are considered after a fixed termination criterion, or over fixed intervals throughout the GA run. For the experiments in this paper, we will consider the best fitness attained so far after a fixed termination criterion. This criterion will be carefully selected in each case to be long enough to stabilize the best and average fitness of the population. The average of several runs will be considered (typically 50) and the standard deviation will be shown in most cases. The first empirical section, however, studies the time dependency of the mutation rate. In this case best-so-far fitness values are reported at fixed intervals.

To study the applicability of the $1/L$ heuristic, we explore the effect of modifying some relevant evolutionary parameters on the magnitude of optimal mutation rates. Specifically, we explore the effects of modifying the selection pressure and population size. Unless otherwise stated, experiments use a generational GA

with tournament selection (tournament size = 2), a population of 100 members, and both mutation and recombination (two-point with a rate of 1.0). Table 1 summarizes these default settings. Further details on the experiments and departures from the default settings are given in the respective results subsections.

Population replacement	Generational
Selection scheme	Tournament (T. Size = 2)
Population size	100
Recombination rate	1.0
Recombination operator	Two-point
Termination criterion	2,000 Generations
Number of runs	50

Table 1: GA default parameters used in the experiments.



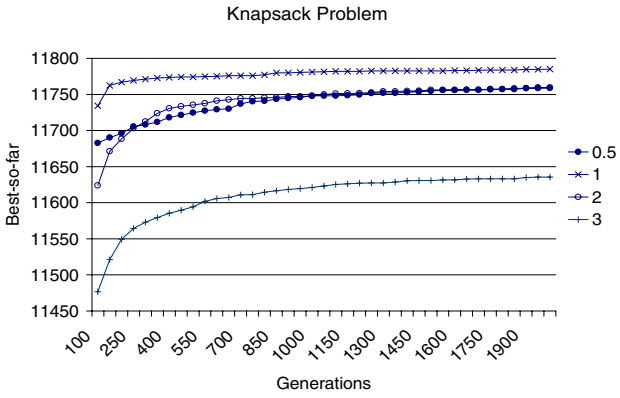


Figure 3: Comparing the performance of different mutation rates over a GA run on both test problem. The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

4 RESULTS

Three groups of experiments were performed with the aim of exploring: (i) the time-dependency of the mutation rate, (ii) the effect of modifying the selection pressure, and (iii) the effect of modifying the population size. Experiments were run on both test problems (Wing-Box and Knapsack). For analyzing the results, it is worth remembering that the Wing-Box is a minimization problem whereas the Knapsack is a maximization problem.

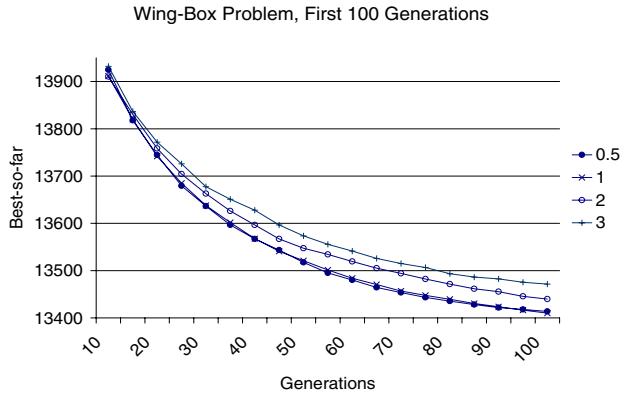
4.1 TIME-DEPENDENCY

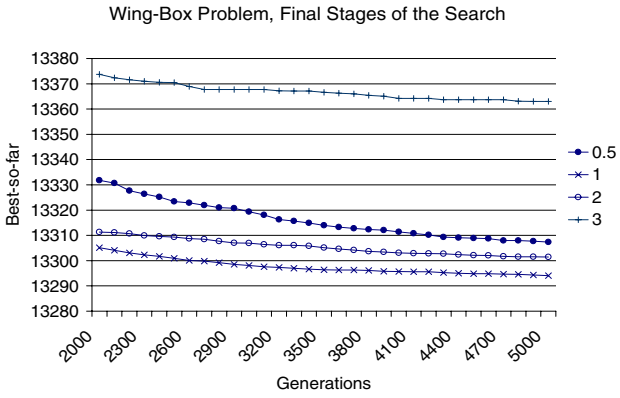
The first set of experiments studies the behavior of different mutation values over the generations of a GA run. The evolutionary parameters used are those summarized in Table 1. Results are presented in three stages. First the "interesting" part of the search, from generation 100 to generation 2,000 (Figure 3). Then, the first stage of the search, the first 100 generations (Figure 4); and, finally, the last stages of the search, from generation 2,000 to 5,000 (Figure 5). The plots show the average best-so-far fitness attained over fixed intervals throughout the GA run on both test problems (the Wing-Box and Knapsack problems). Four mutation values were explored: 0.5, 1.0, 2.0, and 3.0 mutations per genotype. Standard deviations are not shown in these plots for the sake of clarity.

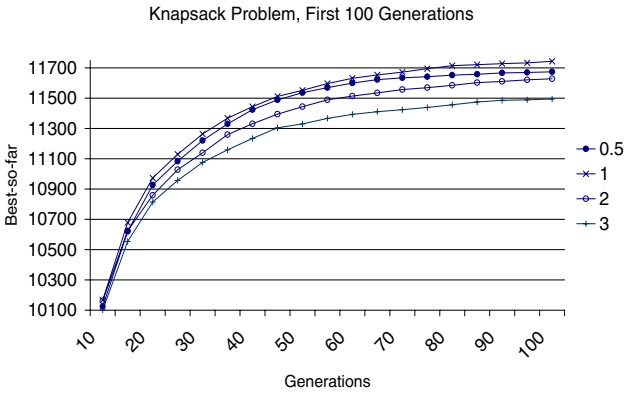
For the intermediate stage of the search, on the Wing-box problem the mutation rates of $1/L$ and $2/L$ produced the best results and performed similarly (Figure 3, top). On the Knapsack problem, a mutation rate of $1/L$ seems to produce the best performance in this stage (Figure 3, bottom).

The initial stage of the search is rather similar for both test problems (Figure 4, recall that the Wing-Box problem is a minimization problems whereas the Multiple Knapsack is a maximization problem) . All the mutation values explored performed similarly. However, the mutation values of 0.5 and 1.0 mutations per genotype seem to produce the best results in this stage.

Again, the final stage of the search is rather similar for both test problems. A mutation rate of $1/L$ produced the best performance in both cases (Figure 5).







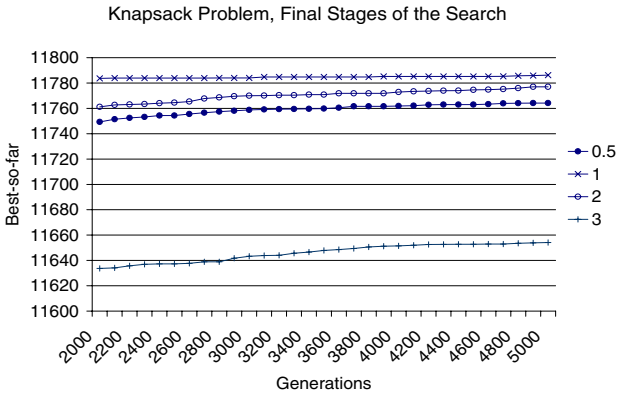


Figure 4: Comparing the performance of different mutation rates over a GA run on both test problems for the first 100 generations. The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

Figure 5: Comparing the performance of different mutation rates over a GA run on both test problems for the final stages of the search (from generation 2,000 until 5,000). The curves show the average best-so-far fitness over fixed intervals throughout the GA run for various mutation rates (expressed as mutations per genotype).

4.2 SELECTION PRESSURE

This subsection explores the effect of increasing the selection pressure on the magnitude of optimal mutation rates. The experiments use tournament selection because this scheme allows the selection pressure to be explicitly controlled. A common tournament size is 2, but selection pressure increases steadily for growing tournament sizes. Two tournament sizes, 2 and 4, were tested. Additionally, on the Knapsack problem, results using proportional selection are also presented for the sake of comparison. Figure 6 compares optimal mutation rates (per genotype) on the two selected test problems. The strength of selection had a noticeable effect on the magnitude of optimal mutation rates: on the Wing-box problem and for a tournament size of 2, the optimal mutation rate was around $1.0 - 2.0$ mutations per genotype, whereas for a tournament size of 4 it was around $2.5 - 3.0$ mutations per genotype. Similarly, on the Knapsack problem the optimal mutation values were around $1.5/L$ for tournament size of 2; and around $2.0 - 3.0$ for tournament size of 4. Moreover, the curve using proportional selection on the Knapsack problem (Figure 6, bottom), strikingly shows the difference in magnitude of optimal mutation

rates for a weak selection pressure. In this case, the optimal mutation rate was as low as 0.05 mutations per genotype.

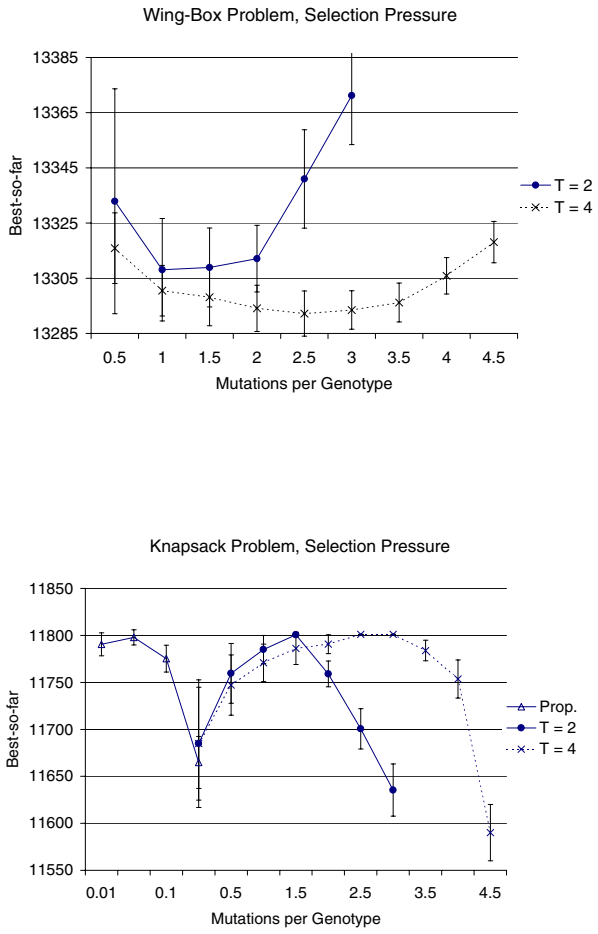


Figure 6: Comparing optimal mutation rates (per genotype) for different selection pressures on the two test problems. Tournament selection with two tournament sizes (2 and 4) was tested. Additionally, proportional selection was tested on Knapsack problem. The curves show the average best-so-far fitness attained after 2,000 generations for various mutation rates.

4.3 POPULATION SIZE

This subsection explores the effect of modifying the population size on the magnitude of optimal mutation rates. Three population sizes: 10, 50, and 100, were tested. The number of generations used as a stop criterion varied according to the population size since the smaller the population, the more generations were needed for equilibrating the best-so-far fitness. So the termination criteria used were 20,000, 4,000, and 2,000 generations for population sizes 10, 50, and 100 respectively. Figure 7 shows results on the two selected test problems. Optimal mutation rates tended to be smaller, the smaller the population size, this tendency was clearer on the Knapsack problem (bottom plot), where optimal mutation rates were around $0.5-1.0/L$ for a population size of 10, and around $1.0-1.5/L$ for population sizes of 50 and 100. Notice that for population sizes of 50 and 100, differences in performance for the various mutation rates tend to stabilize. This was also the case for preliminary experiments on larger populations.

5 DISCUSSION

This paper has been a first attempt to explore the validity of the heuristic suggesting a mutation rate of $1/L$ for GAs with bit-string encoding. Two completely unrelated and complex real-world domains were selected as test problems. Some common behaviors were found, so these findings may convey some generality. Three aspects were studied: (i) the time-dependency of the mutation rate, and the effect (on the magnitude of optimal mutation rates) of modifying (ii) the selection pressure, and (iii) the population size. Our main results are summarized below:

Time-Dependency: It has been suggested elsewhere that mutation rates should not be constant, but should decrease over the GA run. Results in this paper, however, suggest that a mutation rate of $1/L$ will produce optimal or near optimal results throughout the whole search process. So, on the specific but rather standard GA settings used here (generational GA, population size of 100, two-point recombination with a rate of 1.0, tournament selection of size 2, best-so-far fitness as performance measure), a constant mutation regime with a rate of $1/L$ would produce very competitive results.

Selection pressure: The strength of selection had a pronounced effect on optimal mutation rates. The stronger the selection pressure, the higher the magnitude of optimal mutation rates. The use of proportional selection (where there is no control over the selection pressure) may produce much smaller optimal mutation rates as compared to tournament selection. An interesting observation is that for tournament selection with tournament size of 2 (and a population of size 100), optimal mutation rates occurred between 1.0 and 2.0 mutations per genotype, whereas for tournament size of 4 they increased to 2.5 - 3.0 mutations per genotype (Figure 6). This result suggests that se-

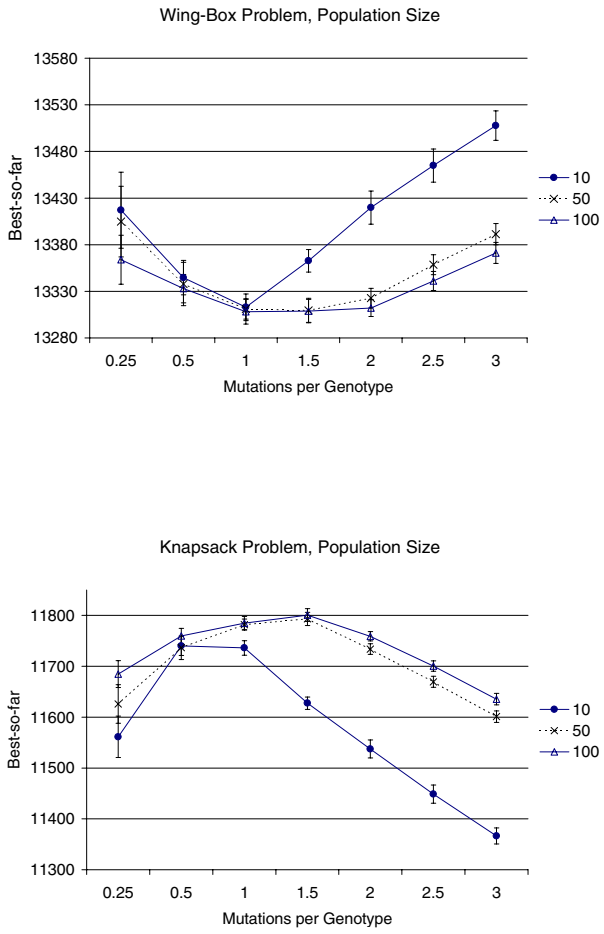


Figure 7: Comparing optimal mutation rates for various population sizes (see legends) on both test problems. The curves show the average best-so-far fitness attained after a fixed number of generations for various mutation rates. These fixed number of generations varied according the population size (20,000, 4,000 and 2,000 generations for population sizes 10, 50, and 100 respectively).

lection pressure is an important component in determining the magnitude of optimal mutation rates.

Population size: The effect of population size on the magnitude of optimal mutation rates was not found to be marked. However, the evidence suggests that optimal mutation rates are smaller, the smaller the population size. These differences in the magnitude of optimal mutation rates tend to stabilize for population sizes of 50 and larger.

6 CONCLUSION

It is very difficult to suggest general principles for setting evolutionary parameters. The evidence gathered in this paper, however, suggest that for a controlled selection pressure (tournament selection, with tournament size of 2), a mutation rate of $1/L$ throughout the whole GA run, will be a good setting, producing optimal or near-optimal results. In general, we suggest that mutation rates should be expressed as mutations per genotype instead of as mutations per bit.

The heuristic of setting a mutation rate of one mutation per genotype ($1/L$) has been proposed before within the evolutionary computation community. However, results in this paper set bounds to the validity of this heuristic. A mutation rate of $1/L$ would be sub-optimal in the following cases:

- a weak selection pressure,
- an excessively high selection pressure, and
- a very small population.

ACKNOWLEDGEMENTS

Many thanks to the anonymous GECCO reviewers for their useful comments, insight and critical reading.

References

[Bäck, 1991] Bäck, T. (1991). Self-adaptation in genetic algorithms. In Varela, F. J. and Bourgine, P., editors, *Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA.

[Bäck, 1992] Bäck, T. (1992). The interaction of mutation rate, selection, and self-adaption within a genetic algorithm. In und R. Manderick, B. M., editor, *Parallel Problem Solving from Nature 2*. North-Holland.

[Bäck, 1993] Bäck, T. (1993). Optimal mutation rates in genetic search. In Forrest, S., editor, *Proceedings of the 5th ICGA*. Morgan Kaufmann.

[Bäck, 1996] Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. The Clarendon Press Oxford University Press. Evolution strategies, evolutionary programming, genetic algorithms.

[Bäck and Khuri, 1994] Bäck, T. and Khuri, S. (1994). An evolutionary heuristic for the maximum independent set problem. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 531–535. IEEE Press.

[Beasley, 1990] Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072. Also available at http://www.ms.ic.ac.uk/info.html.

[Bremerman et al., 1966] Bremerman, H., Rogson, M., and Salaff, S. (1966). Global properties of evolution processes. In *Natural Automata and Useful Simulations*, pages 3–41. Spartan.

[Davis, 1989] Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, George Mason University. Morgan Kaufmann.

[DeJong, 1975] DeJong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI. Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381.

[Fogarty, 1989] Fogarty, T. C. (1989). Varying the probability of mutation in the genetic algorithm. In Schaffer, J. D., editor, *Proceedings of the 3rd ICGA*. Morgan Kaufmann.

[Hesser and Männer, 1991] Hesser, J. and Männer, R. (1991). Towards an optimal mutation probability for genetic algorithms. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature*. Springer-Verlag, Lecture Notes in Computer Science Vol. 496.

[Julstrom, 1995] Julstrom, B. A. (1995). What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87, San Francisco, CA. Morgan Kaufmann.

[Khuri and Bäck, 1994] Khuri, S. and Bäck, T. (1994). An evolutionary heuristic for the minimum vertex cover problem. In Hopf, J., editor, *Genetic Algorithms within the Framework of Evolutionary Computation*, pages 86–90, Saarbrücken, Germany. Max-Planck-Institut für Informatik.

[Khuri et al., 1994] Khuri, S., Bäck, T., and Heitkötter, J. (1994). The zero/one multiple knapsack problem and genetic algorithms. In Deaton, E., Oppenheim, D., Urban, J., and Berghel, H., editors, *Proceedings of the 1994 ACM Symposium of Applied Computation*, pages 188–193. ACM Press.

[McIlhagga et al., 1996] McIlhagga, M., Husbands, P., and Ives, R. (1996). A comparison of search techniques on a wing-box optimisation problem. *Lecture Notes in Computer Science*, 1141.

[Mühlenbein, 1992] Mühlenbein, H. (1992). How genetic algorithms really work: I. mutation and hillclimbing. In Männer, B. and Manderick, R., editors, *Parallel Problem Solving from Nature 2*. North-Holland.

[Ochoa et al., 1999] Ochoa, G., Harvey, I., and Buxton, H. (1999). Error thresholds and their relation to optimal mutation rates. In Floreano, J., Nicoud, D., and Mondada, F., editors, *Proceedings of the Fifth European Conference on Artificial Life (ECAL'99)*. Springer-Verlag.

[Ochoa et al., 2000] Ochoa, G., Harvey, I., and Buxton, H. (2000). Optimal mutation rates and selection pressure in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 315–322.

[Schaffer et al., 1989] Schaffer, J., Caruana, R., Eshelman, L., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer, J. D., editor, *Proceedings of the 3rd ICGA*, San Mateo CA. Morgan Kaufmann.

[Smith and Fogarty, 1996] Smith, J. E. and Fogarty, T. C. (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 318–323, New York. IEEE Press.

[Tuson and Ross, 1998] Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184.

A Comparison of Two Competitive Fitness Functions

Liviu Panait
George Mason University
http://www.cs.gmu.edu/∼lpanait/

Sean Luke
George Mason University
http://www.cs.gmu.edu/∼sean/

Abstract

Competitive fitness is the assessment of an individual's fitness in the context of competition with other individuals in the evolutionary system. This commonly takes one of two forms: one-population competitive fitness, where competition is solely between individuals in the same population; and N-population competitive fitness, often termed competitive coevolution. In this paper we discuss common topologies for one-population competitive fitness functions, then test the performance of two such topologies, Single-Elimination Tournament and K-Random Opponents, on four problem domains. We show that neither of the extremes of K-Random Opponents (Round Robin and Random-Pairing) gives the best results when using limited computational resources. We also show that while Single-Elimination Tournament usually outperforms variations of K-Random Opponents in noise-free problems, it can suffer from premature convergence in noisy domains.

1 INTRODUCTION

Traditional evolutionary computation assesses the fitness of an individual independently of other individuals in the system. But there also exist evolutionary procedures where this is not the case: an individual's fitness is dependent on cooperation or competition with peers in the evolutionary run, and thus may change depending on the makeup of those peers.

Such procedures have several attractive features. First, they permit evolution to search for solutions to problems in the absence of any obvious way to gauge an objective (peer-independent) fitness. Consider: how does one determine the quality of a soccer player program *a priori?* Second, they can gradually ramp up problem difficulty as evolution finds better solutions. This promises to smooth out the search gradient. Third, they seem a natural match for finding solutions to problems that naturally require teamwork or that are most easily discovered through competition.

We are tempted to bring all these procedures under the aegis of *coevolution*, but there are nomenclature difficulties with the use of this term. In biology, coevolution is best reserved for situations where there is more than one population, and an individual's fitness is assessed in the context of individuals in other populations. Such multi-population coevolution is usually used as a self-adaptive mechanism to increase problem difficulty as members of the population become more adapt at solving the given problem. The classic example of multi-population competitive coevolution is [Hillis 1991], which coevolved a population of sorting networks and a population of problem sets. The fitness of sorting networks was based on the number of problem sets they properly solved, and the fitness of the problem sets was based on the number of sorting networks they stumped. [Rosin and Belew 1995] also used a two-population competitive system to evolve players for the games of Nim, Tic-Tac-Toe, and Go with a 7x7 board. Multi-population coevolution is also useful as *cooperative coevolution*. Here individuals from different populations each learn subparts of a common solution, and their fitness is based on the combination of those subparts. Examples of cooperative coevolution include [Eriksson and Olsson 1997; Potter and De Jong 2000; Wiegand et al. 2001].

One-population "coevolution" rarely if ever takes cooperative form. Instead, this technique is nearly universally used to evolve game players by competing amongst themselves. For lack of a standardized term for one-population techniques, we call these *one-population competitive fitness functions*; for the rest of this paper, whenever we say "competitive fitness functions" we imply the one-population sort. [Luke 1998] used such competitive fitness to evolve soccer-playing softbot teams, and [Fogel 2001] used the technique to evolve a highly human-competitive check-

ers program, Blondie24. One-population competitive fitness has also been used to find solutions to the Iterated Prisoner's Dilemma [Axelrod 1987], Tic-Tac-Toe [Angeline and Pollack 1993], Backgammon [Pollack et al. 1997; Pollack and Blair 1998], Othello [Smith and Gray 1993], pursuit-evasion [Cliff and Miller 1995], Go [Lubberts and Miikkulainen 2001] and Tag [Reynolds 1994].

One important part of a competitive system's success is its *topology:* how the fitness-evaluation context is established for a given individual. Do all individuals play against all other individuals in the population? Are they simply paired up for a single game each? Some topologies require a large number of games to evaluate an individual, but may be more accurate than those requiring fewer games.

This paper compares two topological families in one-population competitive-fitness games. We begin by discussing common topologies in the literature and their advantages and disadvantages. Then we introduce four problem domains, and show how various topologies fare in these domains and under different amounts of noise in the fitness-assessment process.

2 COMPETITION TOPOLOGIES

Not all competitive fitness topologies are appropriate for all problems; the primary issue breaks down along lines of fitness-assessment methodology. Imagine if one were trying to evolve chess players. How does one establish that player A is better than player B? The *duel methodology* states that A is better than B if and only if A usually beats B in a match. This is the methodology behind single- and double-elimination tournaments. The *rennaisance-man methodology* says that A is better than B if A beats more competitors than B does on average (or scores against competitors by a wider margin on average), even if A would lose to B in a match. This is the methodology behind chess rankings, for example. It is interesting to note that many sports use a combination of these two methodologies, usually by using average success against opponents during the season to determine the entrants to a single elimination tournament, which then determines the final champion. Whether there is some innate superiority to this combination is questionable: more likely it is due to the excitement of duels: after all, "in the end there can be only one".

There are other interesting issues in designing topologies which we will not delve into save to mention them here. One issue is whether or not individuals should play against themselves as part of their evaluation. Another is whether or not to permit statistical dependencies in fitness assessment: when individual A plays against individual B, should the outcome affect individual A's fitness alone, or should it also affect individual B's fitness?

2.1 ROUND ROBIN

One simple topology is *Round Robin*, where each individual plays every other individual in the population. An individual's fitness is the average of its scores against every other individual in the population [Axelrod 1987; Koza 1992]. The primary drawback to this method is the relatively large number of games necessary to evaluate a population of size N. The number of games is $(N^2 - aN)/b$, where $a = 0$ if individuals may play against themselves, else $a = 1$, and $b = 2$ if a game contributes to the fitness of both individuals, else $b = 1$. At first glance it would appear that Round Robin topologies would promote the rennaisance-man methodology. At the beginning of an evolutionary run, this is plausible. But as the run progresses, the trajectory of the run might shift to the "better" players, so to speak, so that near the end of the run it is searching not for individuals who win the most points on average, but oddly for individuals who win the most points on average against *other such individuals.*

2.2 RANDOM-PAIRING

The other extreme in the number of games is to pair all individuals up and play one game for each pair. This is the approach used in [Luke 1998] for evolving soccer team strategies. The justification for this low number of games was the extreme computational cost of a game: to be evaluated, the two teams were plugged in a simulator, and a standard game could last for up to 10 minutes. Random-Pairing requires only $N/2$ games for a population of size N. The cost savings is dramatic: for a population of 100 and ten minutes per game, Axelrod's Round Robin approach would require 833 hours per generation, whereas Random-Pairing would require about 8 hours. Smith and Gray [1993] also used this technique to evolve Othello players. The danger of Random-Pairing is that noisy evaluation might make it all but impossible to determine the real quality of an individual based on a single trial. Note too that like Round Robin, Random-Pairing has a similar tenuous claim to promoting the rennaisance-man methodology.

2.3 SINGLE-ELIMINATION TOURNAMENT

[Angeline and Pollack 1993] proposed using single-elimination tournaments ("SET") rather than Round Robin or Random-Pairing. Here, individuals are paired at random, and play one game per pair. The losers of the games are eliminated from the tournament; ties are broken by random decision. The winners are again paired off at random, and play one game per pair, with the losers again eliminated. This continues until the tournament has only one "champion" left. The fitness of an individual is the number of games it played. Single-Elimination Tournament is sim-

plest to implement when the population is a power of two. Angeline and Pollack reported good initial results when using SET to evolve players for the game of Tic-Tac-Toe.

SET has interesting properties. First, it would seem to promote the duel methodology rather than the rennaisance-man methodology. However, it only truly promotes the duel methodology under the *strong transitivity assumption:* that if player A beats player B, and player B beats player C, then player A must beat player C. Without this assumption, Single-Elimination Tournament's real dynamics can be murky. The other interesting property of SET is that it seems to allocate games to those players that most need them. A population of size N needs only $N-1$ games. But "fitter" players will be evaluated in more of these games than the "less fit" players — the worst individuals play only one game each, while the champion plays $\ln(N)$ games. Since selection will tend to pick the fitter players, SET would seem to proportion more games, hence more accuracy, among those players more likely to be selected.

2.4 K-RANDOM OPPONENTS

In K-Random Opponents, each individual plays against K individuals picked at random from the population. If a given game between two individuals affects the fitness of just the first individual, then a total of $K(N-1)$ games must be played. This is the approach taken in evolving tag players [Reynolds 1994]. K-Random Opponents can also be used to affect the fitness of both individuals in a game. For example, to evolve the Blondie24 checkers player, Fogel [2001] had every individual play as red against five opponents chosen at random with replacement from the population. An individual's fitness was based not only on its five games as red, but also as its additional games as a black opponent.

This approach does not distribute games very evenly throughout the population, however. With some forethought, it's possible to adapt K-Random Opponents so that a given game affects both individuals, with each individual using the same number of games per evaluation. The technique, which we will use in experiments below, works as follows. Each individual maintains a count of the number of games it has played, and who it has played against. When an individual I is to be evaluated, an opponent is chosen at random from the population to play against I with the constraint that no individual may play against I more than once. At the end of the game, the number-of-games counters for I and for the opponent are incremented. If either counter reaches K, then that individual is "removed" from the population in the sense that it may no longer be considered as a future opponent. A new opponent for I is chosen, and this process continues until individual I has been removed. Then a new player J is picked, and evaluation continues similarly. At some point, for some individual K, there may exist no individuals in the population which can play K. When this occurs, opponents for K are picked at random, without replacement, from among the removed individuals in the population. This approach yields between $\lceil (KN)/2 \rceil$ and $\lceil (KN)/2 \rceil + \lfloor K^2/2 \rfloor$ games.

Round Robin and Random-Pairing may be viewed as extremes of K for this second kind of K-Random Opponents. When $K = N-1$, K-Random Opponents is identical to Round Robin. When $K = 1$, K-Random Opponents is identical to Random-Pairing. Later in the paper, we will examine K-Random Opponents to determine what value of K seems to give the best results: as it will turn out, it is neither of these extremes.

2.5 HALL OF FAME

One last approach in the literature is a family of "hall of fame" techniques, where individuals in the population are evaluated against the good individuals discovered so far in the evolutionary run. Karl Sims used a simple hall of fame when evolving creatures which competed to snatch a cube [Sims 1994]. Individuals were evaluated against the fittest individual discovered in the previous generation.

3 PROBLEM DOMAINS

The problem domains we will test against fall into two categories. First, we use two true competitive fitness domains, namely versions of the Nim game. Second, we have adapted two standard evolutionary algorithm problems and cast them into a competitive fitness form. They are the well-studied Rosenbrock and Rastrigin problem sets. These algorithms are cast into competitive form using a technique proposed by Ken De Jong: each individual's Rosenbrock (or Rastrigin) value is assessed, and an individual's score in a game against an opponent is based on difference in their values.

3.1 THE INTERNAL ROSENBROCK DOMAIN

The Rosenbrock function is a well-known minimization problem widely used to study properties of different evolutionary algorithms [De Jong 1975]. The Rosenbrock function for genomes of n variables is:

$$Ros(x_1,...,x_n) = \sum_{i=1}^{n} 100({x_i}^2 - x_{i+1})^2 + (1-{x_i}^2)$$

Rosenbrock is converted to the "Internal Rosenbrock" competitive fitness function as follows. When a player A plays an opponent B, the score for A, known as $Reward(A:B)$, is given by the following normalizing formula:

$$Reward(A:B) = \frac{Ros(B) - Ros(A)}{\max(Ros) - \min(Ros)}$$

...where $\max(Ros)$ and $\min(Ros)$ are the maximum and minimum values of the Rosenbrock function over the entire domain, which we had precomputed. Thus $Reward(A:B)$ ranges from -1 to 1, where 0 represents a draw. Note that this is a zero-sum, transitive game, hence $Reward(B:A) = -Reward(A:B)$. Keep in mind that Rosenbrock is a minimization function: therefore the smaller $Ros(A)$ is compared to $Ros(B)$, the higher the reward for A.

Parameters Internal Rosenbrock experiments used a genome of 100 real values each between -5.12 and 5.12, a population size of 32, a 0.5 probability of mutation, 1-point crossover with a probability of 1.0, 5-individual elitism, binary tournament selection, and a maximal run limit of 50,000 games.

3.2 THE INTERNAL RASTRIGIN DOMAIN

The Rastrigin function is another well-known test in function optimization; it is considered difficult to minimize because it has a single global optima with numerous local optima in its vicinity [Cervone et al. 2000]. The Rastrigin function is defined as

$$Rastrigin(x_1....x_n) = \sum_{i=1}^{n} {x_i}^2 + a(1 - \cos(2\pi x_i))$$

...where a is a constant (set to 10.0 in our experiments). Like Rosenbrock, Rastrigin is a minimization problem. Rastrigin is converted to the "Internal Rastrigin" competitive function in exactly the same way as Rosenbrock was converted (though $\max(Ras)$ was estimated).

Parameters Internal Rastrigin experiments used a genome of 100 real values each between -5.12 and 5.12, a population size of 32, a 0.5 probability of mutation, 1-point crossover with a probability of 1.0, 5-individual elitism, binary tournament selection, and a maximal run limit of 100,000 games.

3.3 THE NIM VERSION 1 DOMAIN

There are many variations on the game of Nim, and we have chosen two different versions as competitive fitness function domains. The Nim Version 1 domain follows the Nim game as described in [Rosin and Belew 1995, 1996]. This version uses 4 heaps containing 3, 4, 5, and 4 stones respectively. Players take turns removing stones from these heaps. A player may remove as many stones as he likes from any single heap. Whichever player takes the last stone wins the game. Given these rules, there exists a well-understood optimal player strategy for the first player.

A genomic representation for a player behavior in this game is a vector of 599 bits, one for each possible situation ($4 \times 5 \times 6 \times 5 - 1$, because the $\langle 3,4,5,4 \rangle$ position does not ever need to be considered). A player makes its decision as follows: for each pile p from 1 to 4, and for each number x of stones for the given pile in decreasing order down to 1, the individual considers whether or not to remove x stones from pile p. Removing these stones yields a new game state which corresponds to one of the 599 bits in the genome vector. If this bit value is 1, then the player commits to making that move, and no other consideration is made. If all such valid states have 0 bit values, the player makes the first valid move it had considered.

As the existence of a perfect strategy depends on who goes first, a competition between two individuals consists of 2 games, each player starting one of them. $Reward(A:B)$ is the sum of scores for player A in these two games. For each game, a 0.5 is rewarded for a win and a -0.5 for a loss. The sum of the rewards for the two games is therefore -1, 0, or 1.

Parameters Experiments in this domain used a genome of 599 bits, a population size of 128, a 0.003 probability of mutation, 1-point crossover with a probability of 1.0, 10-individual elitism, binary tournament selection, and a run limit of no more than 100,000 games.

3.4 THE NIM VERSION 2 DOMAIN

The second version of Nim used in this paper contains a single heap, but the number of stones a player can remove is bounded by a minimum and a maximum value. For these experiments, the heap starts at 200 stones, and each player is allowed to pick 1, 2 or 3 stones at a time. In this configuration, the second player has an optimal strategy which will force a win.

Just as in Nim Version 1, in this game the individuals are represented as vectors, with a similar mapping of bits to the 199 possible states (excepting the initial state). Decision-making is also similar. The player first considers removing 3 stones (assuming that 3 stones are left in the heap). If 1

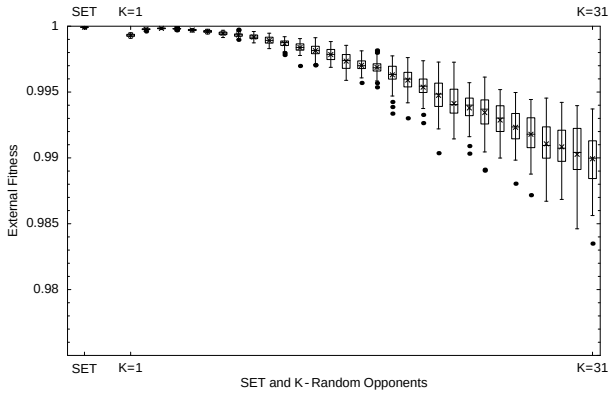


Figure 1: Ranking of SET and K-Random Opponents for Internal Rosenbrock Domain with 0% noise

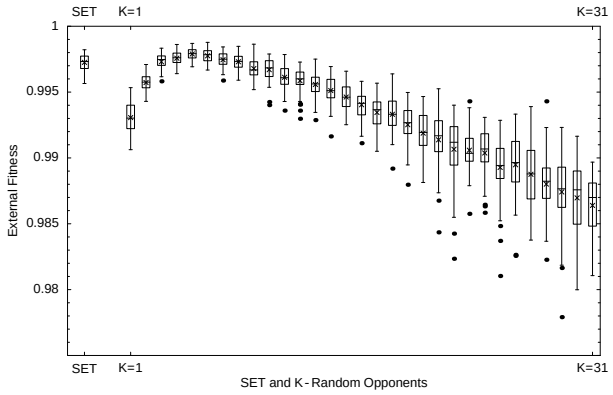


Figure 2: Ranking of SET and K-Random Opponents for Internal Rosenbrock Domain with 30% noise

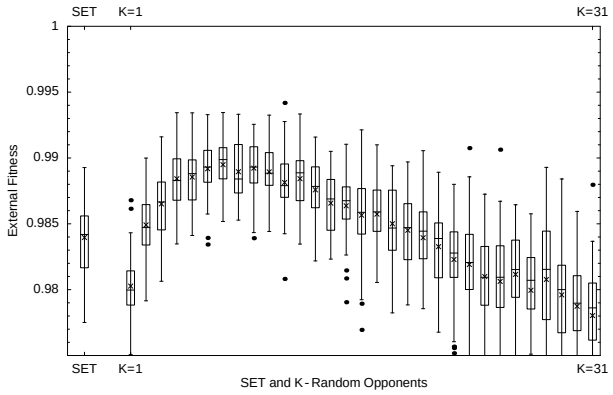


Figure 3: Ranking of SET and K-Random Opponents for Internal Rosenbrock Domain with 40% noise

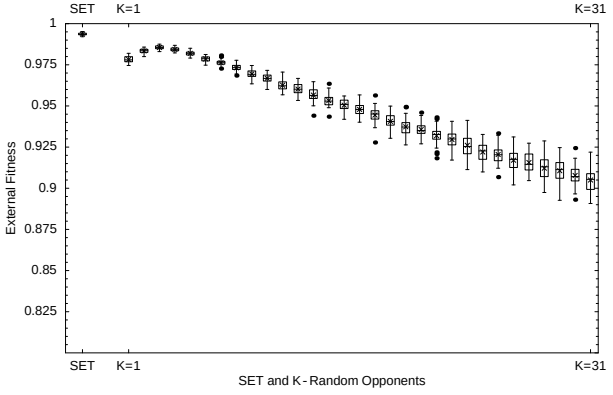


Figure 4: Ranking of SET and K-Random Opponents for Internal Rastrigin Domain with 0% noise

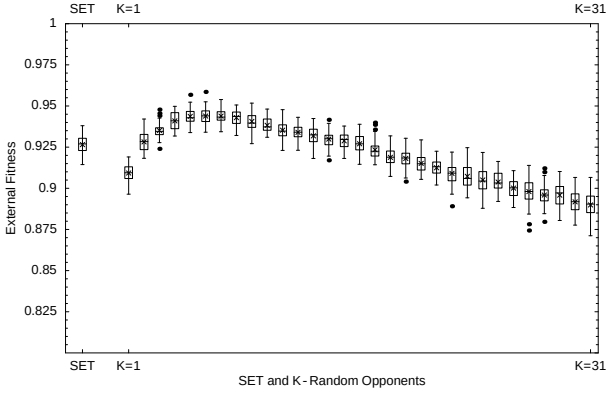


Figure 5: Ranking of SET and K-Random Opponents for Internal Rastrigin Domain with 30% noise

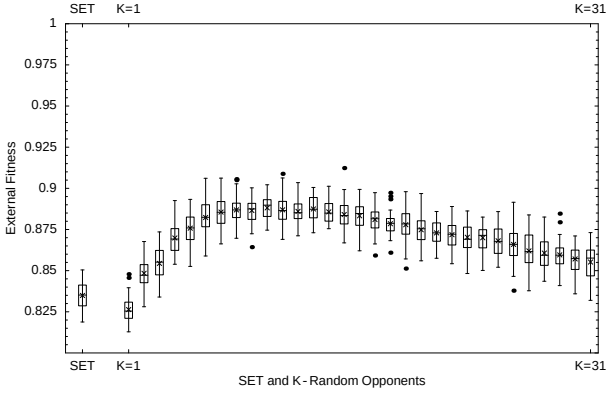


Figure 6: Ranking of SET and K-Random Opponents for Internal Rastrigin Domain with 40% noise

is in the bit position corresponding to the resulting state after removing those 3 stones, then the player will make that move. Otherwise, the player considers removing 2 stones. Barring that, it will consider removing 1 stone. If all three resultant states have 0 in their bit positions, then the player will remove the most stones permissible. A competition between two individuals is done identically to the Nim Version 1 problem.

Parameters Experiments in this domain used a genome of 199 boolean values, a population size of 128, a 0.03 probability of mutation, 1-point crossover with a probability of 1.0, 10-individual elitism, binary tournament selection, and a run limit of no more than 100,000 games.

4 EXPERIMENTS

The experiments presented here probe the following question. You have 3 months until the deadline to submit an evolved game player to a computer gaming competition. Evaluation is expensive and you'll only get one shot. With a fixed maximum number of games playable until competition-time, what topologies are likely to get good results?

We will compare SET and various K-random opponents topologies over the four problem domains, using a single-population, generational genetic algorithm, with binary tournament selection, mutation, crossover, and elitism. Experimental runs are done by evaluating individuals up to some maximal number of games; the maximal number was previously specified in the parameters for each domain. Keep in mind that an *evaluation* is not the same thing as a *game*. Some topologies require a great many games played before an individual's fitness is determined. Thus each graph compares different topologies' performances given the same number of resources.

Ultimately we are trying to determine what topology is likely to give the "best results". To compare topologies, we need a final *external fitness* used for comparing best-of-run results between topologies, as opposed to the subjective *internal fitness* used to select individuals during the runs themselves. For the Internal Rosenbrock and Internal Rastrigin problems, the external fitness of an individual is clearly objective and clearly computable: it's just the individual's performance on the Rosenbrock or Rastrigin functions.

For the Nim games however, we are faced with the classic external-fitness conundrum: the only obvious external fitness measures available are subjective, that is, they're determined in the context of other individuals. In the absence of any clear objective measure, we must resort to a subjective way to score the final performance of the best-of-run individuals for any given Nim topology. To do this, our approach is to determine the better topologies by literally playing their "best" individuals against each other. For any given application of a topology, we perform 50 independent runs. For each run, we determine a "best of run" by taking the best-of-generation individuals from each generation, and placing them in a single elimination tournament. Thus for each application of a given topology, we have 50 best-of-run individuals. To compare several topologies for a particular problem domain, we then take the 50 best-of-run individuals of each topology and play all of them in a Round Robin tournament. The "quality" of a best-of-run individual in the final tournament is equal to its average score against others in the tournament. Thus the "quality" of a particular topology is the mean of the qualities of its best-of-run individuals. This may not necessarily be an ideal comparison metric (we don't know if an ideal even exists), but we feel it is a reasonable one.

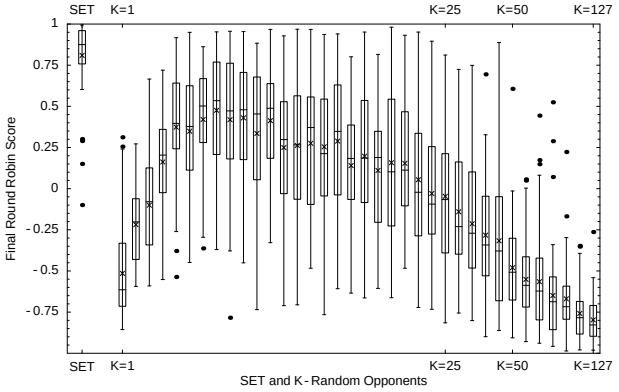


Figure 7: Ranking of SET and K-Random Opponents for the Nim Version 1 game

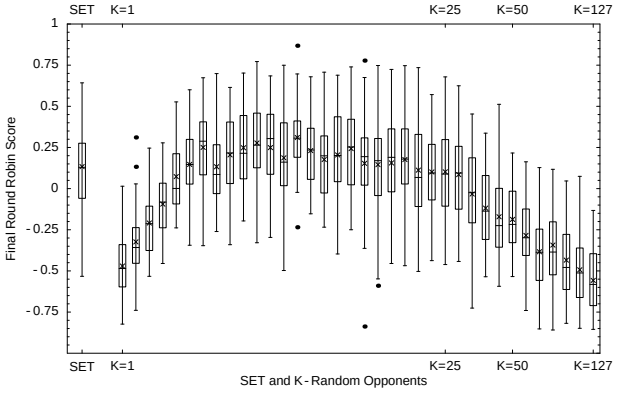


Figure 8: Ranking of SET and K-Random Opponents for the Nim Version 2 game

4.1 RESULTS

We ran all experiments on the ECJ 7 evolutionary computation system [Luke 2001]. Figures 1 through 8 show boxplots[1] comparing SET with K-Random Opponents. Figures 1 through 6 use values of K ranging from 1 to 31; Figures 7 and 8 use K values of 1 to 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 127. The vertical access plots external fitness values of the best-of-run individuals for various topologies. In the Rosenbrock and Rastrigin domains, the external scores were the actual Rosenbrock or Rastrigin function values for the best-of-run individuals. In the Nim domains, the final Round Robin competition to determine external scores consisted of every best-of-run individual plotted in the combined graph.

Figures 2 and 3 show the effects of adding noise to the Rosenbrock domain, and Figures 5 and 6 show similar effects for the Rastrigin domain. Noise was added by flipping a coin with the given noise probability that the players' scores were to be swapped. Noise was *not* used in the display of external fitness results.

K-Random Opponents Results We found that the overall layout of the graphs is very similar across all four domains: as the value of K increased, external fitness rose, then dropped. The dome-like results for K-Random Opponents suggests that neither Random-Pairing (where $K = 1$) nor Round Robin (where K is large) is likely to yield a good result. Indeed, we imagine that Round Robin will often come in dead last! In the Internal Rosenbrock and Internal Rastrigin domains with no noise, Random-Pairing performed reasonably well, but with more noise, it did increasingly poorly.

Why is this happening? Our hypothesis is that in noisy or intransitive domains, only a few games per evaluation is not sufficient to cut through the noise, and evolution proceeds slowly. Then as the number of games per evaluation increase, at some point it becomes overkill: more games are simply cutting the total available evolution time.

This result is similar to the one obtained for non-coevolutionary EAs [Grefenstette and Fitzpatrick 1985] when determining the optimal number of evaluations of an individual in a noisy environment, where the fitness was calculated as the average of the results of several evaluations. Grefenstette and Fitzpatrick too reported that one sample might not provide enough information, while too many samples might not leave enough generations for good results when the total number of evaluations is bounded. They reported that ten samples per evaluation gave the best

[1] In a boxplot, the rectangular region covers all values between the first and third quartiles, the stems mark the furthest individual within 1.5 of the quartile ranges, and the center horizontal line indicates the median. Dots show outliers, and $\times$ marks the mean.

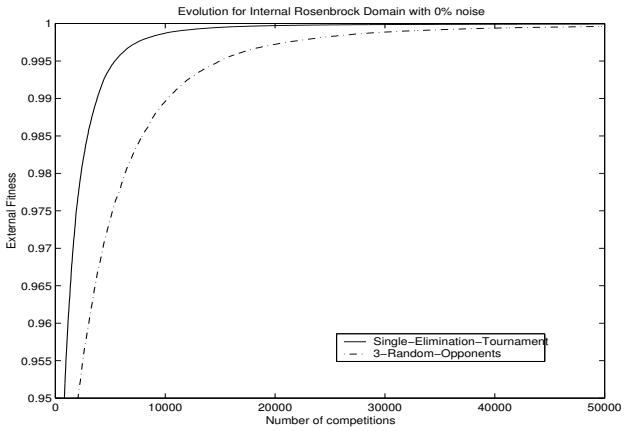


Figure 9: Best-so-far curves for Internal Rosenbrock Domain with 0% noise

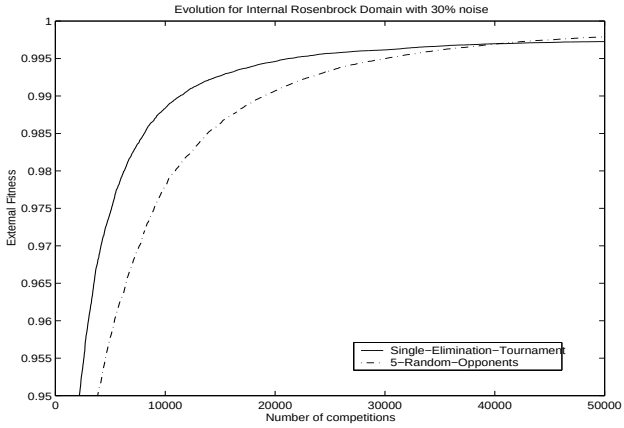


Figure 10: Best-so-far curves for Internal Rosenbrock Domain with 30% noise

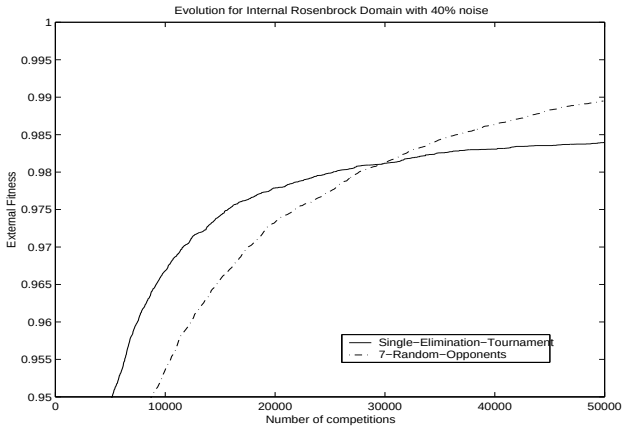


Figure 11: Best-so-far curves for Internal Rosenbrock Domain with 40% noise

results in an image registration problem. While we typically found fewer samples were necessary in our coevolutionary approach, ten gave reasonable results in most cases.

Single-Elimination Tournament Results The SET results were surprising. When the amount of noise is small, SET performs as good as or better than all other methods presented, even though it has relatively few games per evaluation. As noise is increased to 40% in the Rosenbrock domain, though, SET's performance loses its luster. Why?

Figures 9, 10 and 11 compare the external fitness best-so-far curves of SET and the best performing K-Random Opponents topology, with 0%, 30% and 40% noise respectively. These figures suggest that SET is converging too rapidly: as the field improves, this becomes a hindrance. In Figure 11, ultimately 7-Random Opponents is statistically significantly better (using a t-test at 95%).

It seems that K-Random Opponents *might* be a better choice than SET, particularly if noise is high. The trick, though, is determining what value of K to use. In the absence of any prescience, SET might be the best option.

5 CONCLUSIONS AND FUTURE WORK

Our experiments showed that the extremes of the K-Random Opponents method usually lead to worse final results than intermediate (preferably small) values for K. Even if games are very expensive, the concern that led to Random-Pairing in [Luke 1998], we still think 5 to 10 games per evaluation is likely to yield a better result. A full Round-Robin tournament appears to be always a bad choice. Our data suggests that the Single-Elimination Tournament may be too aggressive in noisy competitions, leading to premature convergence relative to 5- to 10-Random Opponents. Otherwise it seems to be a good choice.

Though many graphs are similar, nonetheless interesting features stand out. One surprise is the very strong performance of Single-Elimination Tournament in the Nim Version 1 game. This suggests dynamics special to this domain which, on closer investigation, may shed light on SET's performance in general. Does Nim Version 1 promote the duel methodology in a way not found in Nim Version 2, for example? Except for noise, the Internal Rastrigin and Internal Rosenbrock domains are fully transitive: might this explain the deterioration of SET under noise? In future work we hope to examine the dynamics of such topologies in these and other domains more closely.

Acknowledgments

The authors would like to thank Ken De Jong, Paul Wiegand and Jeff Bassett for helpful comments and suggestions, and Vlad Staicu for his considerable help in running the experiments. We would also like to thank our reviewers for their helpful comments.

References

Angeline, P. and Pollack, J. (1993). Competitive environments evolve better solutions for complex tasks. In Forest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, pages 264–270, San Mateo, CA. Morgan Kaufmann.

Axelrod (1987). The evolution of strategies in the iterated prisoner's dilemma. In Davis, L., editor, *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann.

Cervone, G., Michalski, R., Kaufman, K., and Panait, L. (2000). Combining machine learning with evolutionary computation: Recent results on LEM. In *Proceedings of the Fifth International Workshop on Multistrategy Learning*, pages 41–58.

Cliff, D. and Miller, G. F. (1995). Tracking the red queen: Measurements of adaptive progress in co–evolutionary sumulations. In *Proceedings of the Third European Conference on Artificial Life*, pages 200–218. Springer–Verlag.

De Jong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI.

Eriksson, R. and Olsson, B. (1997). Cooperative coevolution in inventory control optimisation. In Smith, G., Steele, N., and Albrecht, R., editors, *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*, University of East Anglia, Norwich, UK. Springer.

Fogel, D. (2001). *Blondie24: Playing at the Edge of Artificial Intelligence*. Morgan Kaufmann.

Grefenstette, J. J. and Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 37–46.

Hillis, D. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. *Artificial Life II, SFI Studies in the Sciences of Complexity*, 10:313–324.

Koza, J. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press.

Lubberts, A. and Miikkulainen, R. (2001). Co-evolving a Go-playing neural network. In *Coevolution: Turning Adaptive Algorithms upon Themselves, (Birds-on-a-Feather Workshop, Genetic and Evolutionary Computation Conference)*.

Luke, S. (1998). Genetic programming produced competitive soccer softbot teams for RoboCup97. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 214–222, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann.

Luke, S. (2001). ECJ 7: An evolutionary computation research system in Java. Available at http://www.cs.umd.edu/projects/plus/ec/ecj/.

Pollack, J. and Blair, A. (1998). Coevolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240.

Pollack, J., Blair, A., and Land, M. (1997). Coevolution of a backgammon player. In *Artificial Life V*. MIT Press.

Potter, M. and De Jong, K. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

Reynolds, C. (1994). Competition, coevolution and the game of tag. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pages 59–69. MIT Press.

Rosin, C. and Belew, R. (1995). Methods for competitive co-evolution: Finding opponents worth beating. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA)*, pages 373–380. Morgan Kaufmann.

Rosin, C. and Belew, R. (1996). New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pages 28–39. MIT Press.

Smith, R. and Gray, B. (1993). Co-adaptive genetic algorithms: An example in othello strategy. Technical Report TCGA 94002, University of Alabama, Department of Engineering Science and Mechanics.

Wiegand, R. P., Liles, W., and De Jong, K. (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In Spector, L., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001*, pages 1235–1242. Morgan Kaufmann.

Combining the Strengths of the Bayesian Optimization Algorithm and Adaptive Evolution Strategies

Martin Pelikan, David E. Goldberg, and Shigeyoshi Tsutsui
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
104 S. Mathews Avenue, Urbana, IL 61801
Phone/FAX: (217) 333-2346, (217) 244-5705
{pelikan,deg,shige}@illigal.ge.uiuc.edu

Abstract

This paper proposes a method that combines competent genetic algorithms working in discrete domains with adaptive evolution strategies working in continuous domains. We use discretization to transform solution between the two domains. The results of our experiments with the Bayesian optimization algorithm as a discrete optimizer and σ-self-adaptive mutation of evolution strategies as a continuous optimizer combined using k-means clustering suggest that the algorithm scales up well on all tested problems. The proposed method can be used to fill the gap between other optimization methods working in continuous and discrete domains and allow their hybridization.

1 INTRODUCTION

In genetic and evolutionary algorithms, the search is guided by selection and variation operators. Selection biases the search towards high-quality regions by making more copies of good solutions and less copies of the bad ones. Variation operators (such as recombination and mutation) ensure exploration of promising regions of the search space after applying selection. There are two commonly used variation operators: (1) recombination and (2) mutation. Genetic algorithms focus primarily on *recombination* that combines solutions by exchanging some of their parts. On the other hand, the dominant variation operator in evolution strategies is *mutation* that perturbs the solutions slightly. There has been a lot of progress in both mutation-based and recombination-based approaches over the last decades. However, only little has been done to combine the most advanced results of these two lines of research.

The purpose of this paper is to show how some of the advanced algorithms based on the two aforementioned approaches can be combined to solve problems defined in continuous domains. In particular, the Bayesian optimization algorithm (BOA) based on recombination is combined with a mutation-based evolution strategy (ES) with adaptive mutation strength. However, since BOA works only on finite-alphabet strings of fixed length while ES works directly with vectors of real numbers, it is not possible to combine the two approaches without an intermediate step in between. The problem of inconsistent representations is overcome by *discretization*. The resulting approach can be seen both as the Bayesian optimization algorithm with adaptive discretization or a recombinative evolution strategy capable of linkage learning. The same method can be used to combine other competent genetic algorithms and evolution strategies with no or only minor modifications and to solve problems that contain both continuous and discrete variables.

The paper starts by introducing the Bayesian optimization algorithm and evolution strategies with adaptive mutation, which are used as the basic building blocks of the proposed algorithm. Discretization is then discussed in context of genetic and evolutionary computation. Section 3 describes how a competent recombination-based genetic algorithm in a discrete domain can be combined with a mutation-based approach in a continuous domain. Section 4 provides our experimental results. Section 5 concludes the paper.

2 BACKGROUND

Genetic and evolutionary algorithms start with a randomly generated initial population of candidate solutions. In each iteration, the set of promising solutions is selected where the number of copies of each candidate solution is somehow proportional to the solution's quality. New candidate solutions are constructed by applying recombination and mutation operators to the selected solutions. The new solutions replace some of

the old ones or all of them and the process is repeated until the termination criteria are met.

This section starts by introducing two fundamentally different approaches based on the above scheme. First, it describes the Bayesian optimization algorithm (BOA), which is based on recombination and has been recently shown to solve boundedly difficult decomposable problems defined on fixed-length binary strings efficiently and reliably. BOA is capable of learning and exploiting a decomposition of the problem by analyzing the promising solutions. Subsequently, the section describes evolution strategies (ES) that process fixed-length vectors of real numbers and use mutation as the primary variation operator. Several methods for adapting mutation parameters are presented. The section ends by discussing discretization in context of genetic and evolutionary computation that will later be used as a way to bridge the recombination-based discrete BOA and the mutation-based continuous ES.

2.1 BAYESIAN OPTIMIZATION ALGORITHM

Recombination-based genetic algorithms generate new solutions by combining bits and pieces of promising solutions. The simple genetic algorithm (Goldberg, 1989) uses problem-independent crossover operators to combine promising solutions, such as uniform crossover and one-point crossover. Mutation is usually used as only a background operator capable of tuning near-optimal solutions at the end of the run or introduce diversity into the population.

Probabilistic model-building genetic algorithms (PMBGAs) (Pelikan, Goldberg, & Lobo, 2002) also try to combine important parts of the selected solutions but they approach recombination in a different way. They view the set of selected solutions as a sample from the region of the search space that we are interested in. PMBGAs first estimate the distribution of the selected solutions and then use this estimate to generate new solutions. The estimated distribution can encode the interactions among the different variables in the problem as well as superiority of certain combinations of values of different subsets of variables. The algorithms based on this principle are also called *estimation of distribution algorithms* (Mühlenbein & Paaß, 1996), or *iterated density estimation algorithms* (Bosman & Thierens, 2000). It is beyond the scope of this paper to give an overview of PMBGAs. For a survey of PMBGAs, please see Pelikan, Goldberg, and Lobo (2002).

In this paper, we focus on the Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1998) that uses Bayesian networks to model promising solutions and subsequently guide the exploration of the search space. The first population of strings is generated randomly with a uniform distribution. From the current population, the better strings are selected using one of the conventional selection methods such as tournament or truncation selection. A Bayesian network that fits the selected set of strings is constructed. Besides the set of good solutions, prior information about the problem can be used in order to enhance the estimation and subsequently improve convergence. New strings are generated according to the joint distribution encoded by the constructed network. The new strings are added into the old population, replacing some of the old ones.

A Bayesian network is a directed acyclic graph with the nodes corresponding to the variables in the modeled data set (in our case, to the positions in the solution strings) and the edges defining the conditional dependencies among the variables. A directed edge relates the variables so that in the encoded distribution, the variable corresponding to the terminal node is conditioned on the variable corresponding to the initial node. More incoming edges into a node result in a conditional probability of the corresponding variable with a conjunctional condition containing all its parents. To learn the network structure, a scoring metric, such as the Bayesian-Dirichlet metric or the Bayesian information criterion (BIC), can be used to discriminate competing models. A search procedure then searches the space of all potential network structures to find the one that scores the most. A greedy search procedure is often used that iteratively adds, removes, or reverses the edge that improves the score of the network the most until no more improvement is possible.

In BOA, the built Bayesian network encodes important interactions in the problem as well as its decomposition. The decomposition simplifies the problem, while the interactions allow the use of reasonably sized populations and convergence times. It has been shown that BOA is indeed capable of learning how to properly break up the problem to optimize the problems decomposable into subproblems of bounded difficulty in subquadratic or quadratic time.

2.2 EVOLUTION STRATEGIES

Evolution strategies (ES) (Rechenberg, 1973) use mutation as the driving force of the search and usually work on solutions represented by vectors of real numbers. Mutation is usually performed by adding a number generated according to a zero-mean normal distribution to the solution. This section reviews the basic principle behind using mutation as the primary

variation operator. We start by discussing a simple mutation operator that mutates each variable independently and therefore the non-diagonal elements in the covariance matrix of the mutation distribution are equal to zero. Subsequently, we describe the basic idea behind some more sophisticated approaches that allow adaptation of the covariance matrix.

A simple mutation that mutates each variable independently using a normally distributed random variable contains one parameter per variable. Each parameter specifies the standard deviation of the mutation for the corresponding variable. The standard deviations (mutation strengths) can be either fixed to some small constant or adapted as the search progresses. Ideally, the mutation strength should be proportional to the distance to the optimum. A fixed mutation strength results in slower convergence at either the beginning or the end of the run. Adaptive mutation dynamically updates the mutation strengths to maximize the improvement in the current stage of the algorithm. The 1/5-success rule (Rechenberg, 1973), σ-self-adaptive ES (Schwefel, 1977), and adaptive linear rule (Rechenberg, 1994) are examples of the adaptive mutation strategies.

In σ-self-adaptive ES, a vector of standard deviations corresponding to each variable is attached to each solution. Before mutating the solution, its mutation parameters are modified by using the following rule:

$$\sigma_i' = \sigma_i e^{\tau N(0,1)}, \tag{1}$$

where σ_i is the standard deviation corresponding to the mutation of ith variable, σ_i' is its updated value, $N(0,1)$ is a zero-mean Gaussian random variable with variance 1, and τ is the learning parameter. The above update rule assures that the mutation strength is always positive, the expected outcome of the modification without any selection pressure is neutral, and smaller modifications occur more often than the large ones (Schwefel, 1977). Good mutations are filtered by a standard selection mechanism based on the fitness of the resulting solutions because individuals that lead to the best improvements are going to participate in the reproduction in the subsequent iteration. Schwefel suggests that τ should be inversely proportional to the square root of the total number of variables:

$$\tau \propto \frac{1}{\sqrt{n}}. \tag{2}$$

In the above method, mutations for different variables are independent. This resembles the uniform crossover in genetic algorithms where each bit in the two parent strings is exchanged with a certain probability independently of the remaining bits. To reduce the disruptive effects of recombination, methods that were able to learn the structure of the problem and adapt the recombination accordingly were designed.

A similar approach can be used for adapting mutations where by extracting some information from the history of the run one can learn not only how strong the mutation for each variable should be, but also how the mutations for different variables should interact. Schwefel (1981) proposed to extend solutions by including rotation angles in addition to the original deviations (or variances) and adapt both the variances as well as the rotation angles. The generating set adaptation (GSA) method of Hansen, Ostermeier, and Gawelczyk (1995) is an example of a more advanced method for adapting the direction of the mutation together with its strength by adapting the covariance matrix of the mutation distribution.

Although recombination has been used in evolution strategies since the early works in this area, it has been seen as only a minor operator (Beyer, 1995). For recombination in ES, a variant of uniform crossover is usually used. For each new individual, a subset of parents is chosen. The size of the selected subset can range from two individuals to the entire parent population. For the value of each variable in the new individual, a random individual is picked from the subset and the value is copied from that individual.

When using recombination together with adaptive mutation, one must copy not only the value of each variable but also corresponding mutation strengths or the histories of the past mutations of this variable. Since this information is associated with each variable, no major modifications are required.

ES with adaptive mutation are powerful for local search. However, without powerful recombination ES are capable of only local search. So why not combine recombination-based methods capable of learning how to recombine the solutions properly in a discrete domain and mutation-based methods capable of adapting the mutation in a continuous domain? Section 3 proposes a method that combines the two approaches using discretization to transform continuous solutions into a discrete domain and vice versa. But first, the next section discusses the use of discretization in genetic and evolutionary algorithms.

2.3 DISCRETIZATION

Discretization is widely used in many fields of science to reduce the complexity of a problem and make in-

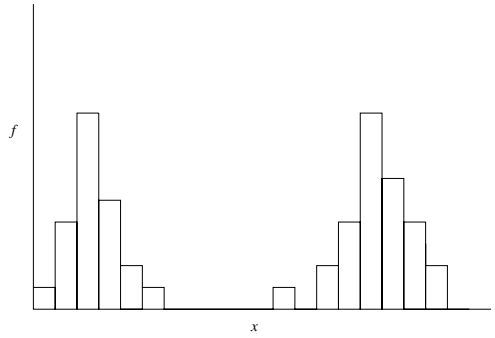


Figure 1: Fixed-width histogram.

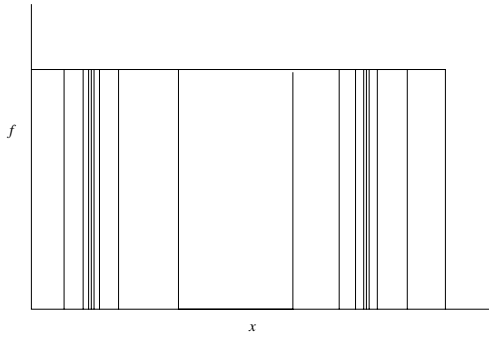


Figure 2: Fixed-height histogram.

tractable problems tractable. In genetic and evolutionary computation, discretization has often been used to first transform continuous solutions into binary strings and then apply the algorithm working in a discrete domain on the resulting problem. The discrete solution can then be transformed back into the continuous domain and either taken as is or further optimized by a local searcher such as the conjugate gradient.

There are several advantages and disadvantages of discretizing the solutions and solving the corresponding discrete problem (Goldberg, 1991). Discrete solutions improve the implicit parallelism of genetic algorithms and allow them to process more partial solutions at the same time. Moreover, the discrete space is finite and thus it is easier to guarantee that the optimal solution in this space is found and that we supply enough information for the optimization to succeed. On the other hand, the locality of the solution decreases and some phenotypically close solutions (similar solutions in a continuous domain) may become distant in the discrete domain. This affects especially mutation that attempts to make small changes in the phenotype by making small changes in the genotype. Additionally, one must know the range of each variable to discretize it and the resulting binary strings may be extremely long for large problems.

A typical way of discretizing continuous solutions in genetic algorithms is to divide the range of each variable into $(2^k - 1)$ intervals of equal width (Goldberg, 1989). Boundary points of the intervals can then be encoded by k-bit binary strings. A string encoding n such continuous variables contains nk bits. Increasing k refines the discretization by factors of 2. For many problems only a couple of bits (say, $k = 3$ or 4) are sufficient to get a solution very close to the optimum. Local optimization methods can then be used to refine the final solutions to get a more accurate result.

A different way of using histograms in evolutionary algorithms for continuous domains is to use histograms as a tool to estimate the distribution of promising points. The created model can then be used to generate new points. The points are allowed to lie within the intervals and not only on their boundaries. That can lead to further improvements of the final solutions.

Algorithms that use histograms in this fashion in order to estimate a univariate distribution where all variables are processed independently have been proposed (Bosman & Thierens, 2000; Tsutsui, Pelikan, & Goldberg, 2001; Cantú-Paz, 2001). Using equal-width histograms was investigated in Bosman and Thierens (2000), Tsutsui et al. (2001), and Cantú-Paz (2001). Equal-height histograms were investigated in Tsutsui et al. (2001), and Cantú-Paz (2001). Decision trees and other supervised discretization methods were investigated in Cantú-Paz (2001).

Various discretization methods were proposed and frequently used in machine learning, statistics, and other fields. Equal-height histograms, decision trees, and clustering algorithms are examples of such methods. All these methods have the same important characteristic—they map a single continuous variable or a group of variables into a finite set of symbols. We discuss some of these methods in the following.

2.3.1 Histograms

Histograms divide the interval for each variable into k subintervals (bins). There are many ways of dividing the interval into k bins. In practice, two basic types of histograms are used: (1) fixed-width histogram and (2) fixed-height histogram.

A fixed-width histogram divides the interval into k bins of equal width. An example of a fixed-width histogram is shown in Figure 1. The disadvantage of fixing the width of each bin is that if points are concentrated in a couple of small regions, only a couple of bins will be nonempty and many bins will simply

be wasted on regions with none or only a few points. Fixed-width histograms are also very sensitive to outliers and one or a few points far away from the rest can significantly decrease the accuracy.

A fixed-height histogram divides the interval for the variable in k bins of equal frequencies (each bin contains the same number of points). An example of a fixed-height histogram is shown in Figure 2. The advantage of using fixed-height histograms is that the density of bins is increased in regions with many points. The regions that seem interesting (those that contain many points) are modeled with high accuracy, while the bins with only few points are merged together to decrease the accuracy where it is not needed. A fixed-height histogram can therefore preserve more information contained in the original set of points.

2.3.2 k-means Clustering

In k-means clustering, each cluster (category) is specified by its *center*. Initially, k centers (where k is given) are generated at random. Each point is assigned to its nearest center. Subsequently, each center is recalculated to be the mean of the points assigned to this center. The points are then reassigned to the nearest center and the process of recalculating the centers and reassigning the points is repeated until no points change their location after updating the centers.

The next section describes how to use a particular discretization or clustering method to combine BOA (or other discrete optimizer) with ES (or other continuous optimizer).

3 COMBINING LINKAGE LEARNING AND ADAPTIVE MUTATION

This section describes an algorithm that combines a discrete recombination-based algorithm (such as BOA) with adaptive mutation techniques of ES.

The algorithm evolves a population of continuous solutions. The first population is generated at random. From the current population the better strings are selected. The processing of the promising solutions has three major phases:

1. Discretize the selected promising solutions.
2. Recombine the discrete solutions.
3. Map the new discrete solutions back in the continuous domain, update the mutation parameters, and mutate the new continuous solutions.

In the first phase, the promising solutions are discretized. Each variable is independently mapped into a finite number of categories (bins, clusters). Any discretization, clustering, or classification method can be used to discretize the continuous variables. Let us denote the resulting number of categories for the ith variable by c_i. There are two major approaches to represent the resulting discrete population. The first approach is to use binary strings and $\lceil \log_2 c_i \rceil$ bits for each variable. The second approach is to use an alphabet of a higher cardinality so that only one symbol is used to represent each variable. The ith letter in the discrete string could then obtain c_i values. Of course, there are many ways between the two extremes. Binary representation results in more possibilities to combine the strings. On the other hand, alphabets of higher cardinality result in shorter representation.

In the second phase, a discrete linkage learning algorithm (such as BOA) is applied to generate the new solutions based on the set of discrete promising solutions. The offspring discrete solutions are constructed by combining the promising solutions.

In the third phase, the resulting set of discrete solutions is mapped back into the continuous domain. However, unlike in all previously proposed approaches, the new points are not generated uniformly within the boundaries of the categories for each variable. Instead, original points within each category are used. Each discrete string determines a category for each variable. To "undiscretize" each variable in a particular string, a random individual in the original set of promising solutions that is consistent with the encoded category for the variable is chosen. The value of the variable is obtained by mutating the value of the variable in the chosen individual.

As a simple example, let us assume we use an equal-width histogram with only two categories for each variable. Each candidate solution is represented in the discrete domain by a binary string with one bit per variable determining whether the variable is in the upper or lower half of its range. To decode a binary string, we look at the value of each of its bits. If the value is 0 (1), we randomly choose a solution from the original set of promising continuous solutions whose value of the considered variable is in the lower (upper) half of the domain. The corresponding variable in the chosen solution is copied to the newly created continuous solution. This is done for each variable separately. Finally, the created continuous solution is mutated.

Using adaptive mutation requires considering additional parameters in the continuous solutions. For σ-self-adaptive mutation, we must attach the mutation

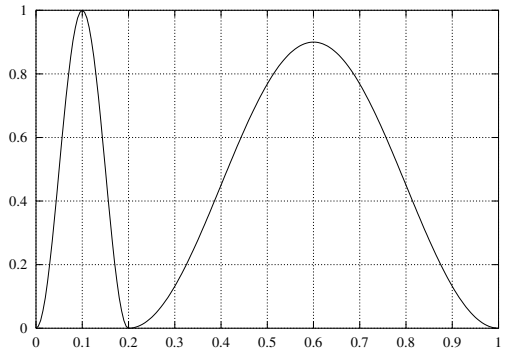


Figure 3: Two-peaks function.

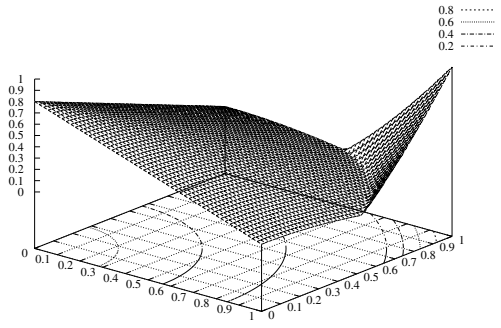


Figure 4: Two-dimensional deceptive function.

strength to each variable in each string. When we copy a value of a particular variable we must also copy the corresponding mutation strength. As we copy the values of the variables and the attached parameters into the new continuous string, the mutation strengths are updated by using the rule discussed earlier in the paper (see Equation 1). The new mutation strengths are used to mutate the created solution. GSA and its successors require copying and updating a history of the mutations or other parameters as well.

The newly generated solutions then replace the original population or its part.

Various algorithms can be used for discretization, linkage learning, and adapting the mutation. Due to our recent successful applications of BOA to many discrete problems, we decided to use this algorithm for linkage learning and recombination in our experiments. To adapt mutation, we used a simple σ-self-adaptive mutation where only a mutation strength of each parameters is adapted. Application of other mutation schemes such as GSA is straightforward. To discretize the solutions, we used equal-height histograms, equal-width histograms, and k-means clustering, but any other popular discretization, classification, and clustering techniques can be used. Using more advanced techniques should further improve the performance. For a discussion of some interesting alternatives, see Cantú-Paz (2001).

4 EXPERIMENTS

This section describes our experiments and presents the experimental results.

4.1 PROBLEMS

We have tested the algorithm on two test functions: (1) two-peak function and (2) deceptive function. Both test functions are created by concatenating basis functions of a small order. The contributions of all the functions are added together to determine the overall fitness and the goal is to maximize the functions. All variables in our test functions are from $[0, 1]$.

The two-peaks function is given by

$$twoPeaks(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} f_{two-peaks}(x_i).$$

Every variable of the two-peaks function contributes to the fitness by

$$f_{two-peaks}(x) = \begin{cases} f_{peak}(x/0.1, 1) & \text{if } x < 0.2, \\ f_{peak}((x-0.2)/0.8, 0.9) & \text{otherwise,} \end{cases}$$

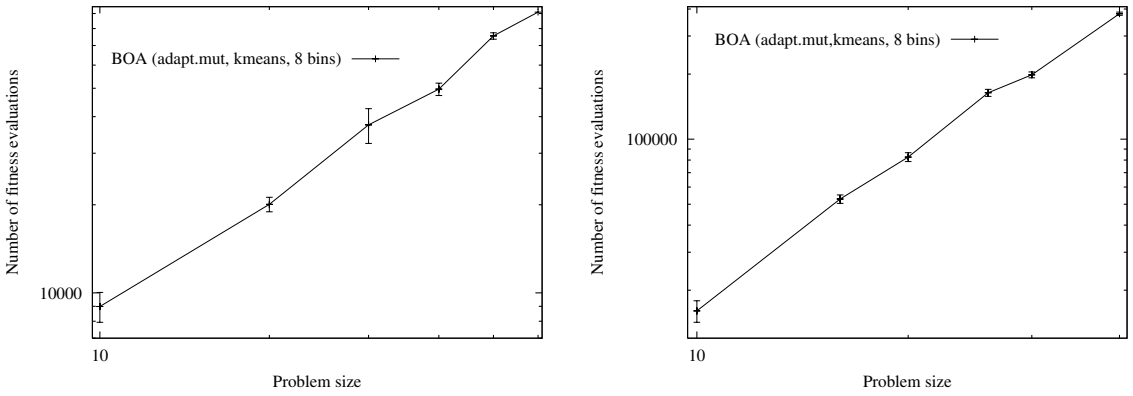
where f_{peak} is a simple function for one peak, defined as

$$f_{peak}(x, h) = h \cos\left(2\pi(x - 0.5)\right).$$

Figure 3 shows the two-peak function. The function has one local and one global optimum for each variable. This yields 2^n optima for a problem of size n out of which only one optimum is global. Local optima are much wider and almost as high as the global one. That makes the problem more difficult. Using mutation only does not yield good results on this problem. Recombination makes uses decomposability of the problem and is capable of solving the problem very efficiently and reliably. Simple uniform crossover is sufficient and thus any ES with recombination should work well.

The deceptive function is composed of two-dimensional deceptive functions:

$$deceptive(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{\frac{n}{2}} f_{two-peaks}(x_{2i}, x_{2i+1}).$$



(a) Two-peaks function.

(b) Deceptive function.

Figure 5: Results of BOA with k-means (8 clusters per variable).

Non-overlapping pairs of variables of the deceptive function contribute to the overall fitness by

$$f_{deceptive}(x,y) = f_{dec}\left(\sqrt{(x^2+y^2)/2}\right)$$

where f_{dec} is a one-dimensional deceptive function defined on $[0,1]$ as

$$f_{dec}(x) = \begin{cases} 0.8 - x & \text{if } x \leq 0.8, \\ \frac{1-x}{0.2} & \text{otherwise.} \end{cases}$$

Figure 4 shows the two-dimensional deceptive function. The two-dimensional deceptive function requires that we learn the linkage of the contributing pairs of variables. Each variable alone is biased to the local optimum in 0 and only when both variables are close to 1 their combination leads to an improvement. In early stages of the run there are more points on the local attractor than the global one. If both variables are treated independently, combinations with both variables near the global attractor vanish and the search progresses toward the local optimum. Moreover, the global optimum is almost isolated and the attractor is small. This makes it quite difficult to hit the global attractor.

4.2 RESULTS

This section presents and discusses empirical results that primarily focus on the scalability. BOA with σ-self-adaptive mutation strength with a learning parameter $\tau = 4/\sqrt{n}$ is used. Due to the limited size of the paper, we only present the results of using k-means clustering for discretization. However, the results of other discretization methods are comparable. In all experiments, a binary tournament selection with replacement is used where to select each new individual, a tournament among two randomly selected individuals is performed and the winner of the tournament is added to the mating pool. An elitist replacement scheme is used that replaces the worst half of the population by the offspring.

For each problem size, we performed 30 independent runs with the optimal population size that was determined empirically for each algorithm and problem size so that the optimum is found in all 30 runs. The average number of fitness evaluations to reach solutions whose Euclidean distance from the optimum is at most 0.01 is provided.

The two-peaks problem is very simple and could be used by using recombination with no linkage learning (i.e. traditional ES recombination based on uniform crossover). However, we present the results to show that the algorithm is capable to solve both simple and difficult problems. Without recombination, the ES with σ-self-adaptive mutation can not solve any of the discussed problems efficiently. The deceptive problem would require exponential population sizes both if no recombination was used as well as if a traditional recombination based on uniform crossover was used. Other fixed recombination methods would also fail if the variables were not ordered according to their dependencies.

Figure 5 shows the results of the proposed algorithm with k-means clustering on the two-peaks and deceptive functions. For the two-peaks function, the population sizes ranged from $N = 700$ for $n = 10$ to $N = 2100$

for $n = 50$, and the required number of evaluations is approximately $O(n^{1.32})$. For the deceptive function, the population sizes ranged from $N = 900$ for $n = 10$ to $N = 8750$ for $n = 40$ and the required number of evaluations grows approximately with $O(n^{2.28})$. Therefore, the performance in both cases can be estimated by a low-order polynomial.

5 CONCLUSIONS

The results of the paper suggest that recombination-based methods for discrete domains and mutation-based methods for continuous domain can be combined to utilize the strengths of both methodologies. The degree to which the methods are combined can be controlled by choosing the resolution of discretization and recombination parameters. For coarse discretization, the algorithm performs very similar to the ES with recombination based on uniform crossover. Refinement of discretization yields to more possibilities for learning the linkage between different variables in the problem. However, learning linkage comes at a price of increased requirements on the population size. While ES usually require only small populations, statistical methods of BOA require quite big populations. But for most multimodal problems it is necessary to combine parts of promising solutions to avoid exponential time requirements.

There are many other alternative uses of the presented scheme. Using supervised discretization methods can yield significant improvements. Additionally, many real-world problems do not require sophisticated linkage learning procedures and simple one, two-point, or uniform crossover may suffice.

6 Acknowledgments

The authors would like to thank Kumara Sastry, Erick Cantú-Paz, Franz Rothlauf, and Hans-Georg Beyer for many useful discussions and valuable comments to the paper.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

Beyer, H.-G. (1995). Toward a theory of evolution strategies: On the benefit of sex – the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation*, *3*(1), 81–111.

Bosman, P. A., & Thierens, D. (2000). Continuous iterated density estimation evolutionary algorithms within the IDEA framework. *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, 197–200.

Cantú-Paz, E. (2001). Supervised and unsupervised discretization methods for evolutionary algorithms. *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 213–216.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, *5*(2), 139–167.

Hansen, N., Ostermeier, A., & Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. *Proceedings of the International Conference on Genetic Algorithms (ICGA-95)*, 57–64.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, 178–187.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks* (IlliGAL Report No. 98013). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, *21*(1), 5–20. Also IlliGAL Report No. 99018.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog Verlag.

Rechenberg, I. (1994). *Evolutionsstrategie '94*. Stuttgart: Frommann-Holzboog Verlag.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, Volume 26 of *Interdisciplinary Systems Research*. Basle, Switzerland: Birkhäuser.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. New York, New York: John Wiley and Sons.

Tsutsui, S., Pelikan, M., & Goldberg, D. E. (2001). *Evolutionary algorithm using marginal histogram models in continuous domain* (IlliGAL Report No. 2001019). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Why use Elitism and Sharing in a Multi-Objective Genetic Algorithm?

Robin C. Purshouse and Peter J. Fleming
Department of Automatic Control and Systems Engineering
University of Sheffield, UK.
{r.purshouse, p.fleming}@sheffield.ac.uk

Abstract

Elitism and sharing are two mechanisms that are believed to improve the performance of a multi-objective evolutionary algorithm (MOEA). Using a new empirical inquiry framework, this paper studies the effect of elitism and sharing design choices using a benchmark suite of two-criterion problems. Performance is assessed, via known metrics, in terms of both closeness to the true Pareto-optimal front and diversity across the front. Randomisation methods are employed to determine significant differences in performance. Informative visualisation of results is achieved using the attainment surface concept. Elitism is found to offer a consistent improvement in terms of both closeness and diversity, thus confirming results from other studies. Sharing can be beneficial, but can also prove surprisingly ineffective. Evidence presented herein suggests that parameter-less schemes are more robust than their parameter-based equivalents (including those with automatic tuning). A multi-objective genetic algorithm (MOGA) combining both elitism and parameter-less sharing is shown to offer high performance across the test suite.

1 INTRODUCTION

Evolutionary multi-criterion optimisation (EMO) practitioners are faced with a number of design choices beyond those encountered in a standard evolutionary algorithm (EA). Suitable strategies for elitism and sharing can significantly improve optimiser performance. This paper presents new evidence and understanding concerning elitism and sharing that will help practitioners to make informed choices. Through the application of tractable algorithm modifications and a rigorous experimental framework, the effect of MOEA component-level choices can be more clearly exposed.

An *EMO empirical inquiry framework* is introduced in Section 2. The dual performance metrics of closeness and diversity are measured using the generational distance and spread metrics respectively. Statistical comparisons are then made using randomisation testing. Information-rich visualisations of the identified trade-off surfaces are obtained using attainment surfaces. The analysis is based on the two-criterion set of test problems proposed by Zitzler *et al* [2000].

The performance of a baseline MOGA optimiser is established in Section 3. The effects of elitism and sharing are then considered with reference to this baseline. An elitist strategy, based on Zitzler's [1999] *universal elitism*, is developed in Section 4. Sharing methodologies for the promotion of diversity are discussed in Section 5. A new parameter-less technique, formulated as an accompaniment to Pareto-based ranking, is compared to the standard parameter-based approach. In Section 6, a high-performance MOGA incorporating both elitism and parameter-less sharing is investigated.

2 EMO INQUIRY FRAMEWORK

2.1 TEST SUITE

The established set of test problems developed by Zitzler *et al* [2000] (ZDT) is used in this study. The suite consists of six, tractable, two-criterion functions, with varying characteristics as summarised in Table 1.

Table 1: Test function characteristics

NAME	ATTRIBUTES
ZDT-1	Convex front
ZDT-2	Non-convex front
ZDT-3	Non-contiguous convex front
ZDT-4	Many local fronts, single global convex front
ZDT-5	Deceptive problem, convex front
ZDT-6	Non-uniform distribution, non-convex front

2.2 MEASURING PERFORMANCE

Performance of a MOEA can be decomposed into two criteria:

- **Closeness** – the nearness of the obtained non-dominated solutions to the true front.
- **Diversity** – the coverage of the trade-off surface by the obtained solutions.

The ideal outcome, in test cases of this type, is a final population with a uniform distribution of globally non-dominated solutions spread across the entire trade-off surface. Various performance metrics have been proposed to measure closeness, diversity, and in some cases both together. Some metrics require that the global trade-off surface is known and can be sampled (straightforward in the ZDT cases), whilst others involve a purely relative

comparison of two results sets. This study utilises three known performance metrics: *generational distance* to measure accuracy, *spread* to measure diversity, and *attainment surfaces* to provide visualisation.

- **Generational distance** – an average of the Euclidean distances between each obtained solution and the nearest point on the true front [Veldhuizen, 1999].
- **Spread** – the sum of the differences between nearest neighbour distances and the mean of all such distances, coupled with a term to account for the extent of the obtained front [Deb *et al*, 2000].
- **Attainment surface** – the boundary in criterion-space that separates the region that is dominated by the obtained solutions from that which is non-dominated [Fonseca and Fleming, 1996].

The superposition of multiple attainment surfaces can be treated statistically and also provides a rich qualitative indication of performance. A typical plot is shown later in Figure 1. The heavy line indicates the 50%-attainment surface (akin to the median), the thinner lines show the 25% and 75% surfaces (quartiles), and the dotted lines describe the 0% and 100% surfaces. Thus location, dispersion, and skewness information can be obtained in a similar manner to the box plot [Cleveland, 1993].

2.3 ANALYSING PERFORMANCE

Upon completion of a single run of a specific MOEA configuration on a particular problem, three sets of non-dominated criterion vectors (and associated solutions) are obtained, namely:

- **final population** – the non-dominated vectors in the final population of the algorithm,
- **on-line archive** – the final elite set of vectors, and
- **off-line archive** – the complete set of non-dominated vectors identified by the algorithm.

The first of these sets is used for analysis and comparison purposes in this study since it provides the most appropriate measure of the on-line trade-off surface *maintenance* capabilities of an algorithm.

An evolutionary algorithm is a stochastic process and, thus, multiple runs (samples) are required in order to infer reliable conclusions as to its performance. Hence, 35 runs have been conducted for each MOEA configuration when applied to a particular test problem. The performance of the algorithm is expressed in the resulting distributions of generational distance and spread. A statistical comparison of two configurations is then possible through the use of a test statistic.

In this study, the mean difference between two generational distance (or, alternatively, spread) distributions is taken as the test statistic. The significance of this observed result is then assessed using *randomisation testing*. This is a simple, yet effective, technique that does not rely on any assumptions concerning the attributes of the underlying processes, unlike conventional statistical methods [Manly, 1991]. The central premise of the method is that, if the observed result has arisen by chance, then this value will not appear unusual in a distribution of results obtained through many random relabellings of the samples. The randomisation method proceeds as follows:

1. Compute the difference between the means of the samples for each algorithm: this is the observed difference.
2. Randomly reallocate half of all samples to one algorithm and half to the other. Compute the difference between the means as before.
3. Repeat Step 2 until 5000 randomised differences have been generated, and construct a distribution of these values.
4. If the observed value is within the central 99% of the distribution, then accept the null hypothesis. Otherwise consider the alternative hypotheses. This is a two-tailed test at the 1%-level.

The null hypothesis is that the observed value has arisen through chance and so there is no performance difference between the two configurations. The alternative hypotheses are that the difference is unlikely to have arisen through chance and that one configuration has outperformed the other (depending on which side of the distribution the observed difference falls, and the direction in which the difference has been calculated).

Note that the observed value is included as one of the random relabellings since, if the null hypothesis is true, then this value is one of the possible randomisation results. 5000 randomisations is regarded as an acceptable quantity for a test at the 1%-level [Manly, 1991].

The results of randomisation testing are simple to visualise, as shown by the example in Figure 3. The randomised results are described by the grey histogram, whilst the observed result is depicted as a filled black circle. Each row shows the performance on a particular test function (from ZDT-1 at the top, to ZDT-6 at the bottom). The left-hand column indicates the relative performance regarding closeness, and the right-hand column shows the corresponding difference in diversity.

3 BASELINE MOGA

3.1 DESCRIPTION

The baseline optimiser used in this study has been developed according to the holistic design principles championed by Michalewicz and Fogel [2000] and has previously been shown to be effective at solving the ZDT test problems [Purshouse and Fleming, 2001]. A summary of the algorithm is provided in Table 2.

The multi-criterion performance of a solution is scalarised using Fonseca and Fleming's [1993] Pareto-based ranking procedure. A solution is ranked according to the number of solutions in the population that are *preferred* to it. If the entire Pareto-optimal front is to be identified, the preference relation collapses to a test for Pareto dominance.

Table 2: Baseline configuration

EMO COMPONENT	STRATEGY
GENERAL	
Population size	100
Total generations	250
ELITISM	None
EVALUATION	[1] Fonseca and Fleming [1993] Pareto-based ranking. [2] Linear fitness assignment with rank-wise averaging. [3] No modification of fitness to account for population density.
SELECTION	Stochastic universal sampling
REPRESENTATION	
Real parameter functions	Concatenation of real number decision variables. Accuracy bounded by machine precision.
Binary function	Binary string, 80 bits in length.
OPERATORS	
For real representations	[1] Naïve crossover Probability = 0.8. [2] Gaussian mutation (initial search power of 40% of variable range; sigmoidal scaling set to 15; feasibility requirement of one standard deviation). Probability = Expected value of 1 phenotype per chromosome.
For binary representations	[1] Single-point binary crossover. Probability = 0.8. [2] Simple bit-flipping mutation. Probability = 1/80.

When ranking is complete, initial fitness values can be prescribed. The population is sorted according to rank and fitnesses are assigned by interpolating between the highest fitness value for the best rank and the lowest fitness value for the worst rank. In the baseline algorithm, linear interpolation is used and fitness is varied between the population size (highest) and unity (lowest). The ratio of these two fitnesses is a definition of the *selective pressure* of the assignment mechanism. Solutions of the same rank then have their fitnesses adjusted to the average of the original assignments for that rank.

Part of this study is concerned with the effect of diversity-preserving mechanisms. Therefore no manipulation of the above fitnesses through sharing is undertaken.

Stochastic universal sampling has been chosen as the selection mechanism [Baker, 1987]. This method achieves maximum spread with minimal bias, but is non-parallelisable. In total, 100 selections are required since the chosen reinsertion strategy is that all offspring replace all parents (no generational gap) and since for the chosen genetic operators two parents are required to produce two offspring.

Since five of the test problems feature real number decision variables, it is logical to use a real number representation for these problems. Hence, a candidate solution is described by a concatenation of phenotypic decision variables. The other test problem, ZDT-5, explicitly uses binary variables, thus a binary representation is natural for this problem.

Different representations require different search operators. For the binary chromosome case, the familiar single-point two-parent crossover and bit-flipping mutation operators are employed. Good results are known to be achievable using this simple approach [Zitzler *et al*, 2000]. For real representations, the so-called *naïve crossover* is used in conjunction with a *Gaussian mutation* operator. The former of these search tools is a very simple two-parent single-point crossover operator, where the crossover sites are limited to points between decision variables. This offers quite a low-power search, since it cannot generate any values for decision variables that were not present in the original population. However, when coupled with a complementary high-power search tool, the resulting search capabilities are considerable[1]. Gaussian mutation is one such operator. Its main benefit is that it provides tuneable search power in the form of the standard deviation. This can be exploited to provide on-line adaptation that avoids the generation of infeasible solutions and controls convergence speed by varying the search from near global early on to very local towards the end. *Sigmoidal* variation, as a function of the percentage of generations completed, of the standard deviation is useful because it allows concentrated periods of high- and low-power search [Purshouse and Fleming, 2001].

3.2 PERFORMANCE

Attainment surfaces illustrating the performance of the baseline algorithm are shown in Figure 1. Particularly good results were achieved for ZDT-1, ZDT-2, and ZDT-3 (Figures 1a, 1b, and 1c respectively) in terms of both closeness to the global Pareto front and diversity across the front. The tight envelopes of attainment indicate the high level of consistency achieved in these cases. The MOGA struggled to achieve good coverage of the surface as f_1 approaches zero on ZDT-2. Note that this is a region where there is little trade-off between the objectives.

As shown in Figure 1d, the wider envelopes of attainment produced for the multi-fronted ZDT-4 signify entrapment in a locally non-dominated front. On no occasions did the MOGA converge to the global trade-off surface although coverage across the identified fronts was good.

The baseline MOGA achieved reasonable closeness to the global front on ZDT-5. Performance on this deceptive test function is depicted in Figure 1e. Note that on no occasions was the algorithm able to identify the extreme right-hand section of the discrete trade-off surface.

Rather poor performance was observed on the non-uniform ZDT-6, as shown in Figure 1f. Coverage was especially poor on the less dense area of the front. This, together with the missing section of the ZDT-5 front, is

[1] Coincidentally, the incorporation of naïve crossover largely prevents the convergence failures encountered by Ikeda *et al* [2001], thus showing that MOEA failure cannot be solely blamed on the use of Pareto ranking in these cases.

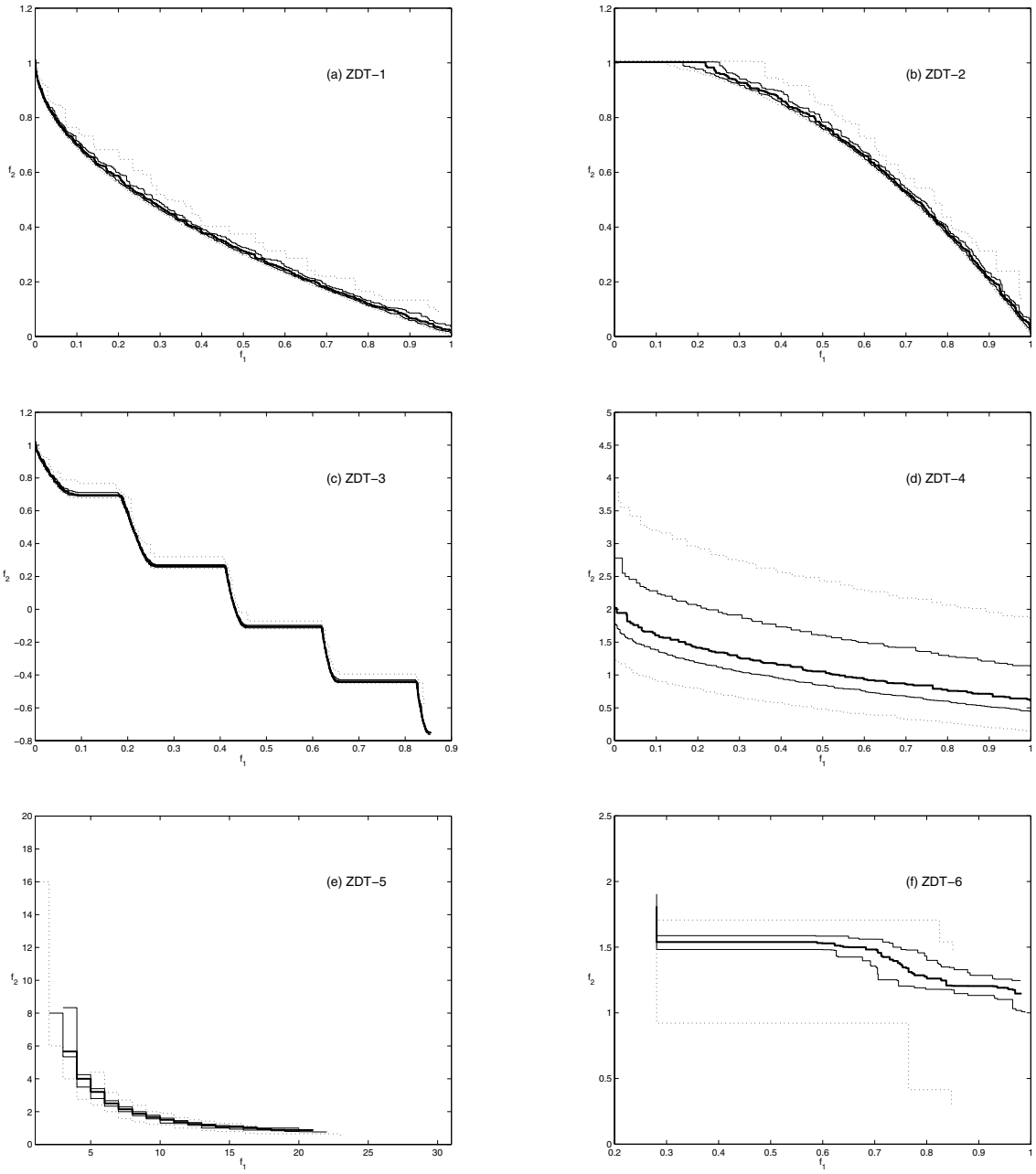


Figure 1: Attainment surfaces – baseline MOGA solving the ZDT problems

the strongest indication that density-based sharing would be beneficial. Closeness to the true Pareto front is also not good. Only the 0%-attainment surface lies on the global front, where coverage is particularly poor. Furthermore, the position of this front with respect to the median and quartiles suggests that this result is something of an outlier.

4 ELITIST STRATEGY

Elitism is the process of preserving previous high-performance solutions from one generation to the next. This is conventionally achieved by simply copying the solutions directly into the new generation. Elitism has long been considered an effective method for improving the efficiency of an EA [De Jong, 1975]. Various recent studies in the EMO community have indicated that the inclusion of an elitist element can considerably improve the performance of an MOEA [Zitzler *et al*, 2000; Deb *et al*, 2000]. The two main issues are (1) how to manage the size of the elite sub-population, and (2) how to use elitism to drive the search effectively.

The elitist strategy adopted in this study is a variant on the approach developed by Zitzler [1999] and is illustrated by the schematic in Figure 2. The key difference is that the archive size is allowed to vary within pre-defined limits, whilst the number of newly generated candidate solutions

is varied such that the total population size (elites plus new solutions) is held constant.

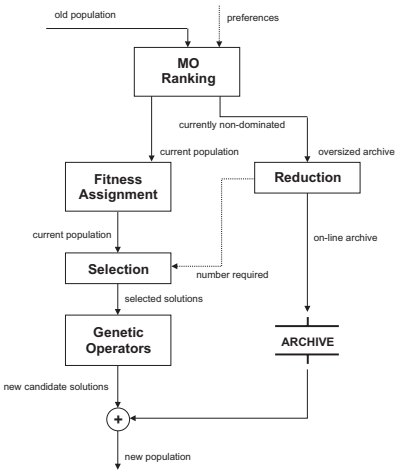


Figure 2: Elitist strategy

The on-line archive is initialised to the empty set, whilst the initial population is initialised to a random set of candidate solutions (possibly seeded with information provided by the decision-maker). The populations at subsequent iterations of the algorithm are the combination of new solutions and current elite solutions. The currently non-dominated solutions in the population are identified and are stored as the new, potentially over-sized, archive. Over-represented solutions are then eliminated from the archive, if necessary, using the *SPEA-2* truncation procedure [Zitzler *et al*, 2001]. This is an effective reduction technique for two-criterion problems.

When the new elite set has been finalised, the size of this set is known, and thus the number of new candidate solutions required to fill the population can be calculated. These solutions are created through the selection and genetic manipulation of members of the current population. The new solutions are then combined with the elite set to form the total population, which completely replaces the old population.

This elitist strategy has been integrated within the baseline MOGA and has been applied to the test problems. Randomisation test results between the elitist model and the baseline are shown in Figure 3. Observed differences to the left of the randomisation distribution offer evidence in favour of the elitist version outperforming the baseline case.

There is considerable evidence, clearly shown by the results in Figure 3, that the elitist algorithm produces results closer to the true front than the baseline for ZDT-1, 2, 3, 4, and 6. Superior performance in terms of diversity is strongly suggested for ZDT-1, 2, 4, 5, and 6.

Elitism increases the convergence speed of the algorithm. The danger of sub-optimal convergence is somewhat reconciled by the distributed nature of the elite set. High-power search operators, such as the Gaussian mutation operator used in this work, can also reduce the risk of premature convergence. Hence, the increased convergence exhibited in this study is expected.

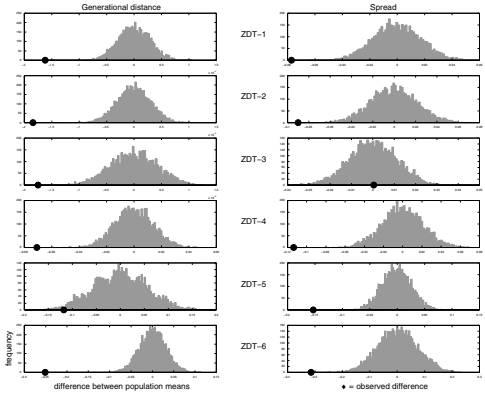


Figure 3: Elite versus baseline

The elitist scheme also maintains the characteristics of the currently identified trade-off surface within the on-line population. Thus, diversity of non-dominated solutions in the population is maintained and encouraged (through the thinning of similar criterion vectors) by the truncation mechanism. This helps to explain the improvement in diversity seen in the results. However, the truncation process only represents the current distribution: it does not, directly (though fitness), drive the search towards a superior distribution. Despite this fact, the inclusion of elitism did lead to improved diversity on the non-uniformly distributed ZDT-6. Modifications to the fitness, such as those arising through sharing, may assist further in improving diversity across the trade-off surface.

5 SHARING STRATEGY

5.1 INTRODUCTION

One of the aims of a multi-objective evolutionary algorithm is to obtain a suitable *distribution* of candidate solutions in regions of interest to the decision-maker. In an evolutionary algorithm, this can be achieved through the formation of sub-population clusters – known as *niches* – within the global population. *Fitness sharing* is the most popular method for fostering this niching process [Goldberg and Richardson, 1987]. In this approach, the raw fitness value of a candidate solution is reduced by a factor dependent on the local population density. This measure should be made in the domain over which a good distribution is of interest: usually criterion-space.

5.2 PARAMETER-BASED METHODS

Fitness sharing has been shown to combat the problem of *genetic drift* (population convergence to a single point due to stochastic selection errors), thus helping to attenuate the possibility of sub-optimal convergence and to enhance coverage of trade-off surfaces. However, the power law equation on which the technique is based requires a definition of *closeness* in order to calculate the population densities. This can be difficult to estimate in practice. Furthermore, the method is sensitive to choice of this

niche size parameter. Several methods have been proposed in order to estimate the niche size, for example Deb and Goldberg [1989] and Fonseca and Fleming [1993], of which the dynamic approach of Fonseca and Fleming [1995] is particularly interesting.

Fonseca and Fleming [1995] noted the similarity between the power law sharing function and the *Epanechnikov* kernel density estimator used by statisticians. The kernel smoothing parameter used in the estimator was found to be directly analogous to the fitness sharing niche size parameter. The key benefit of this is that statisticians have developed successful techniques for estimating the value of this parameter [Silverman, 1986]. Furthermore, the approach is amenable to update at each generation of the EA population. This approach can be regarded as parameter-based sharing with automatic tuning.

Epanechnikov sharing has been added to the baseline MOGA and has been applied to the benchmark problems. Sharing is performed using the Euclidean distance metric in the criterion domain. Results of a randomisation comparison with the baseline algorithm are shown in Figure 4. Observed values that favour the sharing scheme will lie to the left of the randomisation distribution.

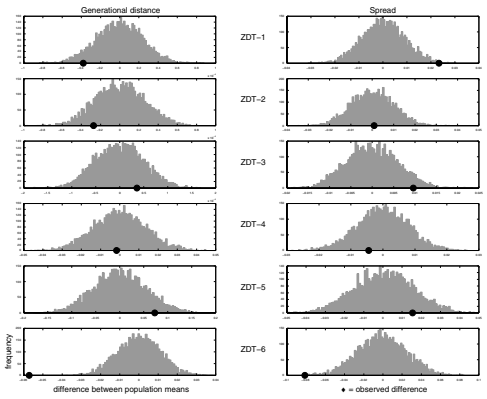


Figure 4: Epanechnikov versus baseline

The inclusion of Epanechnikov sharing has improved both aspects of performance on the non-uniform ZDT-6. Note in particular that a method designed to improve diversity has also helped to improve convergence, thus suggesting the strong interaction between the two performance criteria. However, no improvements in either diversity or closeness have been achieved for any other test function. Indeed there is some evidence to suggest deterioration in diversity on ZDT-1. The lack of improvement to diversity is of particular concern, since the elitist results in Section 4 have indicated that diversity *can* be greatly improved on these problems. A possible explanation for the lack of success is that the automatic parameter selection is providing poor estimates.

5.3 PARAMETER-LESS METHODS

The difficulty and inconvenience involved in determining the niche size value has led many researchers to investigate parameter-less methods for achieving niching. A new approach is presented here that increases the resolution of the Fonseca and Fleming [1993] Pareto-based ranking procedure through the inclusion of population density information. An *intra-ranking* is performed on candidate solutions of identical Pareto-based rank, discriminating on the basis of population density at that rank. Solutions in less dense areas receive a superior intra-ranking to their counterparts in denser regions. This approach requires a definition of *distance* (Euclidean nearest neighbour is used herein) but does not require a definition of closeness. In practice, the distance metric is likely to be problem dependent and could conceivably include decision-maker preference information. Following the new fine-grained ranking process, the fitness assignment procedure remains unchanged.

Using this scheme, if one candidate solution is preferred to (dominates) another, then the former is guaranteed to have a superior fitness value. Also, when all solutions are non-dominated, discrimination is based purely on density. If, in addition, the density is globally uniform then all fitnesses are identical.

With any type of ranking scheme, information content is lost. Ranking indicates that one solution lies in a more densely packed region than another solution but the actual difference in density between the two is lost. This limits the amount of information available to the search procedure but protects against premature convergence to locally *superfit* solutions and removes the requirement for a niche size setting.

The results for this new sharing scheme, compared to the non-sharing baseline model, are shown in Figure 5. The central aim of sharing is to improve the distribution of solutions in criterion-space and this should be primarily evident in the spread results. There is strong evidence to suggest that the new method improved spread on ZDT-3 and ZDT-4. The use of the Epanechnikov kernel, by contrast, did not improve results on these problems. In no case was the absence of a sharing mechanism shown to be preferable. However, there is little evidence to suggest that the use of sharing made any difference to the results for ZDT-6. This is particularly disappointing since this problem has a non-uniform distribution across its trade-off surface: a situation in which sharing is considered a highly appropriate strategy.

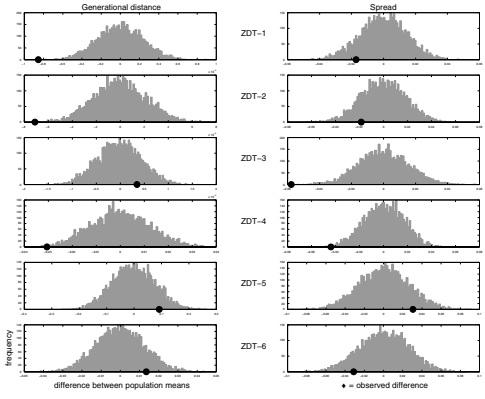


Figure 5: New sharing versus baseline

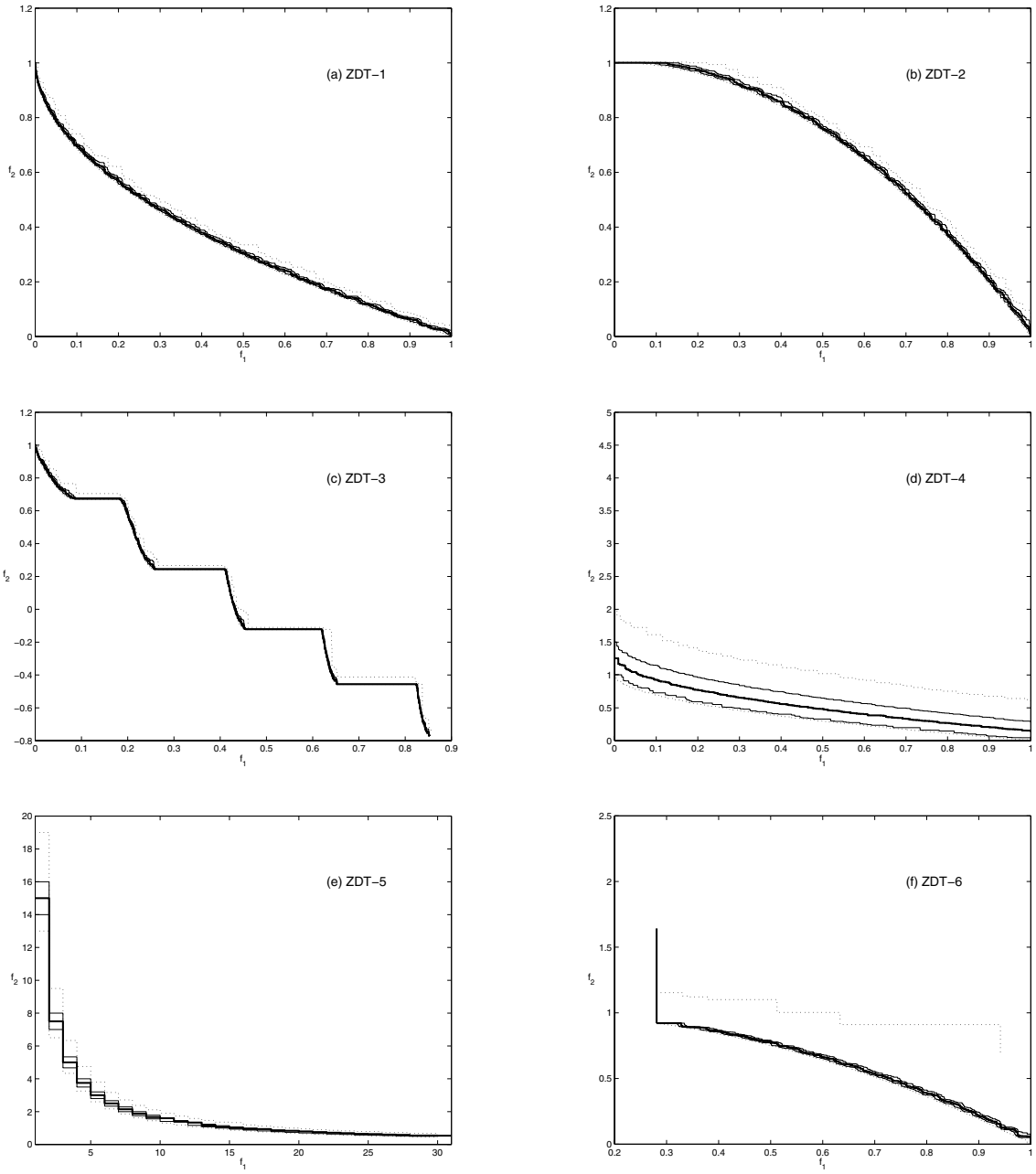


Figure 6: Attainment surfaces – elitist, parameter-less sharing MOGA solving the ZDT problems

6 HIGH-PERFORMANCE MOGA

The use of an elitist strategy or a parameter-less sharing strategy in isolation has been shown to offer improved performance. It is instructive to also consider the effect of these schemes in combination. Attainment surfaces for such an algorithm are shown in Figure 6. The envelopes of attainment are generally very tight, indicating good consistency. As evident from Figure 6d, closeness has been greatly improved on ZDT-4: indeed the 25%-attainment surface lies very close to the global front of this difficult test problem. Complete coverage of the right-hand portion of the trade-off surface has been achieved for ZDT-5, as shown in Figure 6e. Finally, closeness and diversity have been much improved on ZDT-6 (Figure 6f).

Comparisons with the baseline MOGA are made using randomisation testing in Figure 7. Observed differences that lie to the left of the randomisation distribution favour the new algorithm. Compelling evidence points to the algorithm substantially outperforming the baseline in terms of diversity across all six benchmark problems. The *combination* of elitism and new sharing was required in order to achieve this notable result: neither elitism nor sharing alone was shown to be sufficient. Improved closeness was observed for ZDT-1, 2, 4, and 6 (the result for ZDT-5 is not significant at the 1%-level).

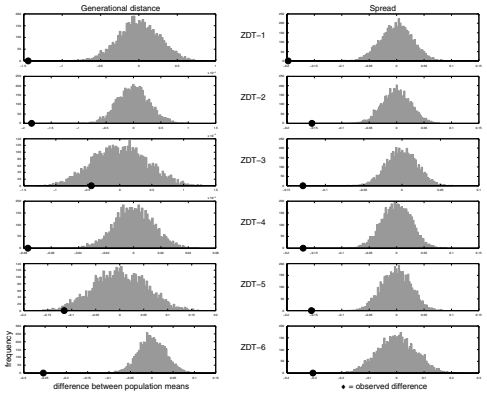


Figure 7: Elitist, sharing MOGA versus baseline

7 CONCLUSION

Using a progressive and tractable experimental approach, supported by appropriate statistical and visual analyses, this paper has shown that elitist and sharing strategies can significantly improve the performance of an evolutionary multi-criterion optimiser. Existing elitist heuristics are again shown to be beneficial, this time using a new analysis technique and in the context of MOGA. However, the shortcomings of a popular parameter-based sharing technique have been exposed, as have the dangers of relying too heavily on an automatic parameter-setting method. A new parameter-less method of sharing has been introduced and has been shown to be more reliable than the standard method. Impressive results were achieved when both elitism and sharing were used together. As a final word of caution, these results have been obtained for two-criterion problems: further research is required to ascertain the effectiveness of these methods as the dimension of the problem increases.

The results described in this paper, together with an extended research report, are available for download from the following site:

`http://www.shef.ac.uk/~acse/research/students/r.c.purshouse/`

Acknowledgments

The first author gratefully acknowledges support from UK EPSRC. The authors would also like to thank the anonymous reviewers for their helpful comments and suggestions, and would like to express special thanks to Dr Nick Fieller from the Department of Probability and Statistics at the University of Sheffield for his valuable advice.

References

Baker, J. E., 1987, *Reducing bias and inefficiency in the selection algorithm*, Proceedings of the 2nd International Conference on Genetic Algorithms, 14-21.

Cleveland, W. S., 1993, *Visualizing Data*, Hobart Press, Summit, NJ.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T., 2000, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II,* KanGAL report 200001.

Deb, K. and Goldberg, D. E., 1989, *An Investigation of Niche and Species Formation in Genetic Function Optimization*, Proceedings of the 3rd International Conference on Genetic Algorithms, 42-50.

De Jong, K. A., 1975, *An analysis of the behavior of a class of genetic adaptive systems*, PhD thesis.

Fonseca, C. M. and Fleming, P. J., 1993, *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*, Proceedings of the 5th International Conference on Genetic Algorithms, 416-423.

Fonseca, C. M. and Fleming, P. J., 1995, *Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction*, Proceedings of GALESIA '95, 45-52.

Fonseca, C. M. and Fleming, P. J., 1996, *On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers*, Parallel Problem Solving from Nature – PPSN IV, Lecture Notes in Computer Science, **1141**:584-593.

Goldberg, D. E. and Richardson, J., 1987, *Genetic Algorithms with Sharing for Multimodal Function Optimization*, Proceedings of the 2nd International Conference on Genetic Algorithms, 41-49.

Ikeda, K., Kita, H., and Kobayashi, S., 2001, *Failure of Pareto-based MOEAs: Does Non-dominated Really Mean Near to Optimal?*, CEC 2001, **2**:957-962.

Manly, B. F. J., 1991, *Randomization and Monte Carlo Methods in Biology*, Chapman and Hall, London.

Michalewicz, Z. and Fogel, D. B., 2000, *How to Solve It: Modern Heuristics*, Springer-Verlag, Berlin.

Purshouse, R. C. and Fleming, P. J., 2001, *The MultiObjective Genetic Algorithm Applied To Benchmark problems – An Analysis*, ACSE Research Report 796, `http://www.shef.ac.uk/~acse/research/students/r.c.purshouse/reports/report796.pdf`.

Silverman, B. W., 1986, *Density Estimation for Statistics and Data Analysis*, Monographs on Applied Statistics and Probability **26**, Chapman and Hall.

Veldhuizen, D. A. V., 1999, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, PhD thesis.

Zitzler, E., 1999, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, PhD thesis.

Zitzler, E., Deb, K., and Thiele, L., 2000, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation **8**(2):173-195.

Zitzler, E., Laumanns, M., and Thiele, L., 2001, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, TIK-Report 103, ETH Zürich.

Genetic Algorithms, Efficiency Enhancement, and Deciding Well with Differing Fitness Variances

Kumara Sastry and **David E. Goldberg**
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
{ksastry,deg}@uiuc.edu

Abstract

This study investigates the decision making between fitness function with differing variance and computational-cost values. The objective of this decision making is to provide evaluation relaxation and thus enhance the efficiency of the genetic search. A decision-making strategy has been developed to maximize speed-up using facetwise models for the convergence time and population sizing. Results indicate that using this decision making, significant speed-up can be obtained.

1 Introduction

Significant progress has been made both in analysis and design of genetic algorithms (GAs) over the last decade. Design procedures for the development of *competent* GAs have been proposed and much progress has been made along these lines (Goldberg, 1999). A GA is called *competent* if it can solve hard problems quickly, accurately, and reliably. In essence, competent GAs take problems that were intractable with the first generation GAs and render them tractable. Competent GAs successfully solve problems with *bounded difficulty* oftentimes requiring only a subquadratic (polynomial) number of function evaluations.

However, for large-scale problems, the task of computing even a subquadratic number of function evaluations can be daunting. This is especially the case if the fitness evaluation is a complex simulation, model, or computation. This places a premium on a variety of *efficiency-enhancement techniques*. Therefore GA practitioners resort to approximate fitness functions that are less expensive to compute. Such approximations introduce error in assessing the solution quality.

Usually, one has to choose among a set of fitness functions with varying degrees of error. The choice of a fitness function has a large impact on the computational resources and the solution quality. Oftentimes, practitioners choose a fitness function on an ad hoc basis which might not necessarily be the correct choice. Therefore, there is a need to investigate which fitness function should be used and under what scenarios.

However, error comes in two flavors: bias and variance. Variance and bias affect the search process in different ways and therefore have to be handled in different manner (Keijzer & Babovic, 2000). This paper considers the decision making under the presence of variance alone and decision making in the presence of bias is presented elsewhere (Sastry, 2001). This separation will not only ease the analytical burden, but also highlight the difference in the decision-making procedure.

This paper investigates the decision-making process between two fitness functions with differing variance values and computational costs. Although the fitness function with low variance requires a smaller population size and converges faster, the overall computational cost can be higher due to its higher cost. On the other hand, the low-cost fitness function is cheaper to compute, but both the population size and the convergence time increase, which in turn increases the total computational cost. Therefore, one has to choose one of the two fitness functions. The objective of this study is to develop a decision-making strategy that yields maximum speed-up. Facetwise models for convergence time and population sizing are used to predict speed-up and these models are verified with empirical results along the way.

This paper is organized as follows. Section 2 briefly discusses the past work on handling error in fitness functions. The problem addressed in this paper is defined in section 3. Then, facetwise models for convergence time, population size and total number of

function evaluations are developed in the subsequent section. The strategy that yields maximum speed-up is discussed in section 5. Finally, a summary and key conclusions of this study is presented.

2 Literature Review

Efficiency-enhancement techniques are essential for solving large-scale, complex search problems. One such technique is *evaluation relaxation*. Evaluation-relaxation schemes try to reduce the computation burden by utilizing inexpensive, but error-prone fitness assignment procedures instead of an expensive, but accurate fitness function.

Grefenstette and Fitzpatrick (1985) studied the utility of approximate evaluations in an image registration problem and obtained significant speed-up by random pixel sampling instead of complete sampling. Follow-up studies (Fitzpatrick & Grefenstette, 1988; Mandava, Fitzpatrick, & Pickens, 1989) have provided further evidence of efficiency-enhancement by using approximate fitness evaluations. Early studies of approximate function evaluations were largely empirical, and a design methodology for predicting the behavior of GAs was lacking. Miller and Goldberg (1995) provided a theoretical framework for handling noisy function evaluations. Specifically they developed convergence-time models in the presence of external noise. Miller and Goldberg (1996) extended the convergence time model for different selection methods. Miller (1997) proposed a detailed design methodology including development of population-sizing model and optimal sampling prediction for noisy environments.

Other studies exist on utilizing approximate fitness functions to speed-up the genetic search (Ratle, 1998; El-Beltagy, Nair, & Keane, 1999; Jin, Olhofer, & Sendhoff, 2000; Albert, 2001). However, an exhaustive survey is beyond the scope of this study.

3 Problem Definition

Consider two noisy fitness functions f_1 and f_2 for a search problem. Functions f_1 and f_2 consist of zero-mean Gaussian noise of variance $\sigma^2_{N_1}$ and $\sigma^2_{N_2}$ respectively. The cost of a single evaluation of f_1 is c_1 and that of f_2 is c_2. Also, $\sigma^2_{N_1} < \sigma^2_{N_2}$, and $c_1 > c_2$. That is, f_1 is a high-cost, low-variance function, and f_2 is a low-cost, high-variance fitness function. The objective is to correctly decide which fitness function to employ so as to obtain highest speed-up. As will be seen later, this decision has to be made spatially. To achieve this goal, we first have to develop appropriate models for the convergence time and the population size required.

4 Facetwise Models

In this section, we will develop a facetwise model for convergence time of GAs in presence of external noise. Then an existing model for population sizing is presented and these models are used to compute an expression for the total number of function evaluations. Finally, these facetwise models are verified with empirical results.

4.1 Convergence Time

Understanding run duration is one of the critical factors for analyzing GAs. Elsewhere, a motivation and the utility of understanding time has been discussed Goldberg (in press). Three main approaches have been used in understanding time: (1) Modeling of takeover time, where the dynamics of the best individual is modeled (Goldberg & Deb, 1991), (2) Selection-intensity model, where the dynamics of the average fitness of the population is modeled (Mühlenbein & Schlierkamp-Voosen, 1993; Bäck, 1995; Miller & Goldberg, 1995; Miller & Goldberg, 1996), and (3) Higher-order cumulant model, where the dynamics of average and higher-order cumulants are modeled (Blickle & Thiele, 1995; Prügel-Bennet & Shapiro, 1994).

Even though higher-order cumulant models are more accurate than selection-intensity models, they do not provide a closed-form solution for either the proportion of correct building blocks or the convergence time. Therefore, in this study we develop a selection-intensity based convergence-time model for the One-Max domain. The OneMax problem has two key properties: (1) Uniform building-block salience, and (2) Gaussian fitness distribution. Uniform building-block salience implies that the contribution of building blocks in different partition to the fitness is equal. The assumption of Gaussian fitness distribution is approximately true as recombination and other genetic operators have a normalizing effect.

Therefore the fitness distribution $F = \mathcal{N}(\mu_t, \sigma_t^2)$, and $N = \mathcal{N}(0, \sigma_N^2)$. Here, μ_t is the mean true fitness at time t. Furthermore, the noisy fitness distribution, F' can be written as $F' = F + N$, where, F is the actual fitness distribution, and N is the external noise (in this case, zero-mean Gaussian noise). Since both the actual fitness and the noise are normally distributed, the noisy fitness function is also normally distributed:

$$F' \sim \mathcal{N}(\mu_t, \sigma_t^2 + \sigma_N^2). \tag{1}$$

Under these assumptions, the expected average fitness of the population after selection, given the current av-

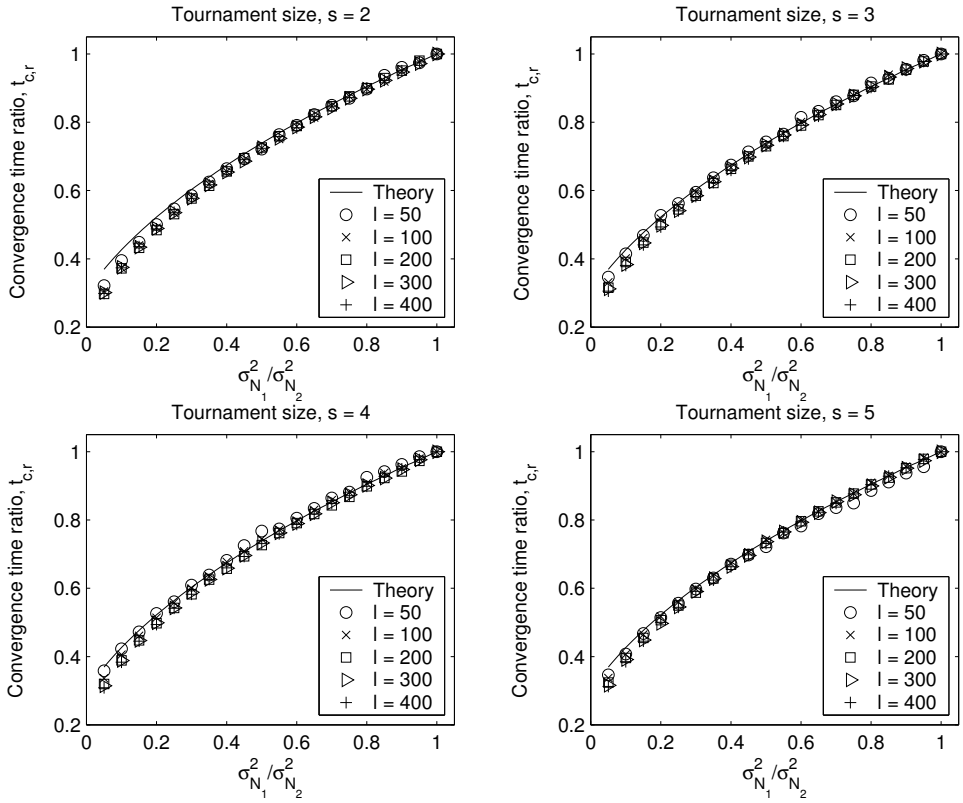


Figure 1: Empirical verification of the convergence-time-ratio model (equation 8).

erage fitness is given by (Miller & Goldberg, 1995):

$$\mu_{t+1} - \mu_t = \frac{I\sigma_t^2}{\sqrt{\sigma_t^2 + \sigma_N^2}}. \tag{2}$$

where, I is the selection intensity (Bulmer, 1985) and is defined as the expected increase in the average fitness of a population after selection is performed upon a population whose fitness is distributed according to a unit normal distribution. The selection intensity for tournament selection depends on the tournament size, s, and can be approximated by the relation (Blickle & Thiele, 1995):

$$I = \sqrt{2\left(\ln(s) - \ln\left(\sqrt{4.14\ln(s)}\right)\right)}. \tag{3}$$

Equation 2 can be rewritten as

$$\mu_{t+1} - \mu_t = \frac{I}{\rho_e}\sigma_t \tag{4}$$

where, $\rho_e = \sqrt{1 + (\sigma_N^2/\sigma_t^2)}$, is the *duration-elongation factor* (Goldberg, in press). Note that for a non-zero noise, $\rho_e > 1$, and the increment in the average fitness after selection would be less than that when the noise is absent. In other words, the presence of external noise, elongates the convergence time, and this elongation is quantified by ρ_e.

Assume that ρ_e is a constant, and is equal to $\sqrt{1 + \left(\sigma_N^2/\sigma_f^2\right)}$, where σ_f^2 is the initial fitness variance. Note that for OneMax problem, $\mu_t = \ell p_t$, and $\sigma_t^2 = \ell p_t(1 - p_t)$, where p_t is the proportion of correct BBs at time t. Using these expressions, equation 4 can be written as

$$p_{t+1} - p_t = \frac{I}{\rho_e\sqrt{\ell}}\sqrt{p_t(1 - p_t)}. \tag{5}$$

Approximating the above difference equation by a differential equation, and integrating it with the initial condition, $p_0 = 0.5$ (randomly initialized population), gives us

$$p_t = \frac{1}{2}\left(1 + \sin\left(\frac{It}{\rho_e\sqrt{\ell}}\right)\right). \tag{6}$$

Equating $p_t = 1$, in the above equation we can solve for the convergence time:

$$t_{\text{conv}} = \frac{\pi\sqrt{\ell}}{2I}\sqrt{1 + \frac{\sigma_N^2}{\sigma_0^2}}. \tag{7}$$

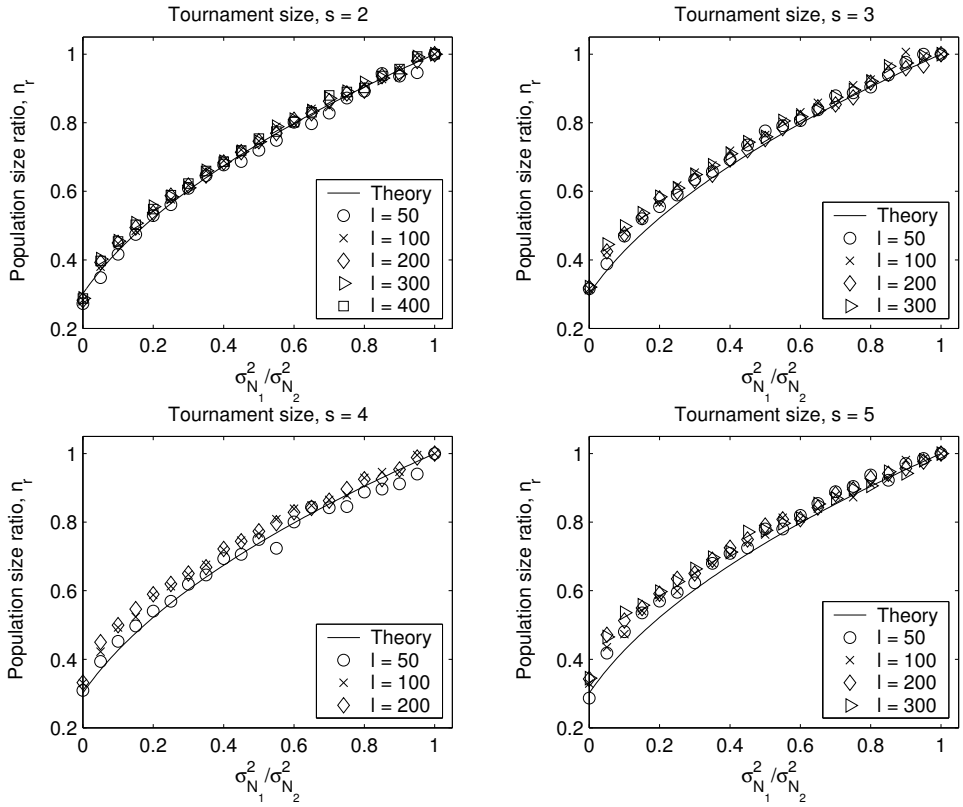


Figure 2: Empirical verification of population-size-ratio model (equation 10).

It must be noted that in deriving the above convergence-time model we assumed ρ_e to be a constant. However, ρ_e changes over time and more accurate solutions for equation 4 exist (Sastry, 2001).

In this study, we are interested in the relative value of convergence times, rather than the absolute values. Specifically, we are interested in the ratio of convergence time when fitness function f_1 is employed to that when fitness function f_2 is employed. This convergence-time ratio is given by

$$t_{c,r} = \frac{t_{\text{conv}}(\sigma_{N_1})}{t_{\text{conv}}(\sigma_{N_2})} = \left(\frac{\sigma_f^2 + \sigma_{N_1}^2}{\sigma_f^2 + \sigma_{N_2}^2} \right)^{\frac{1}{2}}. \tag{8}$$

It should be noted that using more accurate solutions for equation 4 does not improve the accuracy of theoretical model significantly

4.2 Population Size

The previous section presented a convergence-time model for tournament and other I-constant selection schemes. The other factor required to determine complexity is the population-sizing model which is presented in this section. Population size is an important factor in determining the solution quality through a GA run. Adequate population size is required not only to ensure a good number of initial BB supply, but also a good decision-making between competing BBs.

Goldberg, Deb, and Clark (1992) proposed a practical population-sizing bounds for selectorecombinative GAs. Their model was based on deciding correctly between the best and the next best BB in a partition in the presence of noise arising from other partitions. More recently, Harik, Cantú-Paz, Goldberg, and Miller (1997) refined the population-sizing model of Goldberg et al. (1992) to compute a tighter bound on the population size. They incorporated both the initial BB supply model and the decision-making model in the population-sizing relation. Miller (1997) extended the population-sizing model of Harik et al. (1997) for noisy environments.

The following population-sizing model for noisy environments developed by Miller (1997) is used in the current study:

$$n = -\frac{\sqrt{\pi}}{2d} \chi^k \log(\alpha) \sqrt{\sigma_f^2 + \sigma_N^2}, \tag{9}$$

where, d is the signal difference and is given by the

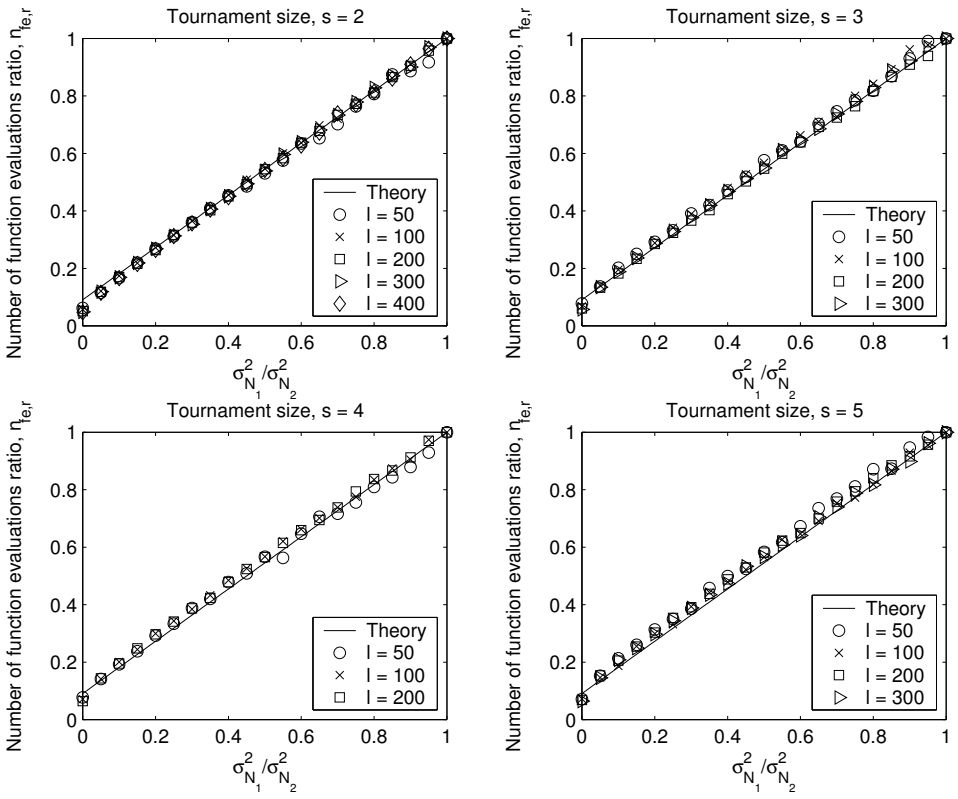


Figure 3: Comparison of empirical and theoretical results for the ratio of total number of function evaluations.

fitness difference of the best and the second best BB, χ is the alphabet cardinality, k is the BB size, and α is the failure rate.

The ratio of population size required to yield a solution of the same quality when fitness function f_1 is used to that when fitness function f_2 is used is then given by

$$n_r = \frac{n(\sigma_{N_1})}{n(\sigma_{N_2})} = \left(\frac{\sigma_f^2 + \sigma_{N_1}^2}{\sigma_f^2 + \sigma_{N_2}^2} \right)^{\frac{1}{2}}. \qquad (10)$$

4.3 Number of Function Evaluations

Using equations 8 and 10, we can obtain the ratio of total number of function evaluations taken if fitness function f_1 is used to those taken if fitness function f_2 is used to obtain solution of the same quality.

$$n_{fe,r} = \frac{n_{fe}(\sigma_{N_1})}{n_{fe}(\sigma_{N_2})} = n_r t_{c,r} = \frac{\sigma_f^2 + \sigma_{N_1}^2}{\sigma_f^2 + \sigma_{N_2}^2}, \qquad (11)$$

4.4 Model Validation

This section empirically verifies the models presented in the previous sections. The empirical results are obtained for the OneMax problem with string lengths ℓ = 50, 100, 200, 300, and 400. Tournament selection without replacement with tournament sizes of s = 2, 3, 4, and 5 is used. Uniform crossover with crossover probability of 1.0 is employed to ensure effective mixing of BBs. The noise variance of fitness function f_2 is taken to be $10\sigma_f^2$ and the noise variance of function f_1 is varied from 0 to $10\sigma_f^2$.

The convergence-time ratio predicted by equation 8 is verified with empirical results and is shown in figure 1. For computing the convergence time, a GA run is terminated if the proportion of correct BBs reaches a value greater than or equal to $(\ell-1)/\ell$. The population size is determined by the following relation (Goldberg, Deb, & Clark, 1992): $n = 8(\sigma_f^2+\sigma_N^2)$. This is a conservative estimate, and is used to reduce the population-sizing effects. The empirical results are averaged over 50 independent runs. Figure 1 clearly validates the convergence-time model of equation 8. Furthermore, as the model predicts, the empirical results show that the convergence-time ratio is independent of ℓ and s values if the ratio of noise variance to the initial fitness variance is constant.

For computing n_r and $n_{fe,r}$, a GA run was terminated when all the individuals in the population converged

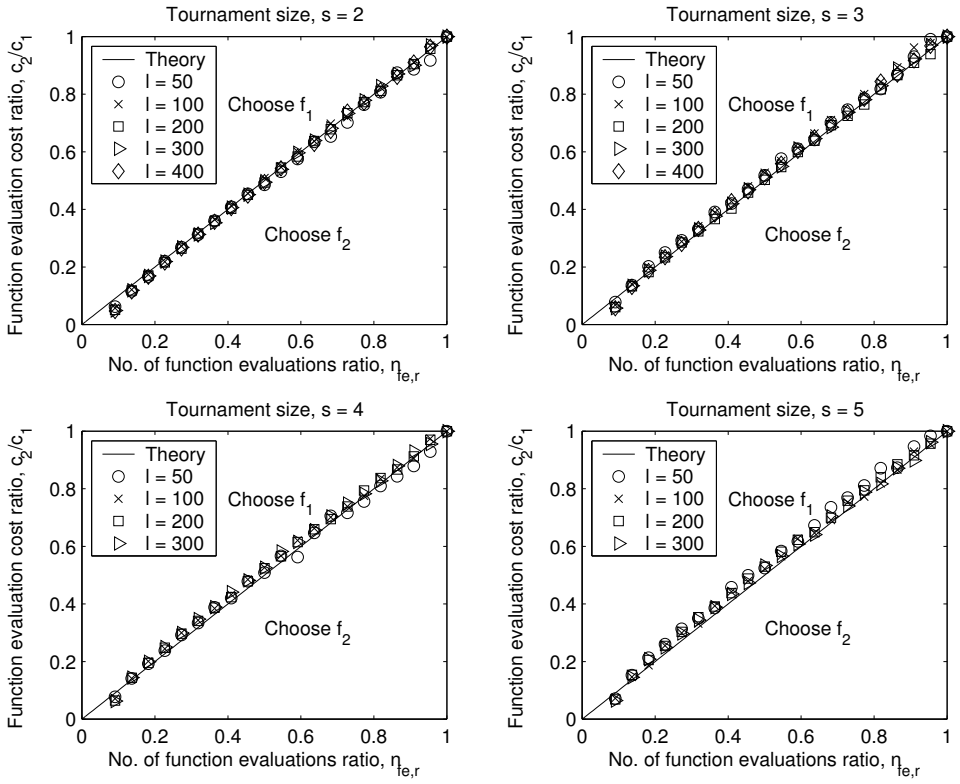


Figure 4: Verification of the optimal decision making between fitness functions with differing variance values.

to the same fitness value. The average number of correctly converged BBs are computed over 50 independent runs. The minimum population size or the total number of function evaluations required for the GA to correctly converge on an average to at least $m-1$ BBs ($\alpha = 1/m$), is determined by the bisection method. The results are averaged over 25 bisection runs.

The population-size ratio predicted by equation 10 is verified with empirical results in figure 2. The prediction of the ratio of total number of function evaluations (equation 11) is compared to the empirical results in figure 3. The results show that the models agrees with empirical results over a broad range of parameter values (specifically, noise variance, problem-size, and tournament-size values).

5 Optimal Decision

As mentioned earlier, we have to decide between two fitness functions, one with low variance, but high cost, and the other with high noise but low cost. The ratio of total cost of employing fitness function f_1 to that of employing fitness function f_2 to obtain solution of the same quality is given by

$$\frac{c_{\text{tot},1}}{c_{\text{tot},2}} = \frac{c_1 n_{fe,1}}{c_2 n_{fe,2}} = \frac{c_1}{c_2}\left(\frac{\sigma_f^2 + \sigma_{N_1}^2}{\sigma_f^2 + \sigma_{N_2}^2}\right), \qquad (12)$$

where, $c_{\text{tot},1}$ is the total cost of employing fitness function f_1, and $c_{\text{tot},2}$ is the total cost of employing fitness function f_2. From the above relation, we can summarize the optimal decision as follows:

- If $c_2/c_1 > (\sigma_f^2 + \sigma_{N_1}^2)/(\sigma_f^2 + \sigma_{N_2}^2)$, then use f_1.
- If $c_2/c_1 < (\sigma_f^2 + \sigma_{N_1}^2)/(\sigma_f^2 + \sigma_{N_2}^2)$, then use f_2.
- If $c_2/c_1 = (\sigma_f^2 + \sigma_{N_1}^2)/(\sigma_f^2 + \sigma_{N_2}^2)$, then either f_1 or f_2 can be used.

This decision making process is shown pictorially in figure 4, where the theory is verified with empirical results. The figure plots the cost ratio of fitness functions for different values of fitness variance ratios. The empirical results shown are obtained for the OneMax problem with string lengths, $\ell = 50, 100, 200, 300$, and 400. A selectorecombinative GA with tournament selection without replacement and uniform crossover is used for this purpose.

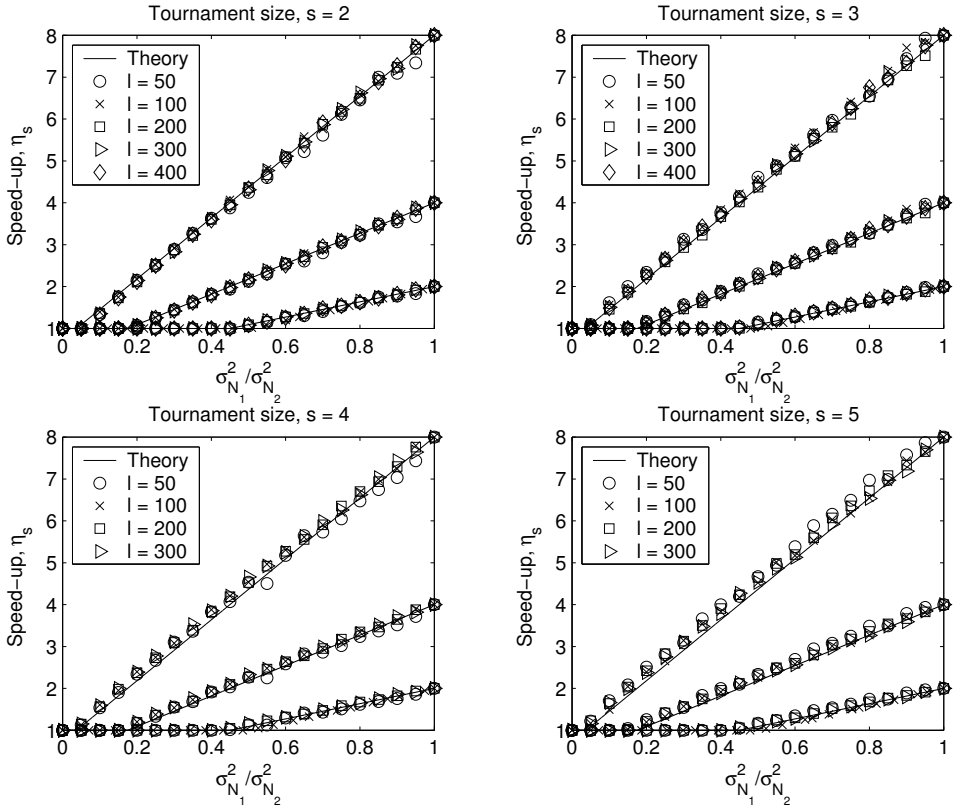


Figure 5: Empirical verification of speed-up predicted by equation 13.

Speed-up is defined as the ratio of the total cost of using a high-cost, low-variance fitness function to the total cost of using a low-cost, high-variance fitness function. Therefore, speed-up obtained by using the aforementioned optimal decision is given by

$$\eta_s = \begin{cases} \frac{c_{\mathrm{tot},1}}{c_{\mathrm{tot},2}} & \frac{c_1}{c_2} > \frac{\sigma_f^2+\sigma_{N_2}^2}{\sigma_f^2+\sigma_{N_1}^2} \\ 1.0 & \text{elsewhere} \end{cases} \tag{13}$$

This definition of speed-up assumes that one always chooses the more accurate fitness function. The above speed-up measures the improvement in efficiency when a correct decision is made instead of a naive decision. When a decision-making procedure, such as the one developed in this section is not available, the naive choice is the use the more accurate fitness function. Justification for using this definition of speed-up is given elsewhere (Sastry, 2001)

The speed-up predicted by equation 13 is verified with empirical relations in figure 5 for different cost-ratio, problem-size, and tournament-size values. The results clearly indicate the a high speed-up can be obtained if the cost-ratio of the fitness functions (c_2/c_1) is much lower than their fitness variance ratios ($\sigma^2_{f_1}/\sigma^2_{f_2}$).

The key thing is that even though we started with simplified assumptions, the decision-making is somewhat general in nature. The only control parameters in the decision making process are the relative cost and fitness variance values. Using dimensional argument, one can extrapolate the results obtained here to other problem domains. In such cases, the decision will be correct in an order-of-magnitude sense. Therefore, the core message of this section is as follows: If an optimization problem has many different fitness function with differing values of variance, and computational costs, then a fitness function with least product of cost and fitness variance should be employed.

6 Conclusions

This paper addressed the issue of deciding between fitness functions with differing variance and cost values. An approximate, but practical convergence-time model was developed and used along with a population-sizing model to develop a decision-making strategy and to predict speed-up. Although in this paper only two fitness functions were considered, the decision making can be easily extended for more than two fitness functions.

The decision-making suggests that the effect of variance can be handled spatially and the choice of the fitness function depends only on the relative cost and variance ratios of the fitness functions. Significant speed-up can be obtained by employing the decision-making strategy developed in this paper. Based on dimensional arguments, the decision-making strategy presented here, though developed for the OneMax problem, should be applicable to other fitness domains.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163, and the National Science Foundation under grant DMI-9908252. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

Albert, L. A. (2001). *Efficient genetic algorithms using discretization scheduling*. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL.

Bäck, T. (1995). Generalized convergence models for tournament—and (μ, λ)—selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 2–8.

Blickle, T., & Thiele, L. (1995). A mathematical analysis of tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 9–16.

Bulmer, M. G. (1985). *The mathematical theory of quantitative genetics*. Oxford: Oxford University Press.

El-Beltagy, M., Nair, P., & Keane, A. (1999). Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. *Proceedings of the Genetic and Evolutionary Computation Conference*, 196–203.

Fitzpatrick, J. M., & Grefenstette, J. J. (1988). Genetic algorithms in noisy environments. *Machine Learning*, *3*, 101–120.

Goldberg, D. E. (1999). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In Bentley, P. (Ed.), *Evolutionary Design by Computers* (Chapter 4, pp. 105–118). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E. (in press). *Design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Acadamic Publishers.

Goldberg, D. E., & Deb, K. (1991). A comparitive analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69–93.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

Grefenstette, J. J., & Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, 112–120.

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 7–12.

Jin, Y., Olhofer, M., & Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. *Proceedings of the Genetic and Evolutionary Computation Conference*, 786–793.

Keijzer, M., & Babovic, V. (2000). Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations. *Genetic Programming: Third European Conference*, 76–90.

Mandava, V. R., Fitzpatrick, J. M., & Pickens, III, D. R. (1989). Adaptive search space scaling in digital image registration. *IEEE Transactions on Medical Imaging*, *8*(3), 251–262.

Miller, B. L. (1997). *Noise, sampling, and efficient genetic algorithms*. Doctoral dissertation, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL. (Also IlliGAL Report No. 97001).

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, *9*(3), 193–212.

Miller, B. L., & Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4*(2), 113–131.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. continous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Prügel-Bennet, A., & Shapiro, J. L. (1994). An analysis of a genetic algorithm using statistical mechanics. *Physics Review Letters*, *72*(9), 1305–1309.

Ratle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. *Parallel Problem Solving from Nature*, *5*, 87–96.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL. (Also IlliGAL Report No. 2002004).

Genetic Algorithms, Efficiency Enhancement, and Deciding Well with Differing Fitness Bias Values

Kumara Sastry and **David E. Goldberg**
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
{ksastry,deg}@uiuc.edu

Abstract

This study develops a decision-making strategy for deciding between fitness functions with differing bias values. Simple, yet practical facetwise models are derived to aid the decision-making process. The decision making strategy is designed to provide maximum speed-up and thereby enhance the efficiency of GA search processes. Results indicate that bias can be handled temporally and that significant speed-up values can be obtained.

1 Introduction

Since the inception of genetic algorithms (GAs) (Holland, 1975), significant progress has been made in designing and analyzing them. A design decomposition has been proposed for the development of *competent* GAs and much progress has been made along these lines (Goldberg, 1999). Competent GAs take problems that were intractable with first generation GAs and render them tractable, oftentimes requiring only a subquadratic number of function evaluations.

However, for large-scale problems, the task of computing even a subquadratic number of function evaluations can be daunting. This is especially the case if the fitness evaluation is a complex simulation, model, or computation. Therefore, one usually resorts to approximate fitness functions that are less expensive to compute. However, approximations introduce error in assessing the solution quality. Also, we may have to choose from many fitness functions with differing error and cost values, and that choice has a large impact on the computational resources and the solution quality.

At present, practitioners make the choice among fitness function alternatives on an ad hoc basis. Therefore, we need to investigate which fitness function should be used under what scenarios. Furthermore, one has to recognize that error introduced through approximations comes in two flavors: Bias, and variance (Keijzer & Babovic, 2000). The decision-making strategy depends on whether variance or bias dominates the error. We have considered the presence of bias and variance in isolation to demonstrate this difference and to ease the analytical burden.

This paper investigates decision making under the presence of bias, while the decision making under the presence of variance is developed elsewhere (Sastry, 2001). Specifically, we investigate the decision making between two fitness functions with differing bias values. A fitness function with higher bias value will yield a more inaccurate solution when compared to the function with a lower bias value. This inaccuracy can be eliminated temporally (not spatially). That is, using the spatial approach—sampling the high-bias fitness function—does not eliminate the effect of bias and yields an inaccurate solution.

On the other hand, a high-bias, low-cost function can be used during the initial few generations of the evolutionary process to obtain a crude solution. The low-bias, high-cost fitness function can then be used (later part of genetic search) to refine the genetic search and to obtain a more accurate solution. The generation at which the fitness functions are switched, called the *switching time* is an important factor in determining the speed-up. The objective of this study is to utilize facetwise models to predict the optimal switching time that yields greatest speed-up and to develop a decision-making strategy to handle bias in fitness functions.

This paper is organized as follows. Section 2 briefly discusses some previous work on handling error in fitness functions. The specific problem that we solve is defined in section 3. Section 4 defines the test problem used for developing models. A convergence-time model that incorporates bias in fitness functions is derived in

section 5. Section 6 develops models for predicting the optimal switching time and the speed-up. Finally, section 7 presents key conclusions of the study.

2 Related Work

Efficiency enhancement is essential for solving large-scale, complex search problems. One such technique is evaluation relaxation, in which the computation burden is reduced by utilizing inexpensive, but error-prone fitness assignment procedures instead of an expensive, but accurate fitness function.

Grefenstette and Fitzpatrick (1985) studied the use of approximate evaluations for an image registration problem. Follow-up studies (Fitzpatrick & Grefenstette, 1988; Mandava, Fitzpatrick, & Pickens, 1989) have further analyzed the utility of approximate fitness evaluations. However, these studies were largely empirical, and a design methodology for handling external noise was developed only recently (Miller & Goldberg, 1995; Miller, 1997). These studies consider only the effects of variance alone, and effects of bias, albeit to a limited extent has also been investigated (Jin, Olhofer, & Sendhoff, 2000; Albert, 2001). For further details on these and other studies on approximate fitness functions in GAs, the interested reader should consult the review presented elsewhere (Sastry, 2001).

3 Problem Definition

Consider two fitness function, f_1 and f_2 for a search problem with bias values of b_1 and b_2 respectively. That is, the optimal solution when f_1 is used is $\mathbf{x}^* + b_1$ and that when f_2 is used is $\mathbf{x}^* + b_2$. Here $\mathbf{x}^*$ is the true optimal solution. The computational costs of f_1 and f_2 are c_1 and c_2 respectively. Furthermore, $b_1 < b_2$ and $c_1 > c_2$. An illustration of the fitness functions with different bias values is shown in figure 1. The figure shows a single variable unimodal fitness functions with and without bias. Note that the optimal value of the fitness functions need not be the same.

Implicitly, we assume that some building blocks (BBs) of f_1 and f_2 are different and others are the same. We recognize that this assumption might not hold true if the biased fitness function introduces multiple false optima. However, this study is the first step toward developing a decision making strategy for handling bias in fitness functions and it serves as a starting point for the analysis of more complex cases. It is important to note that the proposed models can be extended and applied to real-world problems (Albert, 2001).

Since f_1 and f_2 share some BBs, f_2 can be used for the first few generations to obtain good convergence on the BBs shared by both the fitness functions. Fitness function f_1 can then be used to obtain a solution of better accuracy (lower bias). The time, t_s, at which we change from f_2 to f_1 is called the *switching time*. The objective of this study is to optimize the switching time to maximize speed-up and thus develop a decision-making strategy for choosing the correct fitness function. To develop models for solving the problem defined above, we need to first construct a test function. One such test function used in this study is described in the following section.

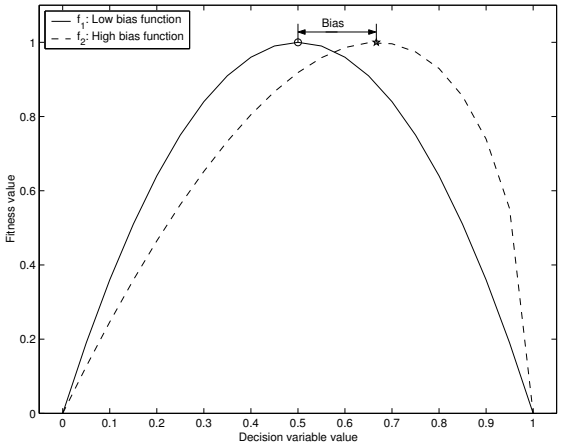


Figure 1: Fitness functions with different bias values.

4 Test Function

The test function used in this study is the *weighted OneMax* defined as:

$$f = \sum_{i=1}^{\ell} w_i x_i, \tag{1}$$

where, x_i is the value of the ith allele and w_i is the weight associated with it. Similar to the OneMax function, the weighted OneMax is a linear unimodal function and the BBs are independent of each other. Therefore, the weighted OneMax function reduces the analytical burden for developing models considerably. Furthermore, fitness functions with differing bias values can be considered as weighted OneMax functions with different weights.

The BBs are uniformly scaled—that is, contribution of every BB to the fitness is equal in magnitude—-if the weights, w_i, are restricted to be either ± 1. Then, the fitness variance of a randomly generated population is equal to that for an OneMax problem. This further eases the analytical burden and the required popu-

lation size does not change with differing bias values. Therefore, we only need to develop a convergence-time model, which is presented in the next section.

5 Convergence-Time Model

Understanding time in GAs is one of the critical factors for a successful design of GAs (Goldberg, in press). Convergence-time model helps us in predicting the scale-up behavior of GAs. Existing studies on understanding time in GAs can be broadly classified into three approaches: (1) *Takeover-time models*, where the growth of the best individual in the population is analyzed (Goldberg & Deb, 1991), (2) *Selection-Intensity models*, where the dynamics of average fitness of the population is analyzed (Mühlenbein & Schlierkamp-Voosen, 1993; Bäck, 1995; Miller & Goldberg, 1995), and (3) *Higher-Order-Cumulant models*, where the dynamics of the average and higher order cumulants of fitness of the population are analyzed (Blickle & Thiele, 1995; Prügel-Bennet & Shapiro, 1994).

In contrast to selection-intensity models, higher-order-cumulant models do not yield closed-form solutions. Therefore, a selection-intensity-based convergence-time model is developed in this paper. For this purpose consider two weighted OneMax functions f_1 and f_2:

$$f_1 = \sum_{i=1}^{\ell} w_i x_i, \tag{2}$$

$$f_2 = \sum_{i=1}^{\ell} w'_i x_i. \tag{3}$$

Without loss of generality assume that the fitness function f_1 has zero bias and that the weights w_i and w'_i are assigned as follows:

$$w_i = \begin{cases} 1 & 1 \le i \le \ell_1 \\ -1 & \ell_1 + 1 \le i \le \ell \end{cases}, \tag{4}$$

$$w'_i = \begin{cases} 1 & 1 \le i \le \ell_1 + b \\ -1 & \ell_1 + b + 1 \le i \le \ell \end{cases}, \tag{5}$$

where, b is the bias. That is, f_1 and f_2 share $\ell - b$ BBs and differ only in b alleles (in this case BBs). For example, the correct BB in any one of the b alleles for f_1 is 1 and for f_2 it is 0.

Note that initially, fitness function f_2 is used in the initial phase ($t < t_s$) of the genetic search. Assuming a uniform BB convergence, and a Gaussian fitness distribution, the expected average fitness of the population after selection is given by (Mühlenbein & Schlierkamp-Voosen, 1993):

$$\mu_{t+1} = \mu_t + I\sigma_t, \tag{6}$$

where, I is the selection intensity and is defined as the expected increase in the average fitness of a population after selection is performed upon a population whose fitness is distributed according to a unit normal distribution. Selection intensity is constant for tournament selection and is approximately given as a function of tournament size s by the following relation (Blickle & Thiele, 1995):

$$I = \sqrt{2\left(\log(s) - \log\left(\sqrt{4.14\log(s)}\right)\right)}. \tag{7}$$

Since fitness function f_2 is used in the first phase ($t \le t_s$) of the run, the mean ($\mu_{f_2,t}$) and variance ($\sigma^2_{f_2,t}$) of fitness are given by

$$\mu_{f_2,t} = \ell p_t - (\ell - \ell_1 - b), \tag{8}$$

$$\sigma^2_{f_2,t} = \ell p_t (1 - p_t), \tag{9}$$

where, p_t is the proportion of ones at time t. Using the mean and variance values in equation 6, we obtain

$$p_{t+1} - p_t = \frac{I}{\sqrt{\ell}}\sqrt{p_t(1-p_t)}. \tag{10}$$

Approximating the above difference equation by a differential equation and integrating it yields

$$p_t = \frac{1}{2}\left[1 - \cos\left(\frac{It}{\sqrt{\ell}} + 2\sin^{-1}\sqrt{p_0}\right)\right]. \tag{11}$$

Assuming that the initial population is randomly generated, we have $p_0 = 0.5$, and we get the following expression for the proportion of correct BBs as a function of time:

$$p_t = \frac{1}{2}\left[1 + \sin\left(\frac{It}{\sqrt{\ell}}\right)\right]. \tag{12}$$

The proportion of correct BBs at switching time t_s is therefore given by

$$p_{t_s} = \frac{1}{2}\left[1 + \sin\left(\frac{It_s}{\sqrt{\ell}}\right)\right]. \tag{13}$$

At the switching time t_s, the low bias fitness function f_1 is used instead of the high bias fitness function f_2. Hence, the proportion of correct BBs changes. Since both f_1 and f_2 share $\ell - b$ BBs, the proportion of correct BBs for those BBs remains the same. That is the proportion of correct BBs for the $\ell - b$ is p_{t_s}. However, since f_1 and f_2 do not share b BBs, the proportion of correct BBs, for the b alleles is $1 - p_{t_s}$. This implies that there are two proportions of correct BBs one for $(\ell - b)$ alleles and the other for b alleles. The adjusted

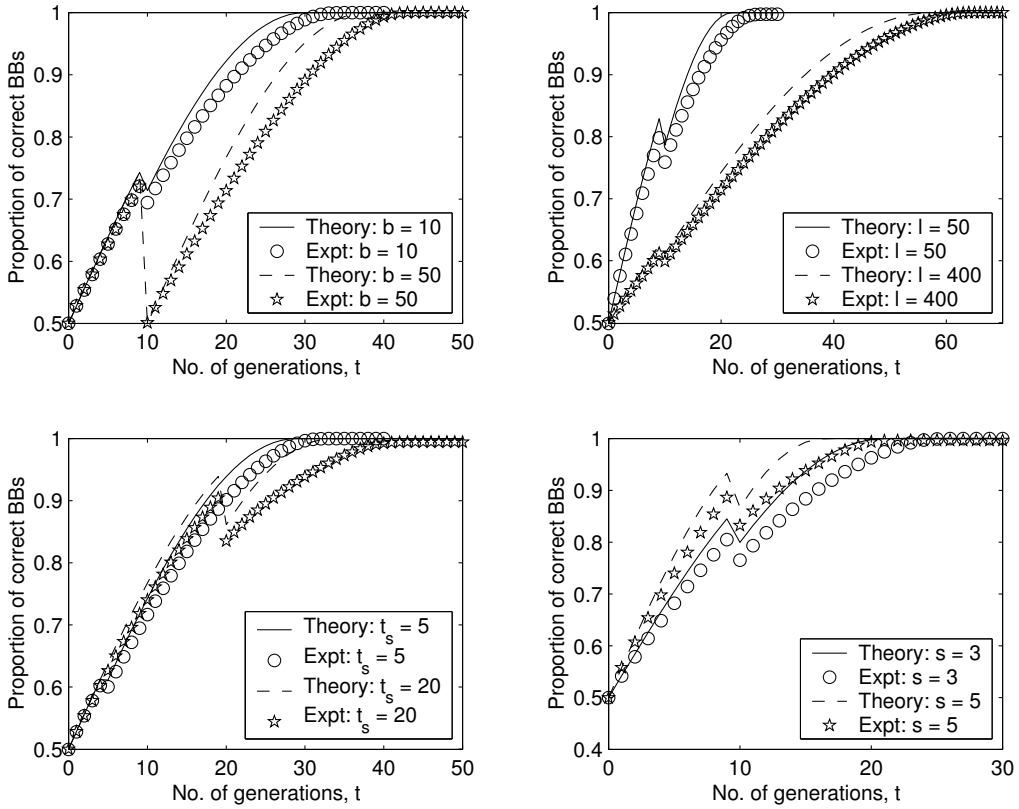


Figure 2: Empirical verification of the proportion of correct building blocks predicted by equations 12, 14, and 16 for different values of b, ℓ, t_s, and s.

proportion of correct BBs for the overall string, p'_{t_s} is given by

$$\begin{aligned} p'_{t_s} &= \frac{1}{\ell}\left[(\ell - b)p_{t_s} + b\,(1 - p_{t_s})\right], \\ &= \left(1 - 2\frac{b}{\ell}\right)p_{t_s} + \frac{b}{\ell}. \end{aligned} \tag{14}$$

From the selection-intensity model assumption, we know that the number of correct BBs in both $\ell - b$ and b portion are distributed normally. Since these two portions are statistically independent of each other, the number of correct BBs for the overall string , and similarly the fitness is also normally distributed. The mean and variance of fitness at time t ($t \geq t_s$) is given by $\ell p'_t - (\ell - \ell_1)$, and $\ell p'_t(1 - p'_t)$ respectively. Proceeding in the same way as we did for $t < t_s$, results in the following difference equation

$$p'_{t+1} - p'_t = \frac{I}{\sqrt{\ell}}\sqrt{p'_t(1 - p'_t)}. \tag{15}$$

Solving the above equation with the initial condition that at $t = t_s$, $p'_t = p'_{t_s}$, we get

$$p'_t = \frac{1}{2}\left[1 - \cos\left(\frac{I(t - t_s)}{\sqrt{\ell}} + 2\sin^{-1}(\sqrt{p'_{t_s}})\right)\right]. \tag{16}$$

From the above relation for the proportion of correct BBs, we can derive an expression for the convergence time, by equating $p'_t = 1$:

$$t_{\text{conv}} = t_s + \frac{\sqrt{\ell}}{I}\left[\pi - 2\sin^{-1}\left(\sqrt{p'_{t_s}}\right)\right]. \tag{17}$$

The models developed above are verified with empirical results. A selectorecombinative GA with tournament selection without replacement, and uniform crossover scheme is employed for this purpose. The probability of crossover is taken to be 1.0 and mutation is not used. The value of ℓ_1 is kept constant at 25 for all the runs. The population size is determined by the relation $8\sigma_f^2$ (Goldberg, Deb, & Clark, 1992). This population-sizing model overestimates the population size and is used to remove any population-sizing effects. Unless otherwise mentioned the following parameters are used: $\ell = 100$, $s = 2$, $b = \frac{\ell}{10}$, and $t_s = 10$. The empirical results are averaged over 100 independent runs.

The proportion of correct BBs predicted by equation 12, 14, and 16 is validated by empirical results. The figures plot the proportion of correct BBs as a

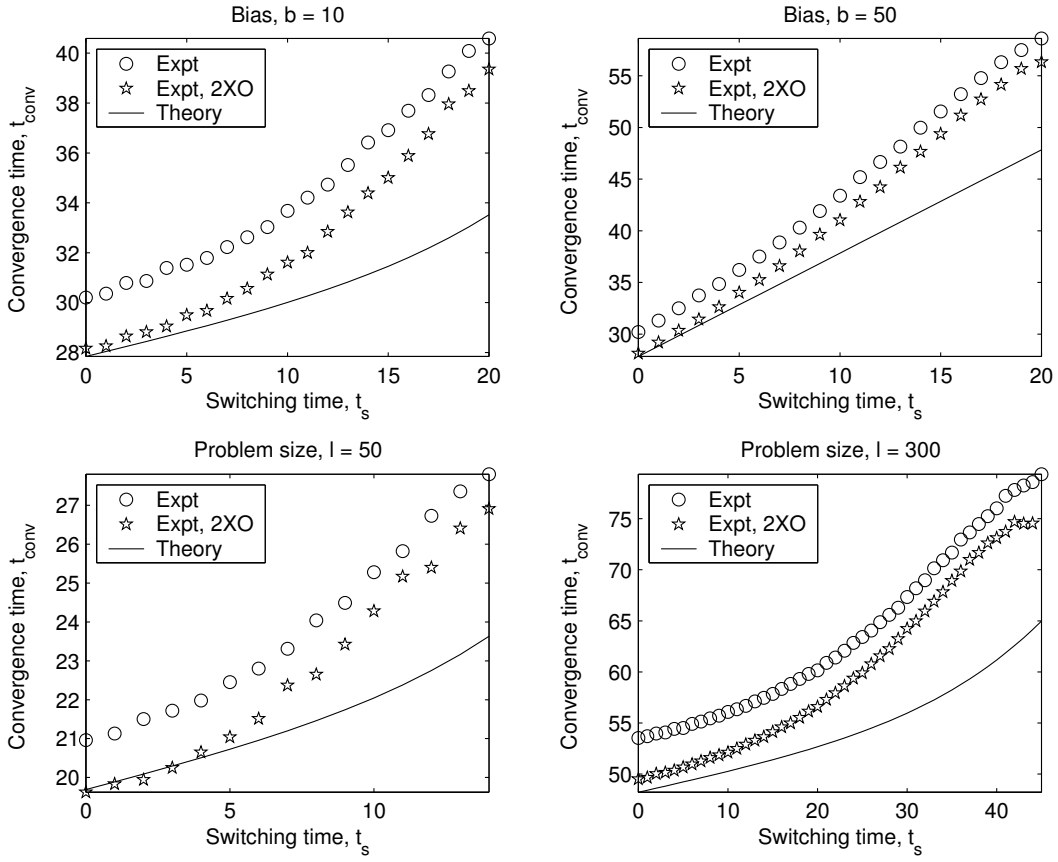


Figure 3: Empirical verification of convergence-time models (equation 17) for different bias values.

function of time. Different values of b, ℓ, t_s, and s are used to validate the model and are shown in figure 2. The results show that the model capture the dynamics accurately over a considerable range of parameter values. The discrepancy between the model and empirical results are due to hitch-hiking and can be further decreased by using multiple crossovers or using a population-wise crossover (Thierens & Goldberg, 1994).

The convergence-time model (equation 17) is compared to empirical results for different bias and problem-size values are shown in figure 3. The figure plots the convergence time as a function of switching time. The empirical results for the case where recombination is applied twice every generation is also shown in the figures. As expected the agreement between the theoretical and experimental results increases when multiple crossover is applied. Note that the compressed convergence-time scale in figure 3 exaggerates the error and the model accuracy is comparable to existing models for other problem domains.

With the convergence-time model at hand, we will now proceed to derive an expression for the optimal switching time. The speed-up that can be obtained by using the optimal switching time is also estimated in the next section.

6 Optimal Switching Time

From the problem definition and the convergence-time model (equation 17), total cost of function evaluation is then given by

$$\begin{aligned} n_{fe} &= n\left(c_2 t_s + c_1(t_{\text{conv}} - t_s)\right), \\ &= nc_2\left(t_s + c_r(t_{\text{conv}} - t_s)\right), \end{aligned} \tag{18}$$

where, $c_r = c_1/c_2$ is the ratio of cost of the high-cost fitness function to the cost of the low-cost fitness function. Employing model 2 (equation 17) for the convergence time, the above equation can be written as

$$n_{fe} = nc_2\left(t_s + c_r\frac{2\sqrt{\ell}}{I}\left[\frac{\pi}{2} - \sin^{-1}\left(\sqrt{p'_{t_s}}\right)\right]\right). \tag{19}$$

We can define the total number of function evaluations in terms of time units by dividing the above equation

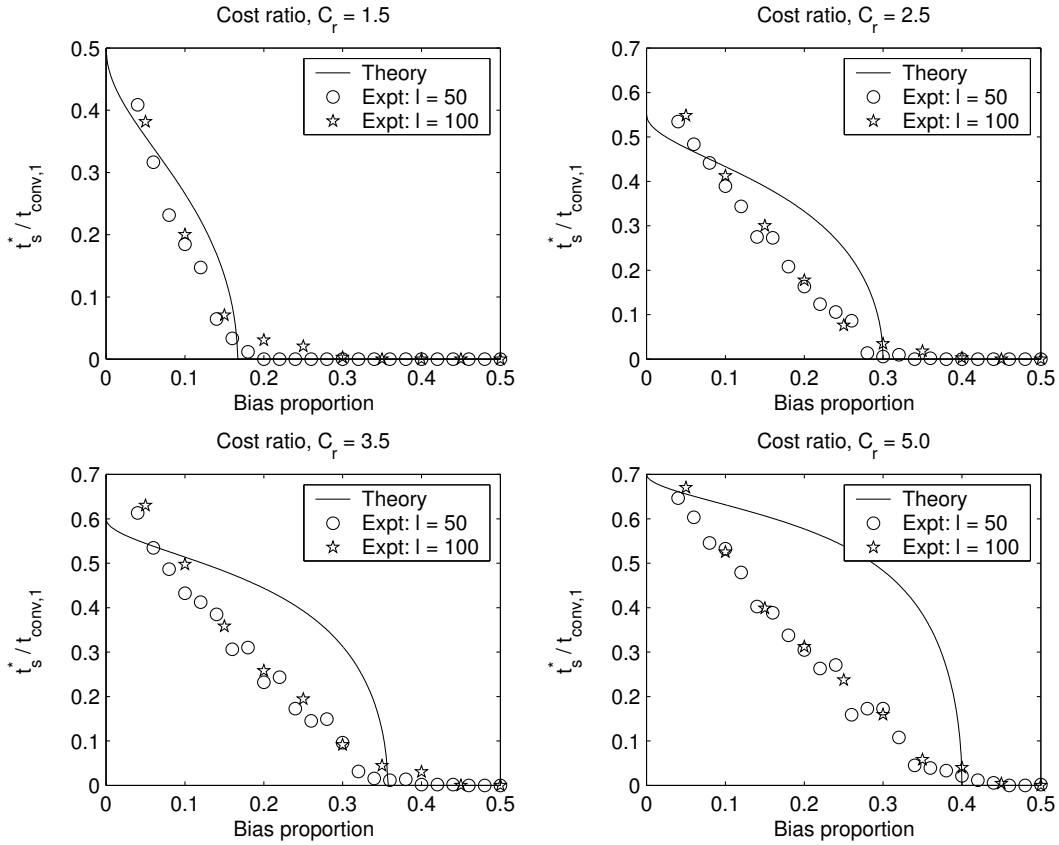


Figure 4: Verification of optimal switching-time model (equation 23).

by nc_2:

$$n'_{fe} = t_s + c_r \frac{2\sqrt{\ell}}{I}\left[\frac{\pi}{2} - \sin^{-1}\left(\sqrt{p'_{t_s}}\right)\right]. \qquad (20)$$

Our objective is to determine t_s that minimizes n'_{fe} (note that this is same as minimizing n_{fe}), which is given by solving

$$\begin{aligned} \frac{\partial n'_{fe}}{\partial t_s} &= 0, \\ 1 - c_r \frac{\sqrt{\ell}}{I} \frac{1}{\sqrt{p'_{t_s}(1-p'_{t_s})}} \frac{\partial p'_{t_s}}{\partial t_s} &= 0. \end{aligned}$$

The optimal switching generation, t_s^*, that minimizes n_{fe} when $c_r \geq \ell/(\ell - 2b)$, comes out to be

$$t_s^* = \frac{\sqrt{\ell}}{I} \cos^{-1}\left[\frac{2\sqrt{\frac{b}{\ell}\left(1-\frac{b}{\ell}\right)}}{\left(1-\frac{2b}{\ell}\right)\sqrt{c_r^2-1}}\right]. \qquad (21)$$

When $c_r < \ell/(\ell - 2b)$, $t_s^* = 0$.

Recognizing that the convergence-time when a low-bias, high-cost fitness function is used is given by (Bäck, 1995)

$$t_{\text{conv},1} = \frac{\pi\sqrt{\ell}}{2I},$$

and dividing equation 21 with the above quantity, we obtain the a dimension-less expression for the optimal switching time when $c_r \geq 1/(1-2\beta)$:

$$\frac{t_s^*}{t_{\text{conv},1}} = \frac{2}{\pi} \cos^{-1}\left[\frac{2\sqrt{\beta(1-\beta)}}{(1-2\beta)\sqrt{c_r^2-1}}\right], \qquad (22)$$

where, $\beta = b/\ell$ is the bias proportion. When $c_r < 1/(1-2\beta)$, $t_s^* = 0$. Equation 22 can be further reduced using the approximation $cos^{-1}(x) \approx \frac{\pi}{2} - x$:

$$\frac{t_s^*}{t_{\text{conv},1}} = \left[1 - \frac{4}{\pi}\frac{\sqrt{\beta\,(1-\beta)}}{(1-2\beta)\sqrt{c_r^2-1}}\right]. \qquad (23)$$

Equation 23 indicates that the strategy of employing the low cost fitness function for the first few generations yields speed-up only if the product of cost ratio, c_r, is above a critical limit which is inversely proportional to the bias proportion. If this is the case, then the optimal switching time is proportional to the

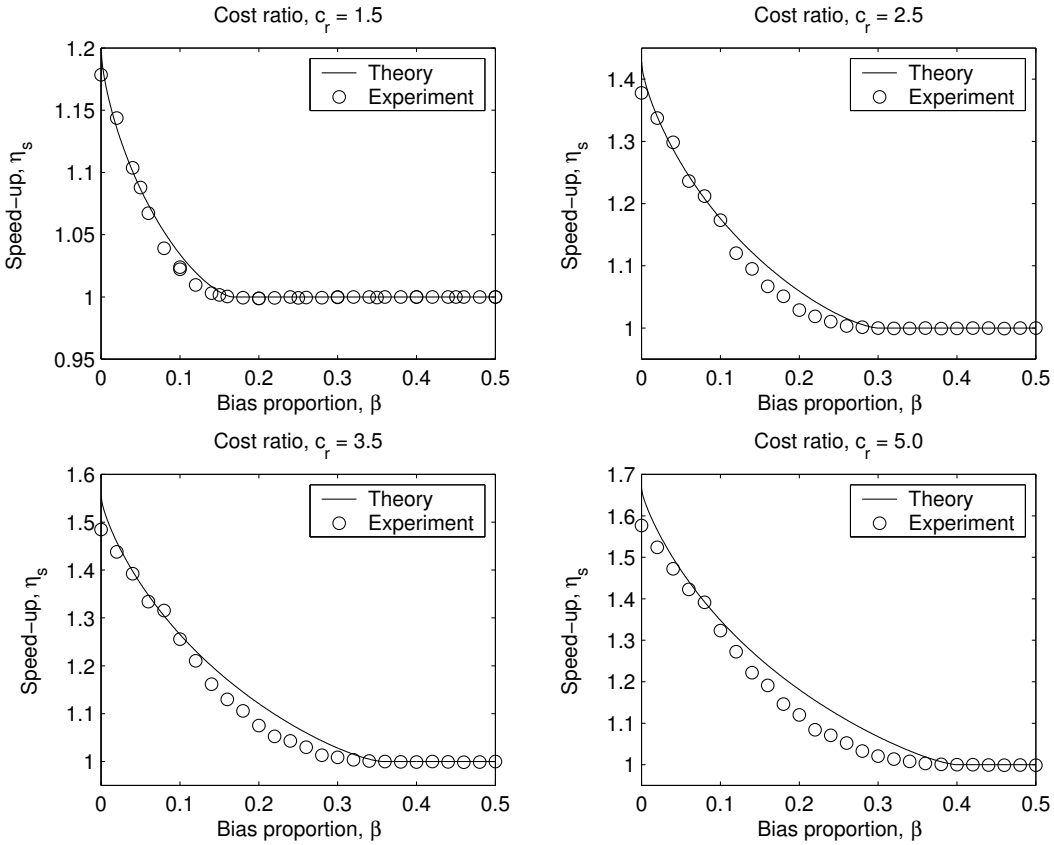


Figure 5: Empirical verification of speed-up prediction (equation 24).

square root of the string length, inversely proportional to the square root of the bias proportion, and inversely proportional to the cost ratio c_r. As expected, if the number of biased bits increases, the switching time decreases, and if the cost ratio increases, the switching time increases. Equation 23 is verified with empirical results in figure 4. The figure plots $t_s^*/t_{\mathrm{conv},1}$ as a function of bias proportion β for different cost-ratio values. A binary tournament selection without replacement, uniform crossover with crossover probability of 1.0 is used. Mutation was not used in obtaining the empirical results. The results are averaged over 50 independent runs.

Using the optimal switching-time given by equation 23, we can compute the speed-up obtained by making the correct decision. Here the speed-up, η_s, is defined as the ratio of the total computational cost incurred if the low-bias fitness function is used to that if the high-bias fitness function is used for t_s^* generations and then the low-bias function is used till the end of the GA run. That is,

$$\eta_s = \frac{n_{fe,1}}{n_{fe,2}} = \frac{nc_1 t_{\mathrm{conv},1}}{n\left[c_2 t_s^* + c_1\left(t_{\mathrm{conv},2} - t_s^*\right)\right]},$$

$$= \frac{c_r}{\left[\left(\frac{t_{\mathrm{conv},2}}{t_{\mathrm{conv},1}}\right) - \left(c_r - 1\right)\tau_s^*\right]}. \quad (24)$$

Where, $\tau_s^* = t_s^*/t_{\mathrm{conv},1}$. Note that the above equation is valid when $c_r \geq 1/(1-2\beta)$. When $c_r < 1/(1-2\beta)$, $\eta_s = 1$. The speed-up predicted by equation 24 is verified with empirical results in figure 5. The figure plots η_s as a function of bias proportion β for different cost-ratio values. Tournament selection without replacement with tournament size $s = 2$ is used. Uniform crossover with crossover probability of 1.0 is employed and mutation is not used. The results are averaged over 50 independent runs.

Figure 5 clearly indicates the improvement in efficiency using the decision-making strategy developed to handle bias in fitness functions. It also validates our hypothesis that bias has to be handled temporally. Furthermore, even though we made some simplifying assumptions the final result for the optimal switching time and the speed-up are in dimensionless quantities and should be easily applicable to other problem domains as well.

7 Conclusions

This paper develops a decision-making strategy for choosing between fitness function with differing bias values. We proposed that bias has to be handled temporally by switching from a high-bias fitness function to a low-bias fitness function. We also hypothesized that an optimal switching time exists and when the fitness functions are switched at this optimal time, the total computation cost will be the minimum. We developed approximate, but practical convergence-time model, and used it to determine the optimal switching time. Based on the computational cost and the total number of function evaluations taken by each fitness function, a decision-making strategy was presented.

The paper shows that bias has to be handled temporally. That is, a high-bias fitness function should be used for coarse-grain optimization and then a low-bias fitness function should be used for fine-grain optimization. Although, we considered only two fitness functions, the decision making can be easily extended for more than two fitness functions. Furthermore, the models developed in this study should provide guidance to GA practitioners in choosing key GA parameters and to provide maximum efficiency enhancement.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163, and the National Science Foundation under grant DMI-9908252. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

Albert, L. A. (2001). *Efficient genetic algorithms using discretization scheduling*. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL.

Bäck, T. (1995). Generalized convergence models for tournament—and (μ, λ)—selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 2–8.

Blickle, T., & Thiele, L. (1995). A mathematical analysis of tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 9–16.

Fitzpatrick, J. M., & Grefenstette, J. J. (1988). Genetic algorithms in noisy environments. *Machine Learning*, *3*, 101–120.

Goldberg, D. E. (1999). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In Bentley, P. (Ed.), *Evolutionary Design by Computers* (Chapter 4, pp. 105–118). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E. (in press). *Design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Acadamic Publishers.

Goldberg, D. E., & Deb, K. (1991). A comparitive analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69–93.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

Grefenstette, J. J., & Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, 112–120.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jin, Y., Olhofer, M., & Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. *Proceedings of the Genetic and Evolutionary Computation Conference*, 786–793.

Keijzer, M., & Babovic, V. (2000). Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations. *Genetic Programming: Third European Conference*, 76–90.

Mandava, V. R., Fitzpatrick, J. M., & Pickens, III, D. R. (1989). Adaptive search space scaling in digital image registration. *IEEE Transactions on Medical Imaging*, *8*(3), 251–262.

Miller, B. L. (1997). *Noise, sampling, and efficient genetic algorithms*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 97001).

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, *9*(3), 193–212.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. continous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Prügel-Bennet, A., & Shapiro, J. L. (1994). An analysis of a genetic algorithm using statistical mechanics. *Physics Review Letters*, *72*(9), 1305–1309.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report no. 2002004).

Thierens, D., & Goldberg, D. E. (1994). Convergence models of genetic algorithm selection schemes. *Parallel Problem Solving from Nature*, *3*, 116–121.

Voronoi Quantized Crossover for Traveling Salesman Problem

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

Abstract

It is known that the performance of a genetic algorithm depends on the survival environment and the reproducibility of building blocks. In this paper, we propose a new encoding/crossover scheme that uses genic distance which explicitly defines the distance between each pair of genes in the chromosome. It pursues both relatively high survival probabilities of more epistatic gene groups and diverse crossover operators for the high creativity of new schemata. The experimental results on benchmark traveling salesman problems showed remarkable improvement in tour cost and running time over state-of-the-art genetic algorithms for the problem.

1 Introduction

In the context of genetic algorithms (GAs), a specific gene pattern is called schema. Holland showed, by Schema Theorem, that highly fit schemata of short defining lengths and low orders have high survival probabilities in the traditional genetic framework [1]. These short, low-order, high-quality schemata are called *building blocks*. Bui and Moon [2, 3] claimed that, in case of multi-point crossover operators, low-order, high-quality schemata with clustered specific-symbol distributions should serve as building blocks. According to the building block hypothesis, a genetic algorithm seeks near optimal performance through the juxtaposition of building blocks [4]. The performance of a genetic algorithm thus highly depends on its survival environment and reproducibility of building blocks.

In the light of the interrelationship between genes, building blocks are gene groups that have strong *interactions* (or *epistases*) among participating genes in the chromosome. Genes' interaction here means that the contribution of a gene to the chromosomal fitness depends on the values of other genes in the chromosome. The stronger the interactions of genes are, the higher the nonlinearity of the problem is; this makes the problems more difficult [5].

For a given problem representation, the survival probability of a gene group through the crossover is determined by the distribution of genes in the chromosome, while the strength of the epistasis of the gene group is an inherent property of the problem. Therefore, the strategy of locating each gene in the chromosome significantly affects the performance of genetic algorithms. In other words, the positions (or loci) of genes in the chromosome may be placed so as to ensure higher survival probabilities for more epistatic schemata. Inversion is a genetic operator devised for changing the loci of genes dynamically during the genetic process [6, 1]. The efforts to exploit gene positions dynamically are called *linkage learning* [7]. Messy genetic algorithm and fast messy genetic algorithm are examples that implicitly pursue dynamic gene repositioning [8, 9]. The chromosomal encoding with fixed gene positions is called locus-based encoding. A number of studies on *static reindexing* (or *reordering*) of gene positions in locus-based encodings showed performance improvement [10, 11, 12, 13, 14].

The representation power[1] of a genetic algorithm is highly dependent on the *chromosome topology*. That is, the higher the dimension of the chromosome topology is, the higher the representation power it has. A typical chromosome topology is a one-dimensional array. Though one-dimensional array is easy to handle and analyze, it has a poor representation power

[1] Here, high representation power means low degree of distortion. Generally, in representing a graph geometrically, the higher the dimension of representation space is, the lower the degree of distortion it shows [15].

which causes great loss of information contained in the problems. To overcome this, the encoding/crossover schemes with multi-dimensional arrays were suggested and remarkable improvements were reported [16, 17, 18, 19]. Recently, Jung and Moon [20, 21] obtained successful results by applying a crossover based on 2D Euclidean encoding to the 2D Euclidean traveling salesman problem (TSP). They used phenotypes themselves for chromosomal cutting.

In a point of view, the studies of changing the loci of genes are to exploit the interactions among genes implicitly. Increasing the representation power of chromosome topologies also can be understood in this context. Jung and Moon's study may be thought to be an extreme case of such approaches to the 2D Euclidean TSP. We keep the philosophy. In this paper, we suggest a new encoding/crossover scheme which *explicitly* exploits the interactions among genes. In the scheme, the *genic distance* between a pair of genes is defined. We apply this scheme to TSP and compare its performance with state-of-the-art methods.

The rest of this paper is organized as follows. We summarize previous approaches to TSP in Section 2 and explain the proposed genetic operators in Section 3. In Section 4, we provide experimental results. Finally, the conclusion is given in Section 5.

2 Traveling Salesman Problem

Given n cities, the *traveling salesman problem* (TSP) is the problem of finding the shortest Hamiltonian cycle visiting the cities. More formally, given a set of cities $\{c_1, c_2, \ldots, c_n\}$ and the distance $d(c_i, c_j)$ for every pair (c_i, c_j), it is the problem of finding an ordering π that minimizes the following:

$$C(\pi) = \sum_{i=1}^{n-1} d(c_{\pi(i)},\ c_{\pi(i+1)}) + d(c_{\pi(n)},\ c_{\pi(1)}).$$

It is a well known NP-hard problem [22]. Thus one should rely on approximation algorithms that do not guarantee optimal solutions.

For decades, TSP has served as an initial proving ground for new problem solving techniques because of its difficulty, applicability, and the simplicity of definition. Various local optimization algorithms such as 2-opt, 3-opt, Lin-Kernighan (LK) algorithm, and their variants were developed and problem independent techniques such as tabu search, simulated annealing, neural networks, genetic algorithms and ant colonies were applied [23, 24].

Recently, hybrid genetic algorithms which combine local optimization algorithms with the genetic framework have been successfully applied to the problem [25, 26, 20]. LK algorithm [27] is the most popular and powerful local optimization algorithm for TSP. Various crossover operators such as order crossover [28], cycle crossover [29], partially matched crossover [4], edge-recombination crossover [30], and matrix crossover [31] were used for TSP. Distance preserving crossover [32, 26], edge assembly crossover [33, 34, 35], and natural crossover [20, 21] are representative state-of-the-art crossover operators proposed recently.

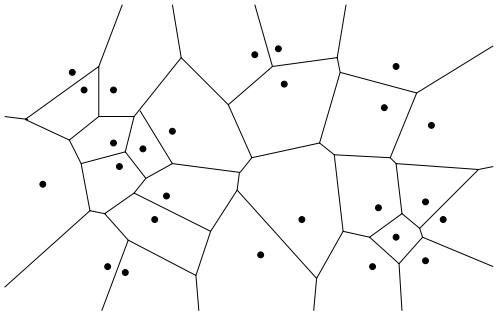


Figure 1: An example of 2D Voronoi regions

```
VQX(n, k, d_g, p_1, p_2)
{
    I ← {1, 2, ..., n};  K ← {1, 2, ..., k};
    Select a subset R = {s_1, s_2, ..., s_k} ⊂ I
        at random;
    for each i ∈ I {
        r[i] ← arg min_{j∈K} {d_g(s_j, i)},  s_j ∈ R;
    }
    for each j ∈ K {
        u[j] ← 0 or 1 at random;
    }
    for each i ∈ I {
        if ( u[r[i]] = 0 ) then o[i] ← p_1[i];
        else o[i] ← p_2[i];
    }
    return o;
}
```

Figure 2: Voronoi quantized crossover

3 New Operators

3.1 Voronoi Quantized Crossover

In Voronoi quantized crossover (VQX), we adapt a chromosome topology in which every gene has a rela-

tive locus determined by the distances between genes, contrary to the others in which every gene has an absolute locus. The distance is called *genic distance*. In a point of view, the chromosome may be thought to be a "complete graph" where each vertex stands for a gene and the edge weight is determined by the epistasis between the two corresponding genes. The graph is directed if the genic distance is asymmetric. By adapting such type of chromosome topology, we aim to represent a chromosome with a minimal degree of distortion.

The name of Voronoi quantized crossover came from Voronoi quantization which is a representative vector quantization method [36]. Vector quantization is a method of approximating arbitrary multi-dimensional vectors by k code vectors each of which represents a subspace of the whole vector space. It is used mostly in data compression. The vector quantization that uses Voronoi regions [37] as the subspaces is called Voronoi quantization. The Voronoi region of a vector is defined to be the nearest neighborhood of the vector. Figure 1 shows an example of Voronoi regions associated with a set of points in 2D Euclidean space.

VQX has a simple structure. Figure 2 shows the pseudo code of VQX where n is the number of genes and k is the crossover degree ranged from 2 to n. The function $d_g : I^2 \to R$ represents the genic distance. The two parents and the offspring are denoted by p_1, p_2 and o, respectively. Following the convention, the notation "arg min" takes the argument that minimizes the value. In VQX, the genic space defined by the genic distance d_g is divided into k Voronoi regions determined by the k randomly selected genes, then a sort of block-uniform crossover [17] is performed on the regions.

VQX has two main properties:

- *Convexity* — Voronoi regions are convex[2] [36]. Therefore, the gene groups of relatively short genic distance have higher survival probabilities than others.
- *Diversity* — It has $\binom{n}{k}2^k$ crossover operators[3].

The first property means that the survival probabilities of gene groups can be controlled by genic distance assignments. We can allow high survival probabilities for building blocks by assigning genic distance inversely proportional to the strength of epistasis. The other means that VQX has a lot of crossover operators. The number of crossover operators affects the creativity of new schemata. The number of crossover operators of a typical k-point crossover is $\binom{n-1}{k}$. For $n = 100$, $k = 6$ and $n = 1000$, $k = 8$, for example, k-point crossover has about 10^{10} and 10^{20} crossover operators, respectively, while VQX has about 10^{11} and 10^{22}. However, we should mention that we do not pursue the maximal number of crossover operators.

The time complexity of VQX is $\Theta(kn)$.

3.2 Survival Probabilities

The survival probability of a gene group[4] (or unspecific schema) in VQX is derived in this section. Given a genic distance measure d_g, a function $h : 2^I \times I \to Z^+$ is defined as

$$h(S,\ i) = |\{l \in I : \forall v \in S,\ d_g(l,\ v) > d_g(i,\ v)\}|,\ S \subset I,\ i \in I \quad (1)$$

where $I = \{1, 2, \ldots, n\}$ and n is the problem size. Given a subset $R = \{s_1, s_2, \ldots, s_k\} \subset I$, the Voronoi region assignment function $r : 2^I \times I \to I$ is defined as

$$r(R,\ i) = \arg\min_{j \in K}\{d_g(s_j,\ i)\},\ i \in I,\ s_j \in R \quad (2)$$

where $K = \{1, 2, \ldots, k\}$. Now, given S and i, the number of R's of k elements that make all v's in S have the same function value $r(R,\ v) = i$ is $\binom{h(S,\ i)}{k-1}$. Assuming that the set R is selected at random, the probability that all genes in a gene group S belong to the same region, i.e., the probability that $r(R,\ j)$'s are the same for all j's in S, is derived as

$$P_{eq}(S) = \frac{\sum_{i=1}^{n}\binom{h(S,i)}{k-1}}{\binom{n}{k}}. \quad (3)$$

In the case of $|S| = 2$, the survival probability $P_{sur}(S)$ of a gene group S is derived as

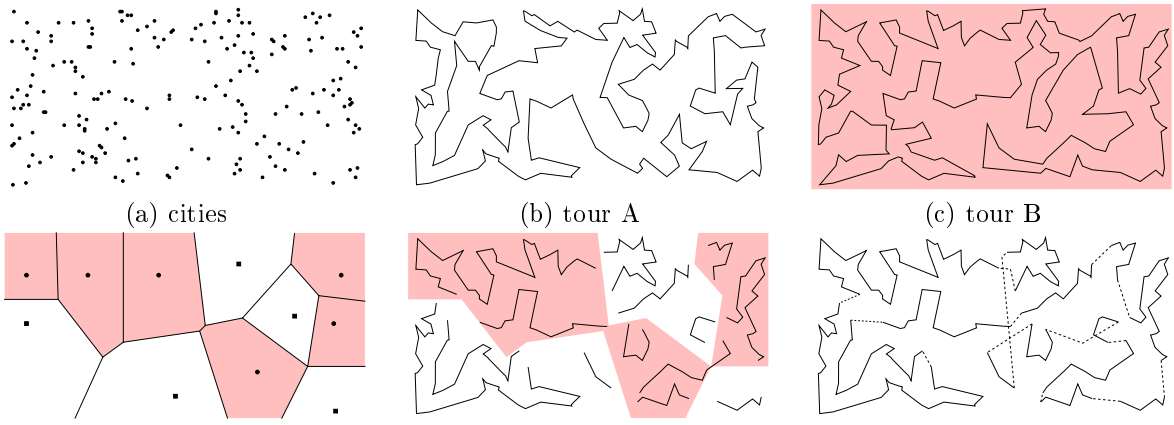
$$\begin{aligned} P_{sur}(S) &= P_{eq}(S) + \frac{1}{2}(1 - P_{eq}(S)) \\ &= \frac{1}{2} + \frac{\sum_{i=1}^{n}\binom{h(S,i)}{k-1}}{2\binom{n}{k}}. \end{aligned} \quad (4)$$

This is used in Section 3.4 to examine the relationship between genic distances and survival probabilities.

[2] A set $S \in R^k$ is *convex* if $a,\ b \in S$ implies that $\alpha a + (1-\alpha)b \in S$ for all $0 < \alpha < 1$.

[3] In fact, we cannot guarantee that the consequent offspring are all distinct. Different quantizations may generate the same offspring, although believed rare.

[4] Generally, a schema is defined by alleles of specific genes. In this paper, we use the term *gene group* rather than schema, because we refer only the set of genes here.



(a) cities (b) tour A (c) tour B

(d) Voronoi regions (e) inherited tour segments (f) repaired tour segments

Figure 4: An example VQX for TSP (kroA200)

```
VQX′(n, k, d_g, p_1, p_2)
{
    I ← {1, 2, ..., n};  K ← {1, 2, ..., k};
    Select a subset R = {s_1, s_2, ..., s_k} ⊂ I
        at random;
    for each i ∈ I {
        r[i] ← arg min_{j∈K} {d_g(s_j, i)},  s_j ∈ R;
    }
    for each j ∈ K {
        u[j] ← 0 or 1 at random;
    }
    for each i ∈ I {
        if ( u[r[i]] = 0 and u[r[p_1[i]]] = 0 )
            then o[i] ← p_1[i];
        else if ( u[r[i]] = 1 and u[r[p_2[i]]] = 1 )
            then o[i] ← p_2[i];
        else o[i] ← nil;
    }
    o ← GreedyRepair(o);
    return o;
}
```

Figure 3: Modified VQX for TSP

3.3 Applying VQX to TSP

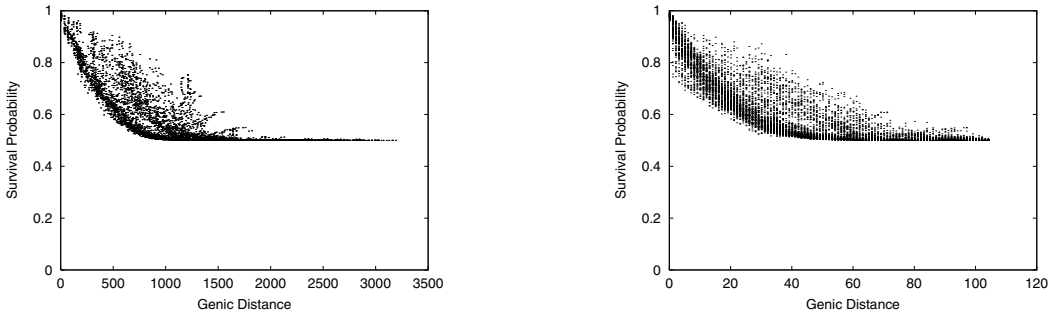
In this paper, the locus-based encoding of [11] is used; one gene is allocated for every city and the gene value represents the index of its next city in the tour. Thus, a solution may be thought to be a mapping s from the set of cities I to I. In TSP, a solution that does not construct a Hamiltonian cycle is infeasible. A solution s is feasible if and only if (i) s is one-to-one and (ii) it has no subcycle. Directly applying the VQX of Figure 2 to TSP may produce infeasible solutions. To avoid this, we need some modification to the crossover.

Figure 3 shows the pseudo code of the modified VQX. The word *nil* is used for the genes whose values are not determined. The consequent solutions have no subcycle but may have genes of *nil* value. In other words, tour segments without any subcycle are created. We use a greedy approach to repair them. In "GreedyRepair()", a segment is selected and connected to its nearest segment to grow into a complete tour. Note that the performance of repairing is not critical here, as a powerful local optimization heuristic follows the crossover and mutation. Figure 4 shows an example of the crossover process. It is obtained by applying VQX′ (in Figure 3) with the genic distance assignment GD1 (described in Section 3.4) to kroA200, a benchmark problem taken from TSPLIB [38]. We use a random tie-breaking in applying the equation (2) in the crossover.

3.4 Genic Distance Assignments

To apply VQX to TSP, the distances among genes (genic distances) are needed. The genic distances may be assigned statically or dynamically in the genetic process. In this paper, the static assignment is used.

Intuitively, an ideal value of a genic distance is a value inversely proportional to the epistasis. This leads to the high survival probabilities of relatively more interactive gene groups because the survival probability of a gene group is inversely proportional to their genic distances in VQX. However, no practical method is known yet for exactly computing epistases; two heuristics are used in this paper. Let I be a set of city indices



(a) GD1 (b) GD2

Figure 5: Survival probability versus genic distance (lin105)

and $d(i,j)$ is the distance from city $i \in I$ to city $j \in I$. The genic distance $d_g(i,j)$ from gene i to gene j is defined in two manners as

- GD1: $d_g(i,j) = d(i,j)$
- GD2: $d_g(i,j) = |\{l : d(i,l) < d(i,j),\ l \in I\}|$.

In GD1, the Euclidean distance itself is used; in GD2, the number of cities closer to city i than city j is used. Usually, d_g is asymmetric in GD2, while it is symmetric as far as d is symmetric in GD1.

Figure 5 shows the results of a simple test to observe the relationship between the genic distance and the survival probability of a gene pair. The horizontal and vertical axes of coordinates represent the genic distance and the survival probability, respectively. The equation (4) was used to acquire the survival probabilities from the genic distances obtained by applying GD1 and GD2 to lin105, an instance from TSPLIB. It shows that the survival probabilities of a close gene pairs in the genic space are high in both cases.

3.5 Heterogeneous Mating

In a preliminary examination, VQX showed faster convergence than the other crossovers in comparison; this may cause the premature convergence of the genetic algorithm. To avoid this, we use a method of mating mutually dissimilar individuals in parallel with VQX. Hollstien called this type of breeding a negative assortive mating [39]. There are various methods, sometimes called niching methods, for maintaining population diversity [40, 41].

Figure 6 shows the pseudo code of the mating used in this paper. First, m individuals are selected from the population P by roulette-wheel selection. Then the most different one from p_1 among them is selected as p_2. Hamming distance[5] is used for the distance function "distance()".

```
MateSelection(P, m, p1)
{
    C ← ∅;
    for i ← 1 to m {
        c ← Selection(P \ ({p1} ∪ C));
        C ← C ∪ {c};
    }
    p2 ← arg max_{c∈C} {distance(p1, c)};
    return p2;
}
```

Figure 6: Heterogeneous mate selection

4 Experimental Results

The genetic algorithm used in this paper is a steady-state hybrid genetic algorithm. Figure 7 shows the template. In the template, n is the problem size, m is the group size in mating, k is the crossover degree, and d_g is the genic distance measure. The two selected parents and the offspring are denoted by p_1, p_2 and o, respectively. The genetic operators and their parameters used in this paper are summarized in the following:

- Population Initialization — Initial solutions are generated at random.
- Population Size — $|P| = 100$.
- Selection — Roulette-wheel selection. The fitness

[5] the number of different edges between two tours.

Table 1: Experimental results of VGA1 and VGA2

Graph (opt)	Xover	OB #	Best (%)		Avg (%)		$\sigma/\sqrt{t}$	Gen	Time (s)
att532	VGA1	97	27686	(0)	27686.37	(0.001)	0.22	3006	103
(27686)	VGA2	95	27686	(0)	27686.69	(0.002)	0.30	3024	109
dsj1000	VGA1	24	18659688	(0)	18659952	(0.001)	24	2803	1026
(18659688)	VGA2	52	18659688	(0)	18659809	(0.001)	13	3470	1251
d2103	VGA1	72	80450	(0)	80470.05	(0.025)	3.22	3874	1084
(80450)	VGA2	76	80450	(0)	80467.07	(0.021)	3.04	4271	1157
pcb3038	VGA1	11	137694	(0)	137707.22	(0.010)	1.40	12234	835
(137694)	VGA2	8	137694	(0)	137706.78	(0.009)	1.22	13021	906
fnl4461	VGA1	0	182573	(0.004)	182607.22	(0.023)	1.97	28518	2011
(182566)	VGA2	0	182571	(0.003)	182605.88	(0.022)	2.21	28992	2057

Table 2: Comparison of VGA with DGA, EGA, and NGA

Graph (opt)	Xover	Best (%)		Avg (%)		$\sigma/\sqrt{t}$	Gen	Time (s)
	DGA	27686	(0)	27692.86	(0.025)	0.75	3971	89
att532	EGA	27686	(0)	27700.51	(0.052)	0.84	13934	271
(27686)	NGA	27686	(0)	27692.13	(0.022)	0.77	3563	167
	VGA2	27686	(0)	**27686.69**	(0.002)	0.30	3024	109
	DGA	18659688	(0)	18660087	(0.002)	78	11267	1038
dsj1000	EGA	18659688	(0)	18679325	(0.105)	1494	41938	1867
(18659688)	NGA	18659688	(0)	18659942	(0.001)	18	3266	1114
	VGA2	18659688	(0)	**18659809**	(0.001)	13	3470	1251
	DGA	80450	(0)	80500.09	(0.062)	5.86	4021	630
d2103	EGA	80450	(0)	80469.82	(0.025)	2.29	82072	8466
(80450)	NGA	80450	(0)	80472.05	(0.027)	4.89	1970	456
	VGA2	80450	(0)	**80467.07**	(0.021)	3.04	4271	1157
	DGA	137699	(0.004)	137751.44	(0.042)	4.24	20261	1408
pcb3038	EGA	137694	(0)	137831.77	(0.100)	7.26	199015	28213
(137694)	NGA	137698	(0.003)	137733.10	(0.028)	3.71	20582	1734
	VGA2	137694	(0)	**137706.78**	(0.009)	1.22	13021	906
	DGA	182593	(0.015)	182822.39	(0.140)	31.80	76331	13728
fnl4461	EGA	182598	(0.018)	182864.60	(0.164)	31.11	338860	160845
(182566)	NGA	182572	(0.003)	182631.82	(0.036)	3.19	84247	8832
	VGA2	182571	(0.003)	**182605.88**	(0.022)	2.21	28992	2057

value f_i of the solution i is calculated as

$$f_i = (C_w - C_i) + (C_w - C_b)/4 \qquad (5)$$

where C_i, C_w, and C_b are the costs of the solution i, the worst solution, and the best solution in the population, respectively. The fitness value of the best solution is five times as great as that of the worst solution in the population.

- Group Size for Mating — $m = 5$.
- Crossover Degree — An empirical value $k = \lfloor \ln n + \frac{1}{2} \rfloor + 2$ is used where n is the problem size and "ln" is the natural logarithm.
- Mutation — Double-bridge kick move [27] was applied once per ten offsprings. Figure 8 shows a symbolic drawing of double-bridge kick move.
- Local Optimization — LK algorithm accelerated by don't-look bit [42] and segment tree [43] was used.
- Replacement — A variant of preselection [44] was used as in [11]. Each offspring is replaced with (i) its more similar parent if the offspring is better, (ii) the other parent if the offspring is better, (iii) the worst solution in the population, otherwise.
- Stop Condition — Until 70 percent of the population converge with the same cost as the best solution in the population. This takes account of the cases that more than one best solution of the same quality competes with each other.

The algorithms were implemented in C on Intel Pentium III 866 MHz running Linux 2.2.14.

```
VGA(n, m, k, d_g)
{
    Initialize population P;
    repeat {
        p_1 ← Selection(P);
        p_2 ← MateSelection(P, m, p_1);
        o ← VQX'(n, k, d_g, p_1, p_2);
        o ← Mutation(o);
        o ← LocalOptimization(o);
        P ← Replacement(P, p_1, p_2, o);
    } until (stop condition);
    return the best of P;
}
```

Figure 7: Steady-state hybrid genetic algorithm for TSP

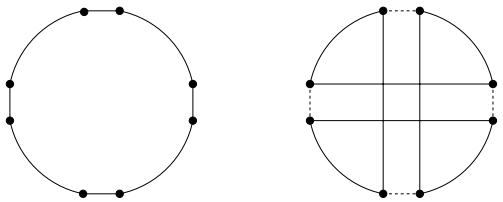


Figure 8: Double-bridge kick move

Table 1 compares two different versions of VQX. VGA1 and VGA2 represent the genetic algorithms using Voronoi quantized crossover with the genic distance assignments GD1 and GD2 (described in Section 3.4), respectively. In the table, the frequency of finding the optimal solutions (OB#), the best tour cost (Best), average tour cost (Avg), group standard deviation ($\sigma/\sqrt{t}$), average generation (Gen), and average running time (Time) over 100 ($= t$) runs are presented on att532, dsj1000, d2103, pcb3038, and fnl4461, problem instances from TSPLIB [38]. The parentheses after best and average tour costs represent the percentages above optima. VGA2 performed better than or equal to VGA1 for all instances except att532 in average cost. Distances between cities were computed in double precision mode and rounded to integer to remove uncertainties.

Table 2 compares the performance of VQX with other state-of-the-art crossovers. DGA, EGA, and NGA represent the genetic algorithms using distance preserving crossover [32, 26], edge assembly crossover [33], and natural crossover [20, 21], respectively. The results of DGA, EGA, and NGA in the table are quoted from [21] in which the same LK implementation as ours was used. VGA2 outperformed the others for all instances in average cost. For att532 and dsj1000, VGA2 consumed comparable running time to NGA. But their growth rates of time consumption with respect to the problem size were much lower than NGA. Thus, the speed of VGA2 for large problems pcb3038 and fnl4461 was much faster than the others. The overall results show that Voronoi quantized crossover is the most attractive among them. They also imply that GD1 and GD2 for the genic distance assignment are reasonable.

Table 3: Comparison of VGA with FCGA

Graph (opt)	Xover	OB #	Avg	Gen	Time (s)
eil101	FCGA	50	629.0	15	2
(629)	VGA2	50	629.0	6	1
lin318	FCGA	50	42029.0	49	60
(42029)	VGA2	50	42029.0	221	29
pcb442	FCGA	50	50778.0	39	233
(50778)	VGA2	50	50778.0	739	45
att532	FCGA	23	27691.3	66	304
(27686)	VGA2	47	27686.9	3199	159
rat575	FCGA	43	6773.2	55	500
(6773)	VGA2	19	6773.8	2720	53
u724	FCGA	41	41912.3	50	845
(41910)	VGA2	50	41910.0	3089	154

Table 3 compares the performance of VGA with FCGA [35]. FCGA stands for family competition genetic algorithm which is a combination of the family competition, near 2-opt and edge assembly crossover [33]. In the table, results over 50 runs are presented on eil101, lin318, pcb442, att532, rat575,and u724 from TSPLIB. These six instances are all those available for comparison in [35]. The results of FCGA in the table are quoted from [35]. The running time is the normalized value for Intel Pentium III 600 MHz. The average tour costs of VGA2 were better than or equal to FCGA for all instances except rat575. It is notable that the speed of VGA2 was much faster than FCGA. (In Table 2, EGA, the ancestor of FCGA, took 80 times more than VGA2 for the instance with 4461 cities.)

5 Conclusions

In this paper, we proposed a new crossover operator, named Voronoi quantized crossover (VQX), that utilizes the explicit genic distances. This allows us to exploit the interactions among genes explicitly. VQX has two main properties of convexity and diversity. These properties are believed to help improve the performance of genetic algorithms by encouraging the survival probability and the reproducibility of high-quality building blocks in the genetic process. The

experimental results supported this.

VQX may be applied to other combinatorial optimization problems than the traveling salesman problem. Of course, a measure for genic distances must be devised for each problem. Future studies include extending VQX to various problems.

Acknowledgments

The authors would like to thank Soonchul Jung for invaluable discussions on various ideas reported in this paper. This work was partly supported by SNU Statistical Research Center for Complex Systems and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

[1] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[2] T.N. Bui and B.R. Moon. Analyzing hyperplane synthesis in genetic algorithms using clustered schemata. In *Parallel Problem Solving from Nature*, pages 108–118. 1994.

[3] T.N. Bui and B.R. Moon. GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Transactions on CAD*, 17(3):193–204, 1998.

[4] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, Machine Learning*. Addison-Wesley, 1989.

[5] S.A. Kauffman. Adaptation on rugged fitness landscapes. In D.L. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison-Wesley, 1989.

[6] J.D. Bagley. *The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, 1967.

[7] G.R. Harik. *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*. PhD thesis, University of Michigan, 1997.

[8] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[9] H. Kargupta. *SEARCH, Polynomial Complexity, and the Fast Messy Genetic Algorithm*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[10] T.N. Bui and B.R. Moon. Hyperplane synthesis for genetic algorithms. In *5th International Conference on Genetic Algorithms*, pages 102–109, 1993.

[11] T.N. Bui and B.R. Moon. A new genetic approach for the traveling salesman problem. In *IEEE Conference on Evolutionary Computation*, pages 7–12, 1994.

[12] T.N. Bui and P. Eppley. A hybrid genetic algorithm for the maximum clique problem. In *6th International Conference on Genetic Algorithms*, pages 478–484, 1995.

[13] T.N. Bui and B.R. Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996.

[14] O.T. Sehitoglu and G. Üçoluk. A building block favoring reordering method for gene positions in genetic algorithms. In *Genetic and Evolutionary Computation Conference*, pages 571–575, 2001.

[15] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Foundations of Computer Science*, pages 577–591, 1994.

[16] J. Cohoon and D. Paris. Genetic placement. In *IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.

[17] C. Anderson, K. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.

[18] T.N. Bui and B.R. Moon. On multi-dimensional encoding/crossover. In *6th International Conference on Genetic Algorithms*, pages 49–56, 1995.

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

[20] S. Jung and B.R. Moon. The natural crossover for the 2D Euclidean TSP. In *Genetic and Evolutionary Computation Conference*, pages 1003–1010, 2000.

[21] S. Jung and B.R. Moon. Toward minimal restriction of genetic encoding and crossovers for the 2D Euclidean TSP. *IEEE Transactions on Evolutionary Computation* (conditionally accepted).

[22] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[23] D.S. Johnson and L.A. McGeoch. The traveling salesman problem: A case study in local optimization. In *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, 1997.

[24] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, 1994.

[25] P. Jog, J. Suh, and D. Gucht. The effect of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In *Third International Conference on Genetic Algorithms*, pages 110–115, 1989.

[26] P. Merz and B. Freisleben. Genetic local search for the TSP: New results. In *IEEE Conference on Evolutionary Computation*, pages 159–164, 1997.

[27] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[28] L. Davis. Applying adapting algorithms to epistatic domains. In *9th International Joint Conference on Artificial Intelligence*, pages 162–164, 1985.

[29] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Second International Conference on Genetic Algorithms*, pages 224–230, 1987.

[30] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *Third International Conference on Genetic Algorithms*, pages 133–140, 1989.

[31] A. Homaifar, S. Guan, and G. Liepins. A new approach on the traveling salesman problem by genetic algorithms. In *5th International Conference on Genetic Algorithms*, pages 460–466, 1993.

[32] B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. In *Parallel Problem Solving from Nature*, pages 890–900. 1996.

[33] Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In *7th International Conference on Genetic Algorithms*, pages 450–457, 1997.

[34] J. Watson, C. Ross, V. Eisele, J. Denton, J. Bins, C. Guerra, D. Whitley, and A. Howe. The traveling salesrep problem, edge assembly crossover, and 2-opt. In *Parallel Problem Solving from Nature*, pages 823–834. 1998.

[35] H.K. Tsai, J.M. Yang, and C.Y. Kao. A genetic algorithm for traveling salesman problems. In *Genetic and Evolutionary Computation Conference*, pages 687–693, 2001.

[36] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[37] G.F. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième Memoire: Recherches sur les parallélloèdres primitifs. *Journal für Reine und Angewandte Mathematik*, 134:198–287, 1908.

[38] TSPLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[39] R.B. Hollstien. *Artificial Genetic Adaptation in Computer Control Systems*. PhD thesis, University of Michigan, 1971.

[40] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[41] S.W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[42] J.L. Bentley. Experiments on traveling salesman heuristics. In *First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 91–99, 1990.

[43] M.L. Fredman, D.S. Johnson, L.A. McGeoch, and Ostheimer G. Data structures for traveling salesmen. *Journal of Algorithms*, 18:432–479, 1995.

[44] D. Cavicchio. *Adaptive Search Using Simulated Evolution*. PhD thesis, University of Michigan, 1970.

Robust Evolutionary Algorithms with Toroidal Search Space Conversion for Function Optimization

Hiroshi Someya
Information Science Center,
Nagasaki University,
1-14 Bunkyo Nagasaki 852-8521 Japan.
someya@net.nagasaki-u.ac.jp

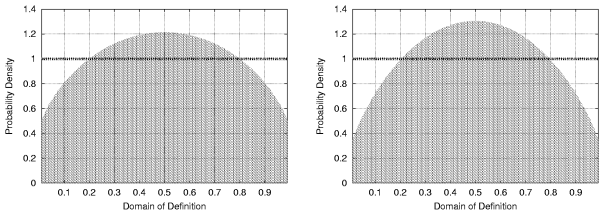
Masayuki Yamamura
Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku
Yokohama 226-8502 Japan.
my@dis.titech.ac.jp

Abstract

This paper presents a new method that improves robustness of Real-Coded Evolutionary Algorithms (RCEAs), such as Real-Coded Genetic Algorithms and Evolution Strategies, for function optimization. It is reported that most crossover (or recombination) operators for RCEAs has sampling bias that prevents to find the optimum near the boundary of search space. They like to search the center of search space much more than the other. Therefore, they will not work on functions that have their optima near the boundary of the search space. Although several methods have been proposed to reduce this sampling bias, they could not cancel the whole bias. In this paper, we propose a new method, *Toroidal Search Space Conversion* (TSC), to remove this sampling bias. TSC converts bounded search space into toroidal one with no parameters. Experimental results show that a RCEA with TSC has higher performance to find the optimum near the boundary of search space and it has improved robustness concerning the relative position of the optimum.

1 INTRODUCTION

Function optimization is one of the most important optimization problems. Several Real-Coded Evolutionary Algorithms (RCEAs) such as Real-Coded Genetic Algorithms and Evolution Strategies, which use the real number vector representation, have been proposed [3–6, 9, 10, 12, 15, 19] and they have shown higher performance than EAs using binary or gray representation [3, 5, 7]. In RCEAs, generally, initial individuals are placed in the search space uniformly. In this case, most crossover operators, such as BLX-α [5], Unimodal Normal Distribution Crossover (UNDX) [10], Center of Mass Crossover (CMX) [17], Simplex Crossover (SPX) [19], like to search the center of search space much more than the other [1, 4, 9, 18]. This bias is called "Sampling Bias" [4, 18]. Fig.1 explains the sampling biases of BLX-α and UNDX. The horizontal axis is domain of definition. The vertical axis is theoretical probability density of generating children when a crossover produces them from a pair of parents, chosen out of the population that is distributed in $0 \sim 1$ uniformly. When a crossover operator has such bias, it will not work on functions whose optima are near the boundary of the search space.



(a) BLX-α (b) UNDX

Figure 1: The sampling biases of BLX-α and UNDX

The sampling bias grows exponentially stronger as the dimension of search space. Therefore, in case objective function is high dimensional and its optima are in the corner of the search space, RCEAs like to be trapped at a local minimum located around the center of the search space. Recently, RCEAs have been applied to real-applications [11, 13, 16]. In real-applications, since we cannot know where are the optima, robust RCEAs considering the sampling bias are needed.

The purpose of this paper is to present a method that cancel the sampling bias to improve robustness of RCEAs. In the next section, we briefly review several major methods that reduce the sampling bias and discuss their features. We propose a new method, *Toroidal Search Space Conversion*, in section three

and reflect on the computational complexity in section four. Empirical verification is performed in section five. In the last section, we conclude this paper.

2 RELATED WORKS

2.1 Existing Methods and Their Features

Several methods, such as Boundary mutation [7], (UX, UNDX)+EMGG [9], boundary extension by mirroring (BEM) and boundary extension with extended selection (BES) [18], have been proposed. Boundary mutation produces individuals on boundary of search space. (UX, UNDX)+EMGG improves the sampling bias of UNDX using Uniform Crossover (UX) [3]. This method selects either UNDX or UX, they complement their searching region each other, as the crossover operator dynamically. BEM and BES extend the search space in order to move the relative position of the optimum toward the center of the search space. They allow individuals to be located outside the search space. The individuals are called "virtual individuals". The details of BEM are introduced in section 2.2. In BES, the number of the virtual individuals is limited by helper individual rate and no functional value of the virtual individuals is used. We can mention that these methods have the following three disadvantages.

(1) Dependence on Search Operator: In (UX, UNDX)+EMGG, UX and UNDX complement their searching region each other. However, when we use another crossover operator as the one of the search operators, we must invent or find the other one that has complementary characteristics to the first one.

(2) Parameter Tuning: All methods introduced in this section have at least one parameter, such as the mutation rate of boundary mutation, the initial probability of applying UNDX of (UX, UNDX)+EMGG, the extension rate of BEM and the helper individual rate of BES, to control how much the sampling bias is reduced. Although we cannot know the positions of the optima and the landscape of the search space, we must tune the parameters before search.

(3) Remaining the Sampling Bias: Although all methods shown in this section succeed in reducing the sampling bias, they cannot remove it. From the viewpoint of robustness, no sampling bias is desirable.

Next, we show BEM in detail because we believe it is the best method in the existing methods. It is independent on the search operator. The number of its parameters is only one. The effectiveness is relatively high.

2.2 BEM [18]

BEM aims to shift the optimum located in the corner toward the center. In BEM, individuals are allowed to be located beyond the boundary of search space. The functional value of individual i with real vector $\vec{X}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$ is calculated as follows:

$$
\begin{aligned}
f(\vec{X}^{(i)}) &= f(\vec{Y}^{(i)}), \qquad (1)\\
\vec{Y}^{(i)} &= (y_1^{(i)}, \ldots, y_n^{(i)}),\\
y_j^{(i)} &= \begin{cases} 2\ \min_j - x_j^{(i)} & : \text{ if } x_j < \min_j \\ 2\ \max_j - x_j^{(i)} & : \text{ if } x_j > \max_j \\ x_j^{(i)} & : \text{ otherwise,} \end{cases}
\end{aligned}
$$

where, $\min_j$ and $\max_j$ are the lower and upper limits of parameter range on the j-th dimension of the original search space respectively. BEM has one parameter, r_e $(0 < r_e < 1)$, that controls how much search space is extended. The parameter range of the extended search space is $l_j(1 + r_e)$ when that of the original one is l_j. The initial individuals are placed in the original search space uniformly.

3 TOROIDAL SEARCH SPACE CONVERSION (TSC)

TSC converts search space with boundary into toroidal one. This conversion is performed as follows:

step1 Extend the search space to the extended search space like BEM with r_e=1.0, (see section 2.2)

step2 Connect each e-$\max_j$ of the extended search space to corresponding e-$\min_j$.

where, e-$\min_j$ and e-$\max_j$ are the lower and upper limits of parameter range on the j-th dimension of the extended search space respectively. An example of the converted search space is shown in Fig.2. The converted search space becomes torus. In this converted search space, the crossover operation is performed as the following pseudo-codes (like *C++*):

```
choose k required parents;
for (int i=1; i<k; i++){
  make pow(2, n)-1 clones of parent_i;
  // ---- n is the dimension
  select the clone whose distance from
    parent_0 is the shortest out of the
    clones and parent_i;
}
do crossover using parent_0 and the
  k-1 selected clones;
```

Fig.3 shows an example of a crossover in a converted search space. First, three `clones` (`clone_1`,

clone_1*, clone_1**) at the corresponding points on the virtual search space are copied from parent_1. Next, clone_1 is allowed to join the crossover operation because its distance from parent_0 is shortest. Thus, the crossover operation, using UNDX as the crossover operator, searches in the gray region.

For implementation on a computer program, the procedures in the above for are described as follows:

```
clone_i = parent_i;
// ---- copy the parent_i vector to clone
for (int j=0; j<n; j++){
  const double distance
    = clone_i[j] - parent_0[j];
  if (fabs(distance) > l){
  // ---- l is the half width of the
  //        extended search space
    if (distance >= 0){ clone_i[j] -= 2l; }
    else              { clone_i[j] += 2l; }
  }
}
```

Although the volume of the search space grows exponentially, the increase of the computational cost for this crossover is only linear, $O(k \times n)$. Since the converted search space is torus, a generated individual i, $\vec{X}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$, is modified as follows:

$$\begin{aligned} \vec{X}^{(i)} &= \vec{Z}^{(i)}, \\ \vec{Z}^{(i)} &= (z_1^{(i)}, \ldots, z_n^{(i)}), \\ z_j^{(i)} &= \begin{cases} x_j^{(i)} + 2l & : \text{ if } x_j < \text{e-min}_j \\ x_j^{(i)} - 2l & : \text{ if } x_j > \text{e-max}_j \\ x_j^{(i)} & : \text{ otherwise.} \end{cases} \end{aligned} \tag{2}$$

For example, in Fig.2, when A and B are generated by a crossover operation, they are modified as A' and B' respectively. Using this modification, when the distance between parents is far, crossover does not generate children in the center of the search space, but does them near the boundary close to the parents (in the gray region in Fig.3). In TSC, initial individuals are placed in the extended search space uniformly. Accordingly, by this proposed method, any position on this search space become equivalent to any others.

TSC clears the three disadvantages of the existing methods. Since TSC is a conversion method, it is independent on any search operator. TSC has no parameter. The converted search space has no sampling bias when the initial individuals are placed in the extended search space uniformly because it is torus.

TSC has one more significant feature. The converted search space maintains global continuity of landscape.

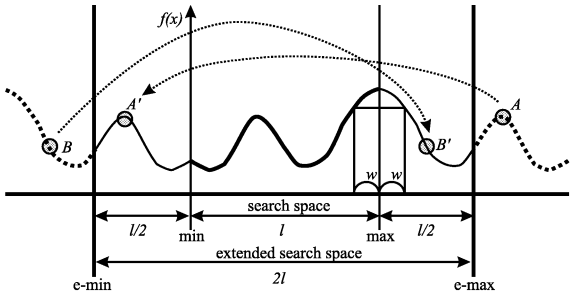


Figure 2: An example of 1-dimensional converted search space by TSC

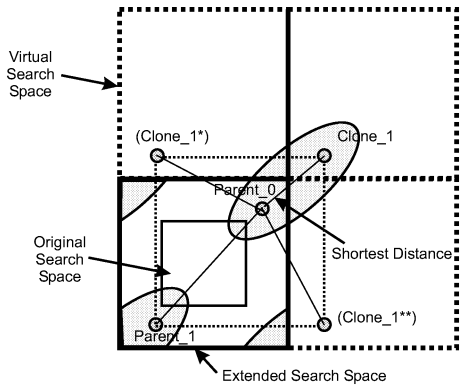


Figure 3: An example of crossover procedure, using UNDX as the crossover operator, on a 2-dimensional converted search space by TSC.

The "global continuity of landscape" means that individuals around an individual have approximate equivalent functional value. In [16], the authors use a method that connects $\min_j$ and $\max_j$ when the coded vector represents an angle. In this method, children are produced only in the supplementary angle region because -180 degrees correspond to 180 degrees. When we apply this method to a search space that does not have such characteristic, the global continuity of landscape should be lost because $f(x_1, \ldots, \min_j, \ldots, x_n)$ will be different from $f(x_1, \ldots, \max_j, \ldots, x_n)$. Since EAs assume that search space has the global continuity of landscape [8], the global continuity of landscape should be maintained. In TSC, it is satisfied because e-$\min_j$ corresponds to e-$\max_j$, even if the original search space does not have the above characteristic.

4 COMPUTATIONAL COMPLEXITY

Let discuss the number of samplings required to find the optimum in a n-dimensional search space whose volume is D, as shown in Fig.4. First, we discuss an EA without any selection mechanism. Then, we consider an EA equipped with a selection mechanism.

Table 1: The test functions

function	equation (n specifies the dimension)	mul.*1	disc.*2	domain	d_i
Sphere	$\sum_{i=1}^{n} x_i^2$	no	no	[-5.12+d_i, 5.12+d_i]	0.0, 1.5, 3.0, 4.5
Step	$\sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$	no	strong	[-5.12+d_i, 5.12+d_i]	0.0, 1.5, 3.0, 4.5
Schwefel	$418.9828873n + \sum_{i=1}^{n} x_i \sin\sqrt{\lvert x_i \rvert}$	low	no	[-512, 512]	-
Rastrigin	$10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	high	no	[-5.12+d_i, 5.12+d_i]	0.0, 1.5, 3.0, 4.5
Griewangk	$\frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	high	no	[-512+d_i, 512+d_i]	0, 150, 300, 450

*1: multi-modality, *2: discontinuity

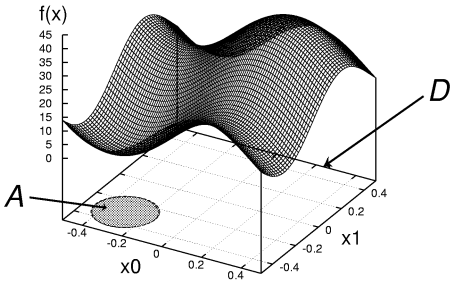


Figure 4: n-dimensional objective function, in which D is its volume and A is the volume of the region as the optimum.

4.1 Without Selection

In case BLX-α is used as the search operator, the probability density curve of generating children, shown in Fig.1 (a), is expressed as follows (the details of BLX-α and $g(x)$ are shown in Appendix A and B respectively):

$$g(x) = \frac{2\{\ln\frac{3}{2} + (x-1)\ln(1-x) - x\ln x\}}{2\ln\frac{3}{2} + 1}. \quad (3)$$

Therefore, when n is 1, the number of samplings required to find the optimum that is located at the corner of the search space is $\frac{1}{g(0)} = \frac{1}{g(1)} \simeq 2.233$ times as many as in case Uniform Random Search (URS) [20], which searches in the domain uniformly, is used. When n is 10 or 20, $\frac{1}{g(0)^n}$ is about 3,000 or 9,500,000, respectively. In case another crossover operator whose sampling bias is stronger than BLX-α, such as UNDX, is used, more samplings are required. The probability to find the optimum, P, when URS is used as the search operator is expressed as follows [20]:

$$P = 1 - \left(1 - \frac{A}{D}\right)^m, \quad (4)$$

where m is the number of samplings and A is the volume of the region as the optimum. When the search space is converted by TSC, since there is no sampling bias even if the search operator is BLX-α, the search works like URS. In this case the probability to find the optimum is equivalent to URS as $\frac{2^n A}{2^n D} = \frac{A}{D}$.

4.2 With Selection

When we consider selection mechanism, the complexity of landscape is important. Unimodal function is often converted into multimodal one by TSC. Generally, optimization of multimodal function is more difficult than that of unimodal one. Moreover TSC converts multimodal function into more complex multimodal one in which the number of local minima is exponentially larger. It has not been cleared that the relation between complexity of landscape and the difficulty of optimization for EAs. However, it has been known that big hill including local minima influences the effectiveness of EAs.

5 EXPERIMENTS

In order to confirm the robustness of EAs in converted search space by TSC, we perform experiments.

5.1 Test Functions

How test functions should be selected has been mentioned in [2]. The five functions in Table 1 are selected under the recommendations. The optimum of Schwefel function, $f(-420.968746, \ldots, -420.968746) = 0$, and those of the others, $f(0, \ldots, 0) = 0$, are located in the corner and at the center of the search space, respectively. To achieve the purpose of these experiments, the relative positions of the optima in their search space are moved by d_i except that of Schwefel function[1], as shown in Fig.5.

5.2 Experimental Conditions

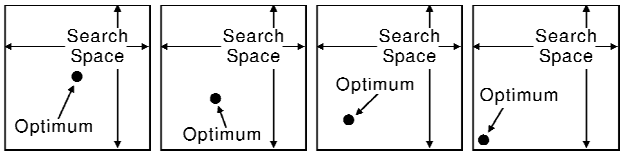
We select UNDX+MGG [10,14] as the performed EA. It has been reported that UNDX has strong sampling

[1]In Schwefel function, when the domain is changed by d_i, the optimum will be changed.

Table 2: The experimental results (#OPT)

		NoExt				BEM				BEMe				TSC			
function	n	00	15	30	45	00	15	30	45	00	15	30	45	00	15	30	45
Sphere	50	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	**99**	100	100	100	**98**	100	100	100	100	100
	150	100	100	100	100	100	100	**98**	100	100	100	**96**	100	100	100	100	100
Step	30	100	96	78	**62**	100	96	**80**	93	100	95	**86**	97	**92**	96	99	100
	40	99	56	27	**11**	95	57	**35**	56	96	64	**28**	54	62	**52**	80	99
	50	81	17	**7**	10	75	19	**8**	12	88	17	**7**	12	**6**	16	35	83
Schwefel	5			87				100				99				100	
	10			8				57				56				100	
	15			0				1				1				98	
Rastrigin	4	100	99	94	**26**	100	100	99	**86**	100	97	98	**87**	100	100	100	100
	6	100	67	20	**1**	99	77	57	**13**	98	84	80	**8**	96	79	**44**	100
	8	84	27	3	**0**	91	26	10	**0**	79	34	22	**1**	79	35	**5**	97
Griewangk	30	80	68	69	**65**	70	71	74	**65**	68	76	72	**64**	65	66	**63**	73
	40	71	74	**63**	70	**65**	67	75	73	**66**	67	69	76	66	**60**	66	64
	50	**69**	73	77	70	76	74	**70**	73	**61**	69	65	68	63	**58**	60	62

$*$ $00, 15, 30, 45$ under the method names specify d_i. For example, 15 means $d_i = 1.5$ or $d_i = 150$.



$d_i = 0$ $d_i = 1.5$ or 150 $d_i = 3.0$ or 300 $d_i = 4.5$ or 450

Figure 5: The relative positions, caused by d_i, of the optimum in a search space

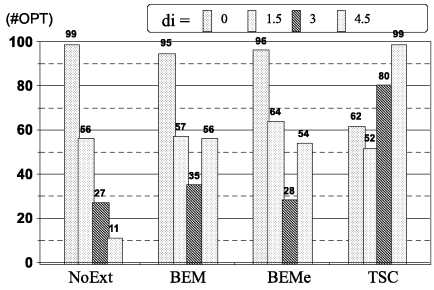
bias as shown in Fig.1 (b) and UNDX+MGG does not work well in a search space whose optimum is in the corner, such as Schwefel function [9]. The details of this EA are shown in Appendix A and C. No mutation is used for focusing on the sampling bias caused by crossover. The population size is set to be 30 for unimodal functions but 100 for multimodal ones. Fifty children are produced in each generation.

TSC is compared to "No Extension method (NoExt)", BEM and BEMe. NoExt means that the EA is performed in the original search space. BEMe is introduced to be fair in our comparison. BEM and BEMe are the same except that BEMe places initial individuals like TSC. TSC places initial individuals in the extended search space, but BEM does them in the original one. The r_e of BEM and BEMe are set to be 0.25 because the value has been used in [19]. In all experiments except TSC, when an individual is generated outside their search space, the crossover retries to generate another inside. Each experiment is performed 100 trials. Each run continues until the optimum is found or the number of evaluation reaches a constant that was set to be enough large number determined in pilot study. The performance measure is the numbers of runs in which the method succeeded in finding the global optimum (#OPT). The robustness of each method is evaluated through the lowest performance in all cases of d_i.

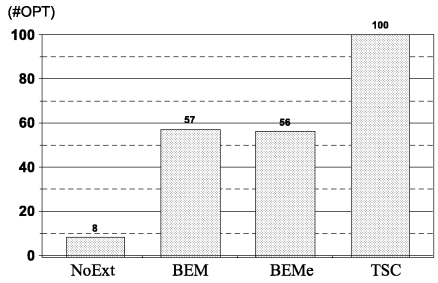
5.3 Results and Discussion

The experimental results are shown in Table 2. Several results that explain the features of the methods obviously are shown in Fig. 6.

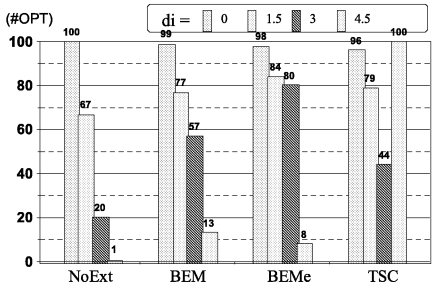
In Sphere function, the all #OPTs are approximately 100. We believe that the optimization in the converted search space by TSC has not become more difficult, because it has had no local minimum although it has become multimodal. Fig.6 (a) and (c) show the robustness of the EA performed in the converted search space by TSC. In the original search space (NoExt), the performance when $d_i = 4.5$ is terrible. We believe that this is caused by the sampling bias. You might consider why the #OPTs are different among the d_i despite no sampling bias when TSC is used. Note, the landscapes are different among the d_i although the equations are the same. In Schwefel function, which has the optimum in the corner of the search space, we can confirm that the performance of the EA is extremely improved by TSC. In Griewangk function, all methods show the robustness as shown in Fig.6 (d). From Table 1, the characteristic of this function seems to be the same as that of Rastrigin function. However, the landscape of this function is similar to that of Sphere function on the broad level, as shown in Fig.7. We believe that this robustness is caused by this similarity. In 50-dimensional Step function and 8-dimensional Rastrigin function, the difference of the effectiveness among the methods is little. Hence, we



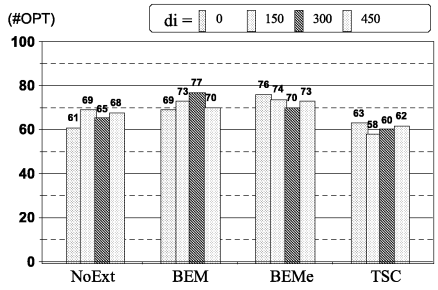
(a) 40-dimensional Step function



(b) 10-dimensional Schwefel function



(c) 6-dimensional Rastrigin function



(d) 50-dimensional Griewangk function

Figure 6: The experimental results (#OPT) that explain the features of the methods obviously

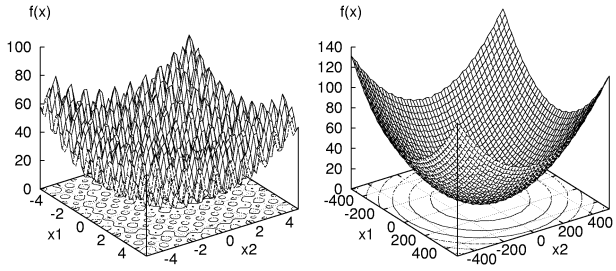
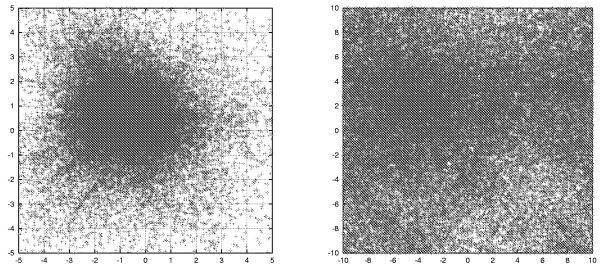


Figure 7: Rastrigin function (left) and Griewangk function (right)



NoExt TSC

Figure 8: The distribution of the overall generated individuals in the function, $f(X) = 1$

studied the average, the worst and the variance of the runs. The statistics have shown that TSC works better than the others.

The stability of convergence speed when TSC is used is the lowest than that when the others are used. The EA performed in the converted search space by TSC can find the optimum located in the corner of the search space rapidly. However, when the optimum is located at the other positions, the convergence velocity is slower. It is a disadvantageous feature of TSC.

5.4 Confirmation of No Sampling Bias

We perform experimental confirmation of no sampling bias in the converted search space by TSC. $f(\vec{X}) = 1$ whose dimension is two is used as the objective function. The domain of definition is $[-5.0, 5.0]$. The number of evaluation is 5.0×10^4. The population size is set to be 100. The other conditions are the same as the previous experiments. Since MGG performs random sampling when all individuals have the same fitness value, the all region should be searched equally if there is no sampling bias. We plot the distribution of the overall individuals generated in a run. Fig.8 shows the results of NoExt and TSC. Although near boundary in the left figure is hardly searched, all region in the right figure are searched equally. We can confirm that there is no sampling bias.

6 CONCLUSIONS

This paper proposed a new method, *Toroidal Search Space Conversion* (TSC), which converts search space

with boundary into toroidal one, to improve the robustness of RCEAs. Experimental results showed that the effectiveness of TSC is greater than those of the other methods. TSC has following three advantages: 1. TSC can be applied widely because it is independent on search operator., 2. It is easy to apply TSC because it has no parameter., 3. There is no sampling bias in the converted search space by TSC. On the other hand, TSC has one disadvantage. The landscape of the converted search space by TSC is often more complex than that of the original search space. The variance of the convergence velocity is also caused by this complexity. To cope with this disadvantageous feature is future work.

References

[1] P. J. Angeline. Using Selection to Improve Particle Swarm Optimization. In *Proc. of the ICEC'98*, pages 84–89, 1998.

[2] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[3] L. Davis. *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1990.

[4] L. J. Eshelman, K. E. Mathias, and J. D. Schaffer. Crossover Operator Biases: Exploiting the Population Distribution. In *Proc. of the 7th ICGA*, pages 354–361, 1997.

[5] L. J. Eshelman and J. D. Schaffer. *Foundations of Genetic Algorithms 2*, chapter Real-Coded Genetic Algorithms and Interval-Schemata, pages 187–202. Morgan Kaufman, 1993.

[6] H. Kita, I. Ono, and S. Kobayashi. Multi parental Extension of the Unimodal Normal Distribution Crossover for Real-Coded Genetic Algorithms. In *Proc. of the CEC'99*, pages 1581–1587, July 1999.

[7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Program*. Springer, third edition, 1996.

[8] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[9] I. Ono, H. Kita, and S. Kobayashi. A Robust Real-coded Genetic Algorithm using Unimodal Normal Distribution Crossover Augmented by Uniform Crossover : Effects of self-Adaptation of Crossover Probabilities. In *Proc. of the GECCO-99*, pages 496–503, July 1999.

[10] I. Ono and S. Kobayashi. A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover. In *Proc. of the 7th ICGA*, pages 246–253, 1997.

[11] I. Ono, Y. Tatsuzawa, S. Kobayashi, and K. Yoshida. Designing Lens Systems Taking Account of Glass Selection by Real-coded Genetic Algorithms. In *Proceedings of 1999 IEEE International Conference on Systems, Man and Cybernetics*, pages III–592–597, 1999.

[12] I. Ono, M. Yamamura, and S. Kobayashi. A Genetic Algorithm with Characteristic Preservation for Function Optimization. In *Proceedings of IIZUKA'96*, pages 511–514, 1996.

[13] S-J. Park and M. Yamamura. An Approach to Structural Alignment with Genetic Algorithm. In *Proceedings of the Second International Conference on Bioinformatics of Genome Regulation and Structure*, pages 201–203, 2000.

[14] H. Satoh, M. Yamamura, and S. Kobayashi. Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation. In *Proceedings of IIZUKA'96*, pages 494–497, 1996.

[15] H. Someya and M. Yamamura. Where should Children be Generated by Crossover Operator on Function Optimization ? In *Proc. of the GECCO-2000*, page 382, July 2000.

[16] O. Tomobe, I. Ono, and S. Kobayashi. Experimental Study on Determination of Protein three dimensional Structure using Genetic Algorithm (in Japanese). In *Proceedings of 25th SICE Symposium on Intelligent Systems*, pages 35–40. The Society of Instrument and Control Engineers, 1998.

[17] S. Tsutsui. Multi-parent Recombination in Genetic Algorithms with Search Space Boundary Extension by Mirroring. In *Proc. of the PPSN V*, pages 428–437, 1998.

[18] S. Tsutsui and D. E. Goldberg. Search Space Boundary Extension Method in Real-Coded Genetic Algorithms. *Information Sciences*, 133(3-4):229–247, 2001.

[19] S. Tsutsui, M. Yamamura, and T. Higuchi. Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms. In *Proc. of the GECCO-99*, pages 657–664, July 1999.

[20] A. A. Zhigljavsky. *Theory of Global Random Search*, volume 65 of *Mathematics and Its Applications*. Kluwer Academic Publishers, 1991.

APPENDIX

A BLX-α [5] and UNDX [10]

BLX-α produces a child in the gray region in Fig.9 (left) randomly. The child vector, $\vec{C}$, which is encoded by real number vector, is determined as follows:

$$\begin{aligned} \vec{C} &= \{c_1, \ldots, c_n\} , \\ c_i &= u(\min(p_{1i}, p_{2i}) - \alpha d_i,\ \max(p_{1i}, p_{2i}) + \alpha d_i) , \end{aligned}$$

where $\vec{P}_1 = \{p_{11}, \ldots, p_{1n}\}$ and $\vec{P}_2 = \{p_{21}, \ldots, p_{2n}\}$ are parent vectors of Parent1 and Parent2 respectively. $d_i = |p_{1i} - p_{2i}|$. n is the dimension of the objective function. $u(x, y)$ is the uniform random number selected from $[x, y]$.

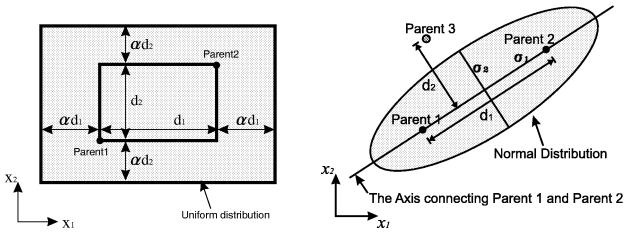


Figure 9: BLX-α (left) and UNDX (right)

UNDX generates two children around their parents using the normal distribution whose standard deviation is determined by the third parent, Parent3, as shown in Fig.9 (right). The children vectors, $\vec{C}_1$ and $\vec{C}_2$, are determined as follows:

$$\begin{aligned} \vec{C}_1 &= \vec{m} + z_1\vec{e}_1 + \sum_{k=2}^{n} z_k\vec{e}_k \ , \\ \vec{C}_2 &= \vec{m} - z_1\vec{e}_1 - \sum_{k=2}^{n} z_k\vec{e}_k \ , \end{aligned}$$

where $\vec{m} = (\vec{P}_1 + \vec{P}_2)/2$. $\vec{e}_1 = (\vec{P}_2 - \vec{P}_1)/|\vec{P}_2 - \vec{P}_1|$, $\vec{e}_k (k = 2, \ldots, n)$ are the orthogonal unit vectors. $z_1 \sim N(0, \sigma_1^2)$ and $z_k \sim N(0, \sigma_2^2)(k = 2, \ldots, n)$ are normally distributed random numbers, where $\sigma_1 = \alpha d_1$ and $\sigma_2 = \beta d_2/\sqrt{n}$. d_1 is the distance between Parent1 and Parent2. d_2 is the distance of the Parent3 from the line connecting Parent1 and Parent2. α and β are constants.

B EQUATION (3)

B.1 Variables

In this section, we use the following variables:

y, z	the positions of parents $(0 < y < z < 1)$
w	the width in which BLX-α produces children
c	the center of parents
$n(y, z)$	the probability of generating children in one crossover operation

α is set to be 0.5, which is recommended value.

B.2 Equation $g(x)$

After the definition of BLX-α, a child is produced in the range of x that satisfies the following inequality.

$$c - \frac{w}{2} < x < c + \frac{w}{2} \ .$$

Substitute $w = (z - y)/\alpha = 2(z - y)$ and $c = (z + y)/2$ into the above inequality,

$$-1 < \frac{2x - (z + y)}{2(z - y)} < 1 \ .$$

Therefore,

$$\begin{aligned} 3y &< 2x + z \ , \\ y &< 3z - 2x \ . \end{aligned}$$

Since the domain of x is $[0.0, 1.0]$, $g(x) = k\{g_A(x) + g_B(x) + g_C(x)\}$, as follows:

$$\begin{aligned} g_A(x) &= \int_x^{\frac{1}{3}+\frac{2}{3}x} \int_{3y-2x}^{1} n(y, z)\ dzdy \ , \\ g_B(x) &= \int_0^x \int_x^1 n(y, z)\ dzdy \ , \\ g_C(x) &= \int_0^x \int_{\frac{1}{3}y+\frac{2}{3}x}^{x} n(y, z)\ dzdy \ , \end{aligned}$$

where $n(y, z) = \frac{1}{w} = \frac{1}{2(z-y)}$. Integrate the above,

$$\begin{aligned} g(x) &= k\left\{ \frac{(1 - x)(\ln 3 - \ln 2)}{2} \right. \\ &\quad + \frac{(x - 1)\ln(1 - x) - x \ln x}{2} \\ &\quad \left. + \frac{x(\ln 3 - \ln 2)}{2} \right\} \\ &= \frac{k}{2}\{\ln 3 - \ln 2 + (x - 1)\ln(1 - x) - x \ln x\} \ . \end{aligned}$$

In order to satisfy $\int_0^1 g(x)\ dx = 1$, $k = \frac{4}{2(\ln 3 - \ln 2) + 1}$. Hence,

$$g(x) = \frac{2\{\ln \frac{3}{2} + (x - 1)\ln(1 - x) - x \ln x\}}{2 \ln \frac{3}{2} + 1} \ .$$

C MGG [14]

MGG is a generation-alternation model. It is described as follows:

step1 Generate an initial population randomly.
step2 Choose a pair of individuals as parents from the population randomly.
step3 Generate a certain number of children by a crossover.
step4 Select the best individual out of the family, the parents and the children.
step5 Choose an individual except the best, selected at *step4*, out of the family randomly according to fitness-based (or ranked-based) wheel selection.
step6 replace the two individuals, selected at *step4* and *step5*, to the parents.
step7 Iterate *step2* $\sim$ *step6* until certain condition is satisfied.

Jumping Genes-Mutators Can Rise Efficacy of Evolutionary Search

Alexander V. Spirov

The Sechenov Institute of Evolutionary Physiology and Biochemistry, 44 Thorez Ave., St. Petersburg, 194223, Russia

and

Dept. of Applied Mathematics and Statistics, The State University of New York at Stony Brook, Stony Brook NY 11794-3600, USA

Email: spirov@kruppel.ams.sunysb.edu

fone: 631-632-8370

fax: 631-632-8490

Alexander B. Kazansky

The Sechenov Institute of Evolutionary Physiology and Biochemistry, 44 Thorez Ave., St. Petersburg, 194223, Russia

Email: kazansky@iephb.nw.ru

fone/fax: +7 (812) 552 3219

Abstract

Genetic Algorithms (GA) and Genetic Programming were inspired by ideas from evolutionary biology. However modern Evolutionary Computation (EC) only in outline reminds the strategies of biological evolution. The application of other algorithms and biological ideas may substantially improve the performance of this area of computer science. Namely, the selfish (or parasitic) mobile genetic elements - transposons are good candidates for this breakthrough. These genomic parasites live on a substratum of genomes of whole biological communities. Many biologists assume that processes in the world of transposons are the main source of evolution creativity. They thought to act as wise higher-level mutators for their hosts. In this communication we propose a strategy of construction of a new approach exploiting the most essential aspects of co-evolution of the hosts-chromosomes and their genetic parasites. We named this strategy as the Two-level Evolving Worlds. The key feature of the approach is usage of artificial transposons. We apply it to one of known benchmark problems - the John Muir ant's trail test. We found that our enhancement of GA technique by the artificial transposons obviously increase the efficacy of searching of the ant's navigation algorithm. We investigate in details the way of the transposons action as intelligent mutators of host-chromosomes.

1 INTRODUCTION

Many areas of evolutionary computation, especially genetic algorithms (GA), and genetic programming (GP), are inspired by achievements in genetics and evolutionary biology. However modern evolutionary biology has since advanced considerably, revealing that genes are not simply parameter settings, but components of a complex biochemical machine (Cf. Luke et al., 1999; Lee and Antonsson, 2001; Lones and Tyrrell, 2001).

On the other hand, many branches of modern evolutionary computation research are aimed at evolution of mechanisms (neural networks, decision trees, cellular automata, L-systems, finite state automata). For these domains, recent genomic achievements seems more appropriate as an inspirational model then classic set of Darwinian algorithms.

There is a feeling that the field of EC is getting more inspired with the latest achievements in biology, trying to make the evolutionary algorithms more effective. Such techniques as transposition, host-parasite interaction, gene-regulatory networks and some others have yet been applied to EC.

•*Host-parasite methods*: These methods are based on the co-evolution of two different populations, one of them acting as “parasite”, and the other acting as “host”. The parasites usually encode a version the problem domain, and the hosts the solution to the problem (Hillis, 1990; Potter and De Jong, 1994; 1995; De Jong and Potter, 1995; Olsson, 1996; 2001).

•*Transposition operators* (*“bacterial” algorithms):* The basic idea of these approaches is to make intra-chromosome crossovers, that is, crossover of a

chromosome with another part of itself, or else asymmetric crossover, in which a donor chromosome transfers part of its genetic material to an acceptor chromosome (Harvey, 1996, Nawa et al., 1996; Simoes and Costa, 2001). In some cases, these operators seem to be better than classical genetic algorithms for combinatorial optimization problems.

•*Gene-regulatory networks approach*: Luke et alls (1999) use a method similar to genetic regulatory networks to evolve finite state automata that represent a language grammar. It is appropriate also to mention here the Burke et alls (1998) project, as well as "enzyme genetic programming" (Lones and Tyrrell, 2001).

•*Evolution based on the selfish elements*: Corno et alls (1998) implemented the *Selfish Genetic Algorithm* inspired by Dawkins concept of the *selfish gene*. The algorithm evolves a *Virtual Population*, in which alleles compete for appearance in their respective locus in the genotype.

So far, it has not been found in the literature a technique that is general enough to be applied to a wide range of problems, and that, in some cases, is able to yield as good or better results than evolutionary algorithms

This stimulates us to search for prospective mechanisms that simulate the creative, heuristic and self-organizing character of (biological) evolution (Spirov, 1996a; 1996b; Spirov and Samsonova, 1997; Spirov and Kadyrov, 1998; Spirov et al., 1998; Spirov and Kazansky, 1999). The mobile selfish genetic elements (synonymous or related terms are jumping genes, transposons, retroviruses) are good candidates for this breakthrough (Makalowski, 1995). Many biologists speculate that processes in the world of transposons, living on a substratum of genomes of the whole biological communities, are the main source of macroevolution creativity (Doolittle and Sapienza, 1980; Orgel and Crick, 1980; Brosius, 1991).

In this connection, special interest is attracted by well-known examples of both competitive and cooperative strategies in populations of transposons.

In this communication we propose a strategy of construction of a new approach exploiting the most essential aspects of co-evolution of the hosts-chromosomes with their genetic parasites. We named this strategy as the *Two-level Evolving Worlds*. The key feature of the approach is usage of *artificial transposons*. We treat transposons as high-level and intelligent mutators. In the next part we give the definition of the strategy. To demonstrate the efficacy of a new approach we apply it to one of known benchmark problems - the John Muir ant's trail test (Jefferson et al. 1992; Koza, 1992).

1.1 THE TWO-LEVEL EVOLVING WORLD

Parasites and parasite ensembles always accompany biological evolution. Tom Ray simulated this process in his *Tierra* (Ray, 1991).

A special kind of parasites is genomic parasites living in the host genome. Known biological proverb says that "*the viruses in all of us - the viruses that make us*".

In the course of evolutionary time, parasites form "community" of their own. They populate the united genomic space of many hosts. We shall name these parasites as *InfoParasites (IP)*, and the "community of the parasites" as *IP world*.

There are examples of evolvable virtual worlds such as Swarm, Creatures, Network Tierra (Daniels, 1999; Cliff and Grand, 1999; Ray, 2001). In the course of evolution the worlds of that type can split over IP and host co-evolving worlds, i.e. they can become the two-leveled. It is the question of time and such worlds' complexity. In less complex virtual worlds similar splitting could be realized "by hand", as in the case of developing world of computer viruses.

1.1.1 Strategy of Development of The Two-level Worlds

We assume that the simplest realization of the two-layer evolving worlds would be as follows:

the hosts-world is GA-like system (standard GA in the simplest case). The manifold of hosts' chromosomes-strings is the environment for IPs. In the simplest case these GAs don't have any mutation operators of their own;

the InfoParasites are the LISP-like programs, manipulating with the hosts' strings. (For our applications these programs must include the SEARCH function performing the search of patterns in the host strings). IPs live in hosts, they are transmitted vertically (when host reproduces) and horizontally (from one host to another, as infection or computer virus);

genotypes of parasite and host are encoded by the same text, i.e. the same string of symbols is interpreted in two different languages, the host's and the parasite's one;

"bad" (too harmful) parasites are eliminated together with their hosts, "good" parasites minimize their harmfulness (for example, by exploiting unessential parts of host's chromosomes).

1.1.2 Intelligent Mutators

IPs acts as intelligent and sophisticated mutators. They can generate arbitrary procedures of manipulations with

hosts' chromosomes. In general, these operators can be the unitary, binary or plural ones. Each host has got the mutators of its own. In the simplest case IPs are the only source of the host's mutations.

If IP founds hopeful mutation strategy, then both host and parasite will get chance for reproduction, the parasite rides on a new turn of evolution on the transformed host. Virtually we have co-evolution of hosts and their intelligent mutators-parasites.

1.2 THE ARTIFICIAL ANT PROBLEM

The artificial ant problem is the simulation of an ant navigation aimed at passing through the labeled trail placed in a grid world (Jefferson et al. 1992; Koza, 1992). The trail was nicknamed as "The John Muir Trail" in the UCLA experiment (Jefferson et al., 1991). Each labeled cell is numbered sequentially, from the 1st which is settled directly next to the starting cell, through to the last cell. The ant's task is to pass through the labeled cells one by one (the more the better) for the limited time period. The ants are simple finite-state automata or an artificial neural network, which can move along the grid world and test their immediate surroundings. The trail starts off quite easy to follow, and gradually gets more difficult, as the turns become more unpredictable and gaps appear (See Fig.2). Therefore, the successful ant's program must be quite sophisticated. The problem has been repeatedly used as a benchmark problem (For references See Langdon and Poli, 1998).

2 METHODS AND APPROACH

While the ant test was implemented at least in two different C++ libraries (Zongker and Punch, 1995), we gave preference to the Peter Brennan's version (Brennan, 1994). This "ANT program" was designed in such a way that to isolate, as far as possible, the components of the genetic algorithm from the trail-following experiment and the ant representation. Brennan's ants are finite state automata.

2.1 TECHNIQUE OF MOBILE GENETIC ELEMENTS - TRANSPOSONS

Mobile Genetic Elements (MGEs) - transposons are akin to computer viruses. They are the autonomous programs, which are transmissible horizontally (viz., from one site to another one on the same or another chromosome) or vertically (from the ancestor to the descendants in the reproduction process). These autonomous parasitic programs cooperate with the host genetic programs, thus realizing process of self-replication - the only aim, which can be associated with that activity. We developed some new operators which are the computer program procedures, performing processes of replication, mutation and invasion of MGEs into specific sites on chromosomes, as well as interactions of MGE with the chromosome (interrelations of parasite - host type).

It is appropriate here to make some notes, concerning the terminology. MGE technique comprises the procedures for initialization of mobile genetic elements and procedures for operating with these elements. Hereinafter in this section mobile elements will be referred to as "viruses", whereas the procedures, operating with them will be termed as "MGE operators". There are only two types of operators. The one-place operator is an analogue of point mutation and the two-place (binary) operator realizing the procedure of transmission of virus from one chromosome (host) to another chromosome (another host).

2.1.1 Viruses

Let us recall that the ant binary string - chromosome is coding a state transition table of finite state automation. Altogether there are 32 finite states of automation, ranging from STATE#0 up to STATE#31. All operators start reading and interpreting the table beginning from the STATE#0. For example, STATE #0 determines one of the four actions or instructions (FWD - "forward", RGT - "to the right", LFT - "to the left" or NOP - "do-nothing") and the number of the next state, depending on binary input value (0 or 1). This finite state automation can be represented as a state transition diagram and interpreted as a decision tree but, as far as references to already passed by states are permissible, that tree can have loops.

Henceforward we will refer to these state number sequences, which ant can pass through moving along the branches of the tree and according sequences of instructions (routines), which it will perform, as "patterns". In other words, pattern is concrete sequence of states, which an ant can come through and sequence of instructions, which an ant can perform, when it passes from state to state. Concrete example of patterns are given on the Fig. 1. Hereinafter, the abbreviations of instructions in the pattern will be referred to as elements of pattern.

We use this concrete definition of our *virus* (mobile genetic element - transposon). Virus is the pattern, having the following properties:

the pattern should include elements which number lie in the range between minimum and maximum values;

the pattern should not contain NOP elements and internal circles;

the pattern should be finished up with a reference to the initial state. The transitions cycle will be executed until only white squares remain ahead of the ant.

2.1.2 MGE - operators

MGE - operators scan the predetermined quota of chromosomes in population. Successively decoding chromosome record, this operator is seeking for procedure sequences, which are identified as virus. But, MGE operator perceives procedures and state transitions only with the proviso that there is no labeled square ahead of the ant, i.e. under condition *input=0* (See fig. 1).

State	Input=0
0	**LFT/#17**
17	**FWD/#13**
13	**FWD/#21**
21	**LFT/#9**
9	**LFT/#0**

Figure 1. Here is an example of a virus. The virus is a closed five-element cycle of states transitions (0, 17, 13, 21, 9, and again, 0). There are 32 states at all. Each state determines two alternate actions, depending on input signal. The input signal is what an ant sees before him. If the cell before him is black then the input is 1, in opposite case the input is 0. Each of alternative actions includes one of four possible movements (FWD, RGT, LFT or NOP) and transition to the next state.

Two-place MGE operator provides the transmission of the virus from an ant to another one, thus realizing the reproduction procedure of this virus in gene pool of the host (ant) population. This procedure performs the following operations.

First, a pair of ants is chosen at random. Then, the chromosome of any of them is scanned in search of the virus. If the virus is found, it is replicated in the partner chromosome, irrespectively of initial record character in that chromosome. The chromosome scanning starts from the zero line (state#0) and goes on as far as the first virus is met. If no virus is met, scanning finishes up only when the chromosome record ends. So, scanning ceases irrespectively of the remaining chromosome un-scanned part content.

One-place MGE operator is a sort of point mutation, realized under particular conditions. This is what we call an *intelligent mutator*. In detail, the operator acts in such a way. If it finds a pattern in the predetermined length range, and the action NOP completes this pattern, then this instruction is substituted for the one of the three other actions (FWD, RGT or LFT). Specifically, this NOP is substituted for the action from the fifth element of the pattern, counted in order. But, if the found pattern is completed by the reference to the one of the elements inside pattern (internal cycle), then we have the following. The action of this element is substituted for the action of the fifth element, counted backward from the end, the reference being substituted for found at random reference to the element outside of the pattern.

3 RESULTS

The test trail, used in this work is illustrated in Fig. 2. It can be seen that up to the 64th element our trail coincide with the Los Altos one, but the next part of the trail includes chaotically scattered elements of high complexity. Being trained on much simpler preceding trail part, the ant is not prepared to surmount the subsequent, complicated sector (biologists would say that the ant is not pre-adapted to new conditions it faced with in this sector). More specifically, problems arise at attempts to get over gaps between the 64th and the 65th, or the 67th and the 68th cells.

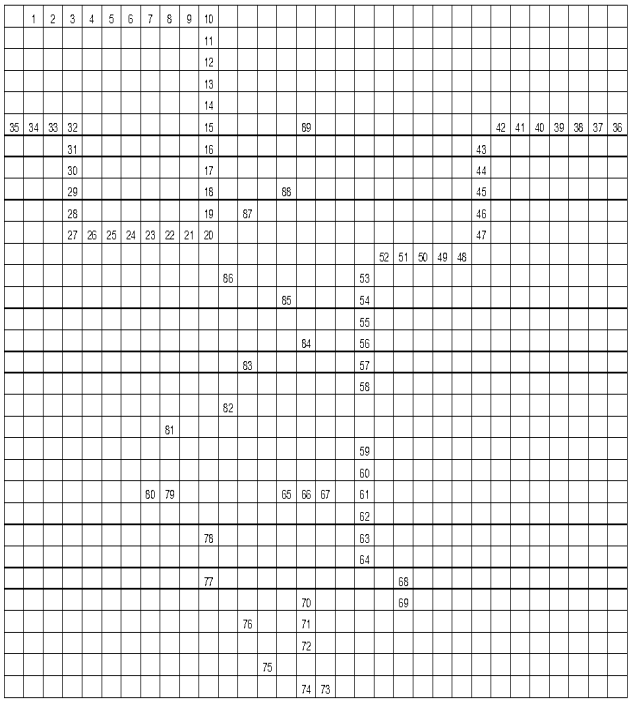


Figure 2. Ant trail used in our computer experiments. The trail itself is a series of squares on a 32x32 white toroidal grid. Each cell is numbered sequentially, from the 1st to the 89th. The first two gaps of the higher complexity are between 64th and 65th and 67th and 68th.

3.1 MGES REALLY ACCELERATES THE EVOLUTIONARY SEARCH

The preliminary computer experiments showed that the accelerating effect of MGE is especially noticeable for small populations, when the probability of the effective navigation algorithm finding by applying standard crossover and mutation operators is low.

On this basis, the following experiments were carried out on populations of 100 ants. The choice of such a small

population is also explained by our aim to carry out a comprehensive analysis of MGE dynamics. Such an analysis is not feasible for large populations of ants because of great number of viruses.

With the aim of demonstrating of the MGE technique efficiency we performed 100 independent runs of the program, 5000 generations each. The results of test and control runs (population with MGE and without MGE correspondingly) were compared in several series with the different values of standard mutation parameters. Everywhere in this section we will accept that the effective navigation algorithm should overcome the level of maximum score in 64 for 330 time steps.

The results of program runs with the MGE operator and without it are illustrated in Fig. 3. It can be seen, that MGE technique obviously increases the probability of finding of effective navigation algorithm for small populations and for a little number of generations.

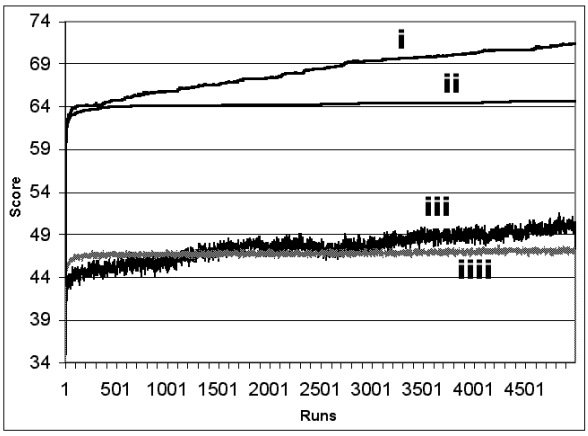


Figure 3. Numerical experiments, demonstrating statistically certain increasing of the GA efficiency due to the effect of MGE operators. A comparison of the mean and the best-of-generation score dynamics (MGE operator being activated) with the control (MGE operator is disabled). The score values are averaged over 100 runs in both cases. The size of population = 100; the number of generations = 5000; the pattern size varies from 5 to 11; crossover rate (P/bit)/generation = 0.0001; mutation rate (P/bit)/generation = 0.04; **i** are the best-of-generation scores and **iii** are the mean scores for the runs with MGE operators; **ii** are the best-of-generation scores and **iiii** are the mean scores for the control runs (without MGE operators).

As it is evident from the graphs on Fig. 3, the mean and the best-of-generation score scores in experiment and in control are growing, to a first approximation, linear in time. But the increment of growth in experiment with MGE is substantially higher, than in control.

It may be suggested that MGE operators raise ant variability mainly in nonspecific manner thus supplementing mutation effect of standard operators. But, this suggestion is not substantiated by the detailed analysis of mutation process. We carried out control runs with different values of standard mutations: the high level of standard mutation does not raise the effectiveness of the navigation algorithm search, moreover, it decreases this effectiveness.

3.2 HORIZONTAL TRANSMISSION OF MGES IS NECESSARY FOR THEIR EFFECTIVE ACTION

As far MGEs are transmitted vertically (from ancestors to descendants), MGE of the host, that have superiority in reproduction success is rapidly spreading in the population and gives new forms. But this process *per se* is insufficient for the effective acceleration of ant learning. Two-place MGE operator, performing horizontal distribution of MGE from one ant to another is a necessary for rising of ant training ability. In Fig. 4 we illustrate the results of comparing of the test, presented in Fig.3, with the similar test, in which frequency of applying of two-place MGE operator was reduced by the factor of 10 and accounted 5%. This parameter determines the proportion of population, which is subjected to the action the two-place MGE operator in a generation. In previous experiments, this quota accounted 50%.

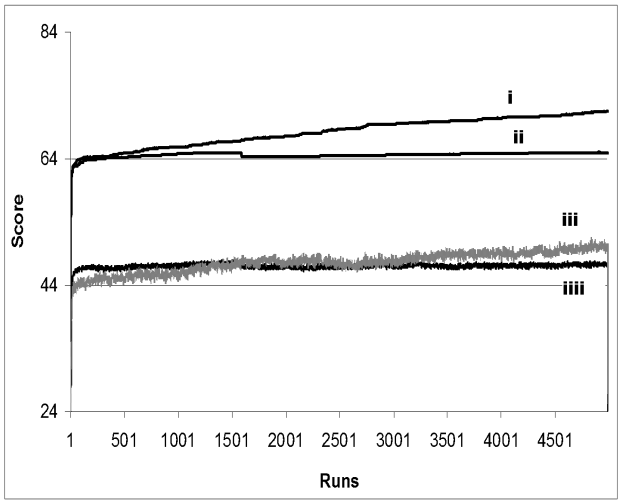


Figure 4. The influence of decreasing of frequency of applying of two-place MGE operator on the ant learning abilities. **i** are the best-of-generation scores and **iii** are the mean scores for the runs with high frequency of the two-place MGE operator action (50%); **ii** are the best-of-generation scores and **iiii** are the mean scores for the runs with low frequency of the two-place MGE operator action (5%). The other parameters are the same as in the previous experiments (see caption to Fig. 3).

The obvious lowering of ant learning abilities with the decreasing of frequency of the two-placed operator application is seen from the diagram. Disabling of the

operator lowers the efficacy further and makes it almost equal to the control (case without MGE).

4 DISCUSSION

The problem of programming an artificial ant to follow the Santa Fe trail has been repeatedly used as a benchmark problem in GP (For references See Langdon and Poli, 1998). Recently Langdon and Poli have shown that performance of several techniques is not much better than the best performance obtainable using uniform random search (Langdon and Poli, 1998). According to these authors, the search space is large and forms a Karst landscape containing many false peaks and many plateaus riven with deep valleys. The problem fitness landscape is difficult for hill climbers and the problem is also difficult for Genetic Algorithms as it contains multiple levels of deception.

There are many techniques capable of finding solutions to the ant problem (GA, GP, simulated annealing, hill clmbing) and although these have different performance the best typically only do marginally better than the best performance that could be obtained with random search (Langdon and Poli, 1998). That is why the ant problem may be indicative of real optimization problem spaces.

4.1 DOMINANT MGE ARE THE COMPONENTS OF THE EFFECTIVE NAVIGATION ALGORITHMS

The results of careful analysis of organization of several tens of dominant viruses, taken from those ant populations, which coped with the navigation task, can be summarized as follows.

1) By the definition, the virus program begins and ends with the zero state, i.e., it is a loop, executed over and over until the ant will meet the labeled cell.

2) Four-fold execution of the virus-program produces in most cases the closed ant trajectories, i.e., the ant will return to the starting position. As a rule, the closed contour is located in domains the size of 4×4 or 5×5 cells.

3) As a rule, the virus-program is beginning to work not from the zero state but from the Nth state, which is specific to every virus, not beginning with the initial, zero state. This transition into the Nth state takes place as soon as the ant (host of the virus) runs against the unlabeled cell.

4) Start the virus-program from the Nth state provides the execution of the simplest navigation algorithm, necessary for overcoming the simplest gaps, arranged in the first half of the trail ("looking around", then one step ahead, "looking around" again and so forth). This algorithm provides the successful passage of trail up to the 64^{th} cell inclusive.

5) The majority of program-viruses guarantee overcoming of the element of high complexity between the 64^{th} and the 65^{th} cells.

6) Some viruses are not suitable for the navigation programs. In that case the chromosome elements, arranged in virus-free domain take control over navigation.

The detailed analysis of the organization of dominant MGE forms in populations, which are succeeded in finding of the effective navigation programs, showed, that the MGE themselves become the components of these programs. Namely, the case in point is about the part of navigation program that is used for effective "snuffing around" in situation, when ant faces with a wide gap.

4.2 WISE MUTATORS HAVE A SEARCH SPACE CONFINING EFFECT

The Muir's Trail search space has rugged geometry due to specific and discrete character of the problem. That is why, the gradient methods are not effective here. Moreover, this ant navigation problem is classified as a GA hard problem, especially if trail is not designed specially for ant population training. The efficiency of MGE in the role of intelligent mutators can be measured by their search space domain confining ability. Therefore, the selection criteria inserted into MGE operators had to increase the probability of the effective navigation algorithm finding on the element of high complexity.

A comparison of mutation frequencies in experiment and control with the according learning rates confirms multiple reduction of evaluation numbers, needed for reaching of the same required learning in experiments with MGE. Mutation frequencies for basic experiments (Fig.3) in control accounts: crossover rate + mutation rate = 0.0001+0.04 P/bit/generation; MGE1 and MGE2 operators add in average 0.0027 and 0.0075 P/bit/generation accordingly. In other words, MGE in average adds to value 0.041 about 0.012 P/bit/generation. This addition brings to multiple acceleration of ant population learning! Hence, according to fig. 3, up to the end of the experiment (4622 time-step) the control set gives max score 6.47, whereas in the test set this value is attained already on the 451 time-step, i.e. 10 times sooner.

5 CONCLUSIONS

- The enhancement of GA by jumping genes-mutators substantially increases the efficacy of GA performance in known benchmark test – ant problem.
- The jumping genes-mutators (artificial transposons) act as intelligent mutators, that "elaborate" code blocks with high evolvability value.

Acknowledgments

This work is supported by INTAS grant No 97-3095.

References

Altenberg L. (1994) The evolution of evolvability in genetic programming. In: K. E. Kinnear, ed. *Advances in Genetic Programming*. MIT Press, Cambridge, pp. 47-74.

Brennan P. (1994) ANT: Simulated Evolution on a PC, manuscript.

Brosius J. (1991) Retroposons - Seeds of evolution. *Science* 251, 753.

Burke D.S., De Jong K.A., Grefenstette J.J., Ramsey C.L. and Wu A. S. (1998) Putting more genetics into genetic algorithms, *Evolutionary Computation*, 6:4, 387-410.

Cliff D. and Grand S. (1999) The *Creatures* Global Digital Ecosystem. *Artificial Life* 5(1): 77-93.

Corno F., Reorda M. S. and Squillero G. (1998) The selfish gene algorithm: a new evolutionary optimization strategy. In: *Proceedings of the 1998 ACM symposium on Applied Computing*, February 27 - March 1, 1998, Atlanta, GA, USA, pp. 349-355.

Daniels M. (1999) Integrating Simulation Technologies with Swarm, *Agent Simulation: Applications, Models and Tools*, October 1999, Argonne National Laboratory, University of Chicago.

Doolittle W. F. and Sapienza C. (1980) Selfish genes, the phenotype paradigm and genome evolution. *Nature* 284: 601-603.

Harries K. and Smith P. (1997) Exploring alternative operators and search strategies in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, eds, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann, pp. 147-155.

Harvey I. (1996) The microbial genetic algorithm, unpublished work, available at ftp://ftp.cogs.susx.ac.uk/pub/users/inmanh/Microbe.ps.gz

Hillis W.D. (1990) Co-evolving parasites improve simulated evolution as an optimization procedure, *Physica D*, 42:228-234.

Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C. and Wang A. (1991) Evolution as a Theme in Artificial Life: The Genesys/Tracker System. In: *Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X*, edited by C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen. Addison-Wesley, pp.417-434.

De Jong K.A. and Potter M.A. (1995) Evolving complex structures via cooperative coevolution, In: *Forth Annual Conference on Evolutionary Computation*, San Diego, CA, 1-3 March 1995.

Koza J.R. (1992) *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Mass.

Langdon W. B. and Poli R. (1998) Why ants are hard. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann, pp.193-201.

Lee C-Y, and Antonsson E.K., Adaptive Evolvability via Non-Coding Segment Induced Linkage, Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, 2001.

Lones M.A. and Tyrrell A.M., Biomimetic Representation in Genetic Programming, In: *Proceedings of the Workshop on Computation in Gene Expression at the Genetic and Evolutionary Computation Conference 2001 (GECCO2001)*, San Francisco, California, USA, July 2001, pp. 199-204.

Luke S., Hamahashi S. and Kitano H. (1999) "Genetic" programming, GECCO-99: *Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W. et al, eds. San Fransisco: Morgan Kaufmann.

Makalowski W. (1995) SINEs as a Genomic Scrap Yard. Chap. 5 in *The Impact of Short Interspersed Elements (SINEs) on the Host Genome*, edited by Richard J. Maraia. Austin: R.G. Landes Company.

Nawa N. E., Furuhashi T., Hashiyama T. and Uchikawa Y. (1999) A study on the discovery of relevant fuzzy rules using pseudo-bacterial genetic algorithms, *IEEE Transactions on Industrial Electronics*, 7, (5), 608-616, October 1999.

Orgel L. E. and Crick F. H. C. (1980) Selfish DNA: The ultimate parasite. *Nature* 284: 604-607.

Olsson B. (1996) Optimization using a host-parasite model with variable-size distributed populations. In: *Proceedings of the 1996 IEEE 3rd International Conference on Evolutionary Computation*, IEEE Press, pp. 295-299.

Olsson B. (2001) Co-evolutionary search in asymmetric spaces, In Wang, P.P., ed., *Proceedings of The Fifth Joint Conference on Information Sciences*, Association for Intelligent Machinery, pp. 1040-1043.

Potter M.A. and De Jong K.A. (1994) A cooperative co-evolutionary approach to function optimization, In: *Third Parallel Problem Solving from Nature*, Jerusalem, Israel, pp 249-257.

Potter M.A. and De Jong K.A. (1995) Evolving neural networks with collaborative species, In: *Proc. of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 24-26 July 1995, pp. 340-345.

Ray T. S. (1991) An approach to the synthesis of life. In: Langton, C., C. Taylor, J. D. Farmer, & S. Rasmussen [eds], *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, vol. XI, Redwood City, CA: Addison-Wesley, pp. 371-408.

Ray T. S. (2001) Overview of Tierra at ATR. In: *Technical Information, No.15, Technologies for Software Evolutionary Systems.* ATR-HIP. Kyoto, Japan.

Simoes A. and Costa E. (2001) An evolutionary approach to the Zero/One knapsack problem: testing ideas from biology; In: *Procs. 5th Int. Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA 2001)*, Prague, Czech Republic, 22-25 April 2001.

Spirov A.V. (1996a) Self-Assemblage of gene Networks in Evolution via Recruiting of New Netters. *Lecture Notes in Computer Sciences.* 1141: 91-100.

Spirov A.V. (1996b) Self-organisation of gene networks in evolution via recruiting of new netters. In: *Proceedings of the First International Conference on Evolutionary Computations and Its Applications*, Moscow, Russia, pp. 399-405.

Spirov A.V. and Samsonova M.G. (1997) Strategy of Co-evolution of Transposons and Host Genome: Application to Evolutionary Computations. In: *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, 20 - 22 August 1997, Helsinki, Finland, Ed. Jarmo T. Alander, Finnish Artificial Intelligence Society, pp. 71-82.

Spirov A.V., Kadyrov A.S. (1998) Transposon Element Technique Applied to GA-based John Muir's Trail Test, In: *High-Performance Computing and Netw*orking, pp. 925-928.

Spirov A.V., Kazansky A.B. and Kadyrov A.S. (1998) Utilizing of "Parasitic" Mobile Genetic Elements in Genetic Algorithms. In: *International Conference on Soft Computing and Measurments*, St.Petersburg, pp. 266-269.

Spirov A.V. and Kazansky A.B. (1999) Evolutionary Biology and Evolutionary Computations: Parasitic Mobile Genetic Elements in Artifical Evolution, In: *2nd Int. Conf. on Soft Computing and Measurments*, St.Petersburg.

Zongker, D. and Punch, B. (1995) lil-gp 1.0, http://isl.cps.msu.edu/GA/software/lil-gp

Efficient Reinforcement Learning through Evolving Neural Network Topologies

Kenneth O. Stanley
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
kstanley@cs.utexas.edu

Risto Miikkulainen
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

Abstract

Neuroevolution is currently the strongest method on the pole-balancing benchmark reinforcement learning tasks. Although earlier studies suggested that there was an advantage in evolving the network topology as well as connection weights, the leading neuroevolution systems evolve fixed networks. Whether evolving structure can improve performance is an open question. In this article, we introduce such a system, NeuroEvolution of Augmenting Topologies (NEAT). We show that when structure is evolved (1) with a principled method of crossover, (2) by protecting structural innovation, and (3) through incremental growth from minimal structure, learning is significantly faster and stronger than with the best fixed-topology methods. NEAT also shows that it is possible to evolve populations of increasingly large genomes, achieving highly complex solutions that would otherwise be difficult to optimize.

1 INTRODUCTION

Many tasks in the real world involve learning with sparse reinforcement. Whether navigating a maze of rubble in search of survivors, controlling a bank of elevators, or making a tactical decision in a game, there is frequently no immediate feedback available to evaluate recent decisions. It is difficult to optimize such complex systems by hand; thus, learning with sparse reinforcement is a substantial goal for AI.

Neuroevolution (NE), the artificial evolution of neural networks using genetic algorithms, has shown great promise in reinforcement learning tasks. For example, on the most difficult versions of the pole balancing problem, which is the standard benchmark for reinforcement learning systems, NE methods have recently outperformed other reinforcement learning techniques (Gruau et al. 1996; Moriarty and Miikkulainen 1996).

Most NE systems that have been tested on pole balancing evolve connection weights on networks with a fixed topology (Gomez and Miikkulainen 1999; Moriarty and Miikkulainen 1996; Saravanan and Fogel 1995; Whitley et al. 1993; Wieland 1991). On the other hand, NE systems that evolve both network topologies and connection weights simultaneously have also been proposed (Angeline et al. 1993; Gruau et al. 1996; Yao 1999). A major question in NE is whether such Topology and Weight Evolving Artificial Neural Networks (TWEANNs) can enhance the performance of NE. On one hand, evolving topology along with weights might make the search more difficult. On the other, evolving topologies can save the time of having to find the right number of hidden neurons for a particular problem (Gruau et al. 1996).

In a recent study, a topology-evolving method called Cellular Encoding (CE; Gruau *et al.*, 1996) was compared to a fixed-network method called Enforced Subpopulations (ESP) on the double pole balancing task without velocity inputs (Gomez and Miikkulainen 1999). Since ESP had no a priori knowledge of the correct number of hidden nodes for solving the task, each time it failed, it was restarted with a new random number of hidden nodes. However, even then, ESP was five times faster than CE. In other words, evolving structure did not improve performance in this study.

This article aims to demonstrate the opposite conclusion: if done right, evolving structure along with connection weights can significantly enhance the performance of NE. We present a novel NE method called NeuroEvolution of Augmenting Topologies (NEAT) that is designed to take advantage of structure as a way of minimizing the dimensionality of the search space of connection weights. If structure is evolved such that topologies are minimized and grown incrementally, significant performance gains result.

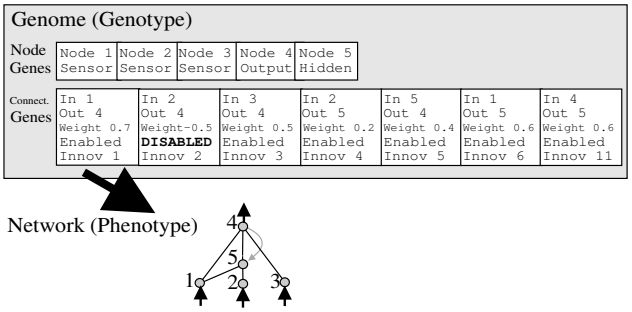


Figure 1: **A Genotype to Phenotype Mapping Example.** A genotype is depicted that produces the shown phenotype. Notice that the second gene is disabled, so the connection that it specifies (between nodes 2 and 4) is not expressed in the phenotype.

Evolving structure incrementally presents several technical challenges: (1) Is there a genetic representation that allows disparate topologies to crossover in a meaningful way? (2) How can topological innovation that needs a few generations to optimize be protected so that it does not disappear from the population prematurely? (3) How can topologies be minimized *throughout evolution* without the need for a specially contrived fitness function that measures complexity?

The NEAT method consists of solutions to each of these problems as will be described below. The method is validated on pole balancing tasks, where NEAT performs 25 times faster than Cellular Encoding and 5 times faster than ESP. The results show that structure is a powerful resource in NE when appropriately utilized.

2 NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

NEAT is designed to address the three problems with TWEANNs raised in the Introduction. We begin by explaining the genetic encoding used in NEAT, and continue by describing the components that specifically address each issue.

2.1 GENETIC ENCODING

NEAT's genetic encoding scheme is designed to allow corresponding genes to be easily lined up when two genomes crossover during mating. Thus, genomes are linear representations of network connectivity (figure 1). Each genome includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows finding corresponding genes (as will be explained below).

Mutation in NEAT can change both connection weights and

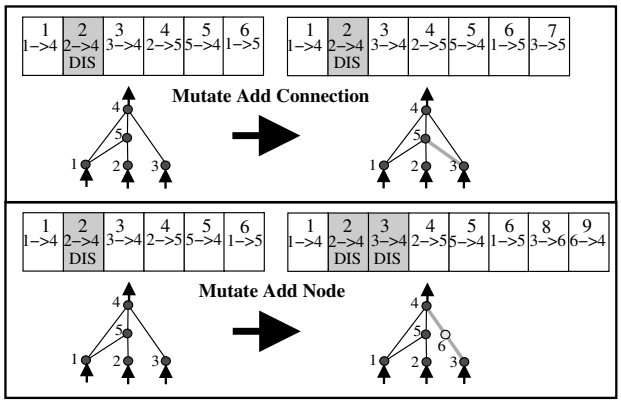


Figure 2: **The two types of structural mutation in NEAT.** Both types, adding a connection and adding a node, are illustrated with the genes above their phenotypes. The top number in each genome is the *innovation number* of that gene. The innovation numbers are historical markers that identify the original historical ancestor of each gene. New genes are assigned new increasingly higher numbers.

network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not at each generation. Structural mutations occur in two ways (figure 2). Each mutation expands the size of the genome by adding gene(s). In the *add connection* mutation, a single new connection gene is added connecting two previously unconnected nodes. In the *add node* mutation an existing connection is split and the new node placed where the old connection used to be. The old connection is disabled and two new connections are added to the genome. This method of adding nodes was chosen in order to integrate new nodes immediately into the network.

Through mutation, the genomes in NEAT will gradually get larger. Genomes of varying sizes will result, sometimes with completely different connections at the same positions. How can NE cross them over in a sensible way? The next section explains how NEAT addresses this problem.

2.2 TRACKING GENES THROUGH HISTORICAL MARKINGS

It turns out that there is unexploited information in evolution that tells us exactly which genes match up with which genes between *any* individuals in a topologically diverse population. That information is the historical origin of each gene in the population. Two genes with the same historical origin must represent the same structure (although possibly with different weights), since they are both derived from the same ancestral gene from some point in the past. Thus, all a system needs to do to know which genes line up with which is to keep track of the historical origin of every gene in the system.

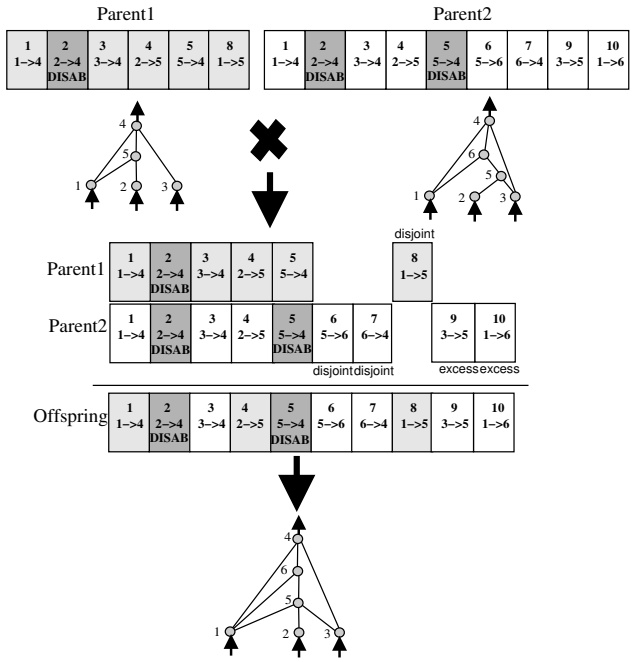


Figure 3: **Matching Up Genomes for Different Network Topologies Using Innovation Numbers.** Although Parent 1 and Parent 2 look different, their innovation numbers (shown at the top of each gene) tell us which genes match up with which. Even without any topological analysis, a new structure that combines the overlapping parts of the two parents as well as their different parts can be created. In this case the parents are equally fit and the genes are inherited from both parents. Otherwise, the offspring inherit only the disjoint and excess genes of the most fit parent.

Tracking the historical origins requires very little computation. Whenever a new gene appears (through structural mutation), a *global innovation number* is incremented and assigned to that gene. The innovation numbers thus represent a chronology of the appearance of every gene in the system. As an example, let us say the two mutations in figure 2 occurred one after another in the system. The new connection gene created in the first mutation is assigned the number 7, and the two new connection genes added during the new node mutation are assigned the numbers 8 and 9. In the future, whenever these genomes mate, the offspring will inherit the same innovation numbers on each gene; innovation numbers are never changed. Thus, the historical origin of every gene in the system is known throughout evolution.

The historical markings give NEAT a powerful new capability, effectively avoiding the problem of competing conventions (Montana and Davis 1989; Radcliffe 1993; Schaffer et al. 1992). The system now knows exactly which genes match up with which (figure 3). When crossing over, the genes in both genomes with the same innovation numbers are lined up. These genes are called *matching* genes. Genes that do not match are either *disjoint* (D) or *excess* (E), depending on whether they occur within or outside the range of the other parent's innovation numbers. They represent structure that is not present in the other genome. In composing the offspring, genes are randomly chosen from either parent at matching genes, whereas all excess or disjoint genes are always included from the more fit parent, or if they are equally fit, from both parents. This way, historical markings allow NEAT to perform crossover using linear genomes without the need for expensive topological analysis.

By adding new genes to the population and sensibly mating genomes representing different structures, the system can form a population of diverse topologies. However, it turns out that such a population on its own cannot maintain topological innovations. Because smaller structures optimize faster than larger structures, and adding nodes and connections usually initially decreases the fitness of the network, recently augmented structures have little hope of surviving more than one generation even though the innovations they represent might be crucial towards solving the task in the long run. The solution is to protect innovation by speciating the population, as explained in the next section.

2.3 PROTECTING INNOVATION THROUGH SPECIATION

Speciation is commonly applied to multimodal function optimization and the coevolution of modular systems, where its main function is to preserve diversity (Mahfoud 1995; Potter and De Jong 1995). We borrow the idea from these fields and bring it to TWEANNs, where it protects innovation. Speciation allows organisms to compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected in a new niche where they have time to optimize their structure through competition within the niche.

The idea is to divide the population into species such that similar topologies are in the same species. This task appears to be a topology matching problem. However, it again turns out that historical markings offer a more efficient solution.

The number of excess and disjoint genes between a pair of genomes is a natural measure of their compatibility. The more disjoint two genomes are, the less evolutionary history they share, and thus the less compatible they are. Therefore, we can measure the compatibility distance δ of different structures in NEAT as a simple linear combination of the number of excess (E) and disjoint (D) genes, as well as the average weight differences of matching genes ($\overline{W}$):

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}. \quad (1)$$

The coefficients, c_1, c_2, and c_3, allow us to adjust the im-

portance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size (N can be set to 1 if both genomes are small, i.e. consist of fewer than 20 genes).

The distance measure δ allows us to speciate using a compatibility threshold δ_t. Genomes are compared to each species one at a time; if a genomes' distance to a randomly chosen member of the species is less than δ_t, it is placed into this species. Each genome is placed into the first species where this condition is satisfied, so that no genome is in more than one species. Measuring δ for a pair of genomes is linear in the number of connections even though δ precisely expresses compatibility between multidimensional topologies. This efficiency is possible because of the historical markings.

As the reproduction mechanism for NEAT, we use *explicit fitness sharing* (Goldberg and Richardson 1987), where organisms in the same species must share the fitness of their niche. Thus, a species cannot afford to become too big even if many of its organisms perform well. Therefore, any one species is unlikely to take over the entire population, which is crucial for speciated evolution to work. The original fitnesses are first adjusted by dividing by the number of individuals in the species. Species then grow or shrink depending on whether their average adjusted fitness is above or below the population average:

$$N'_j = \frac{\sum_{i=1}^{N_j} f_{ij}}{\overline{f}}, \qquad (2)$$

where N_j and N'_j are the old and the new number of individuals in species j, f_{ij} is the adjusted fitness of individual i in species j, and $\overline{f}$ is the mean adjusted fitness in the entire population. The best-performing $r\%$ of each species is randomly mated to generate N'_j offspring, replacing the entire population of the species. [1]

The net effect of speciating the population is that topological innovation is protected. The final goal of the system, then, is to perform the search for a solution as efficiently as possible. This goal is achieved through minimizing the dimensionality of the search space.

2.4 MINIMIZING DIMENSIONALITY THROUGH INCREMENTAL GROWTH FROM MINIMAL STRUCTURE

TWEANNs typically start with an initial population of random topologies (Angeline et al. 1993; Dasgupta and McGregor 1992; Gruau et al. 1996; Zhang and Muhlenbein 1993). This way topological diversity is introduced to the population from the outset. However, it is not clear that such diversity is necessary or useful. A population of random topologies has a great deal of unjustified structure that has not withstood a single fitness evaluation. Therefore, there is no way to know if any of such structure is *necessary*. It is costly though because the more connections a network contains, the higher the number of dimensions that need to be searched to optimize the network. Therefore, with random topologies the algorithm may waste a lot of effort by optimizing unnecessarily complex structures.

In contrast, NEAT biases the search towards minimal-dimensional spaces by starting out with a *uniform* population of networks with zero hidden nodes (i.e. all inputs connect directly to outputs). New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. In other words, the structural elaborations that occur in NEAT are always justified. Since the population starts minimally, the dimensionality of the search space is minimized, and NEAT is always searching through fewer dimensions than other TWEANNs and fixed-topology NE systems. Minimizing dimensionality gives NEAT a performance advantage compared to other approaches, as will be discussed next.

3 POLE BALANCING EXPERIMENTS

3.1 POLE BALANCING AS A BENCHMARK TASK

There are many reinforcement learning tasks where the techniques employed in NEAT can make a difference. Many of these potential applications, like robot navigation or game playing, are open problems where evaluation is difficult. In this paper, we focus on the pole balancing domain because it has been used as a reinforcement learning benchmark for over 30 years (Anderson 1989; Barto et al. 1983; Gomez and Miikkulainen 1999; Gruau et al. 1996; Michie and Chambers 1968; Moriarty and Miikkulainen 1996; Saravanan and Fogel 1995; Watkins and Dayan 1992; Whitley et al. 1993; Wieland 1991, 1990), which makes it easy to compare to other methods. It is also a good surrogate for real problems, in part because pole balancing in fact *is* a real task, and also because the difficulty can be adjusted.

Earlier comparisons were done with a single pole, but this version of the task has become too easy for modern methods. Therefore, we demonstrate the advantage of evolving structure through double pole balancing experiments. Two poles are connected to a moving cart by a hinge and the neural network must apply force to the cart to keep the poles balanced for as long as possible without going

[1] In rare cases when the fitness of the entire population does not improve for more than 20 generations, only the top two species are allowed to reproduce, refocusing the search into the most promising spaces.

beyond the boundaries of the track. The system state is defined by the cart position (x) and velocity ($\dot{x}$), the first pole's position (θ_1) and angular velocity ($\dot{\theta}_1$), and the second pole's position (θ_2) and angular velocity ($\dot{\theta}_2$). Control is possible because the poles have different lengths and respond differently to control inputs.

Double-pole balancing is sufficiently challenging even for the best current methods. Neuroevolution generally performs better in this task than standard reinforcement learning based on value functions and policy iteration (such as Q-learning and VAPS; Watkins and Dayan 1992, Meauleau et al. 1999, Gomez and Miikkulainen 2002). The question studied in this paper is therefore whether evolving structure can lead to greater NE performance.

3.2 COMPARISONS

Two versions of the double pole balancing task are used: one with velocity inputs included and another without velocity information. The first task is Markovian and allows comparing to many different systems. Taking away velocity information makes the task more difficult because the network must estimate an internal state in lieu of velocity, which requires recurrent connections.

On the double pole balancing with velocity (DPV) problem, NEAT is compared to published results from four other NE systems. The first two represent standard population-based approaches (Saravanan and Fogel 1995; Wieland 1991). Saravanan and Fogel used Evolutionary Programming, which relies entirely on mutation of connection weights, while Wieland used both mating and mutation. The second two systems, SANE (Moriarty and Miikkulainen 1996) and ESP (Gomez and Miikkulainen 1999), evolve populations of neurons and a population of network blueprints that specifies how to build networks from the neurons that are assembled into fixed-topology networks for evaluation. SANE maintains a single population of neurons. ESP improves over SANE by maintaining a separate population for each hidden neuron position in the complete network. To our knowledge, the results of ESP are the best achieved so far in this task.

On the double pole balancing without velocity problem (DPNV), NEAT is compared to the only two systems that have been demonstrated able to solve the task: Cellular Encoding (CE; Gruau *et al.*, 1996), and ESP. The success of CE was first attributed to its ability to evolve structures. However, ESP, a fixed-topology NE system, was able to complete the task five times faster simply by restarting with a random number of hidden nodes whenever it got stuck. Our experiments will attempt to show that evolution of structure can lead to better performance if done right.

3.3 PARAMETER SETTINGS

We set up our pole balancing experiments as described by Wieland (1991) and Gomez (1999). The Runge-Kutta fourth-order method was used to implement the dynamics of the system, with a step size of 0.01s. All state variables were scaled to $[-1.0, 1.0]$ before being fed to the network. Networks output a force every 0.02 seconds between $[-10, 10]N$. The poles were 0.1m and 1.0m long. The initial position of the long pole was 1° and the short pole was upright; the track was 4.8 meters long.

The DPV experiment used a population of 150 NEAT networks while the DPNV experiment used a population of 1,000. The larger population reflects the difficulty of the task. ESP evaluated 200 networks per generation for DPV and 1000 for DPNV, while CE had a population of 16,384 networks. The coefficients for measuring compatibility were $c_1 = 1.0$ and $c_2 = 1.0$ for both experiments. For DPNV, $c_3 = 3.0$ and $\delta_t = 4.0$. For DPV, $c_3 = 0.4$ and $\delta_t = 3.0$. The difference in the c_3 coefficient reflects the size of the populations; a larger population has more room for distinguishing species based on connection weights, whereas the smaller population relies more on topology.

If the maximum fitness of a species did not improve in 15 generations, the networks in that species were not allowed to reproduce. Otherwise, the top 40% (i.e. the elite) of each species reproduced by random mate selection within the elite. In addition, the champion of each species with more than five networks was copied into the next generation unchanged and each elite individual had a 0.1% chance to mate with an elite individual from another species. The offspring inherited matching genes randomly from either parent, and disjoint and excess genes from the better parent, as described in section 2.2. While other crossover schemes are possible, this method was found effective and did not cause excessive bloating of the genomes.

There was an 80% chance that the connection weights of an offspring genome were mutated, in which case each weight had a 90% chance of being uniformly perturbed and a 10% chance of being assigned a new random value. The system tolerates frequent mutations because speciation protects radically different weight configurations in their own species. In the smaller population, the probability of adding a new node was 0.03 and the probability of a new link was 0.05. In the larger population, the probability of adding a new link was 0.3, because a larger population has room for a larger number of species and more topological diversity.

We used a modified sigmoidal transfer function, $\varphi(x) = \frac{1}{1+e^{-4.9x}}$, at all nodes. The steepened sigmoid allows more fine tuning at extreme activations. It is optimized to be close to linear during its steepest ascent between activations -0.5 and 0.5.

Method	Evaluations	Generations	No. Nets
Ev. Programming	**307,200**	150	2048
Conventional NE	**80,000**	800	100
SANE	**12,600**	63	200
ESP	**3,800**	19	200
NEAT	**3,578**	24	150

Table 1: **Double Pole Balancing with Velocity Information.** Evolutionary programming results were obtained by Saravanan (1995). Conventional neuroevolution data was reported by Wieland (1991). SANE and ESP results were reported by Gomez (1999). NEAT results are averaged over 120 experiments. All other results are averages over 50 runs. The standard deviation for the NEAT evaluations is 2704 evaluations. Although standard deviations for other methods were not reported, if we assume similar variances, all differences are statistically significant ($p < 0.001$), except that between NEAT and ESP.

3.4 DOUBLE POLE BALANCING WITH VELOCITIES

The criteria for success on this task was keeping both poles balanced for 100,000 time steps (30 minutes of simulated time). A pole was considered balanced between -36 and 36 degrees from vertical.

Table 1 shows that NEAT takes the fewest evaluations to complete this task, although the difference between NEAT and ESP is not statistically significant. The fixed-topology NE systems evolved networks with 10 hidden nodes, while NEAT's solutions always used between 0 and 4 hidden nodes. Thus, it is clear that NEAT's minimization of dimensionality is working on this problem. The result is important because it shows that NEAT performs as well as ESP while finding more minimal solutions.

3.5 DOUBLE POLE BALANCING WITHOUT VELOCITIES

Gruau *et al.* introduced a special fitness function for this problem to prevent the system from solving the task simply by moving the cart back and forth quickly to keep the poles wiggling in the air. (Such a solution does not require computing the missing velocities.) Because both CE and ESP were evaluated using this special fitness function, NEAT uses it on this task as well. The fitness penalizes oscillations. It is the sum of two fitness component functions, f_1 and f_2, such that $F = 0.1f_1 + 0.9f_2$. The two functions are defined over 1000 time steps:

$$f_1 = t/1000 \tag{3}$$

$$f_2 = \begin{cases} 0 & \text{if } t < 100, \\ \frac{0.75}{\sum_{i=t-100}^{t}(|x^i|+|\dot{x}^i|+|\theta_1^i|+|\dot{\theta}_1^i|)} & \text{otherwise.} \end{cases} \tag{4}$$

where t is the number of time steps the pole remains balanced during the 1000 total time steps. The denominator in (4) represents the sum of offsets from center rest of the cart and the long pole. It is computed by summing the absolute value of the state variables representing the cart and long pole positions and velocities. Thus, by minimizing these offsets (damping oscillations), the system can maximize fitness. Because of this fitness function, swinging the poles wildly is penalized, forcing the system to internally compute the hidden state variables.

Method	Evaluations	Generalization	No. Nets
CE	**840,000**	300	16,384
ESP	**169,466**	289	1,000
NEAT	**33,184**	286	1,000

Table 2: **Double Pole Balancing without Velocity Information (DPNV).** CE is Cellular Encoding of Gruau (1996). ESP is Enforced Subpopulations of Gomez (1999). All results are averages over 20 simulations. The standard deviation for NEAT is 21,790 evaluations. Assuming similar variances for CE and ESP, all differences in number of evaluations are significant ($p < 0.001$). The generalization results are out of 625 cases in each simulation, and are not significantly different.

Under Gruau *et al.*'s criteria for a solution, the champion of each generation is tested on generalization to make sure it is robust. This test takes a lot more time than the fitness test, which is why it is applied only to the champion. In addition to balancing both poles for 100,000 time steps, the winning controller must balance both poles from 625 different initial states, each for 1000 times steps. The number of successes is called the *generalization performance of the solution*. In order to count as a solution, a network needs to generalize to at least 200 of the 625 initial states. Each start state is chosen by giving each state value (i.e. x, $\dot{x}$, θ_1, and $\dot{\theta}_1$) each of the values 0.05, 0.25, 0.5, 0.75, 0.95 scaled to the respective range of the input variable ($5^4 = 625$). At each generation, NEAT performs the generalization test on the champion of the highest-performing species that improved since the last generation.

Table 2 shows that NEAT is the fastest system on this challenging task. NEAT takes 25 times fewer evaluations than Gruau's original benchmark, showing that the way in which structure is evolved has significant impact on performance. NEAT is also 5 times faster than ESP, showing that structure can indeed perform better than evolution of fixed topologies. There was no significant difference in the ability of any of the 3 methods to generalize.

4 DISCUSSION AND FUTURE WORK

4.1 EXPLAINING PERFORMANCE

Why is NEAT so much faster than ESP on the more difficult task when there was not much difference in the easier task? The reason is that in the task without velocities,

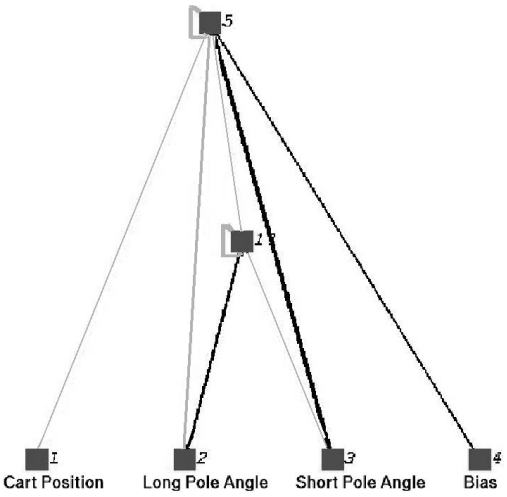


Figure 4: **A NEAT Solution to the DPNV Problem.** Node 2 is the angle of the long pole and node 3 is the angle of the short pole. This clever solution works by taking the derivative of the difference in pole angles. Using the recurrent connection to itself, the single hidden node determines whether the poles are falling away or towards each other. This solution allows controlling the system without computing the velocities of each pole separately. Without evolving structure, it would be difficult to discover such subtle and compact solutions.

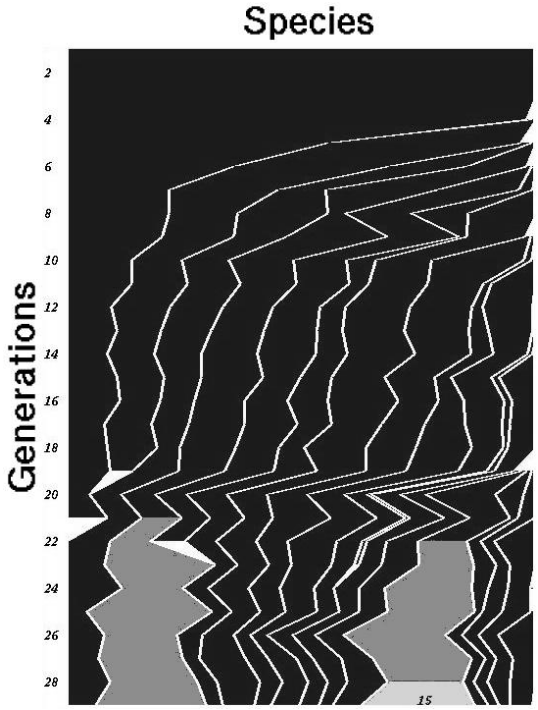


Figure 5: **Visualizing speciation.** The fixed-size population is divided into species, shown horizontally with newer species appearing at right. Time, i.e. evolution generations, are shown vertically. The color coding indicates fitness of the species (lighter colors are better). Two species began to close in on a solution soon after the 20th generation. Around the same time, some of the oldest species became extinct.

ESP needed to restart an average of 4.06 times per solution while NEAT never needed to restart. If restarts are factored out, the systems perform at similar rates. NEAT evolves many different structures simultaneously in different species, each representing a space of different dimensionality. Thus, NEAT is always trying many different ways to solve the problem at once, so it is less likely to get stuck.

Figure 4 shows a sample solution network that NEAT developed for the problem without velocities. The solution clearly illustrates the advantage of incrementally evolving structure. The network is a compact and elegant solution to this problem, in sharp contrast to the fully-connected large networks evolved by the fixed-topology methods. It shows that minimal necessary structures are indeed found, even when it would be difficult to discover them otherwise.

A parallel can be drawn between structure evolution in NEAT and incremental evolution in fixed structures (Gomez and Miikkulainen 1997; Wieland 1991). NE is likely to get stuck on a local optimum when attempting to solve a difficult task directly. However, after solving an easier version of the task first, the population is likely to be in a part of fitness space closer to a solution to a harder task, allowing it to avoid local optima. This way, a difficult task can be solved by evolving networks in incrementally more challenging tasks. Adding structure to a solution is analogous to this process. The network structure before the addition is optimized in a lower-dimensional space. When structure is added, the network is placed into a more complex space where it is already close to a solution. This process is different from incremental evolution in that adding structure is *automatic* in NEAT whereas the sequence of progressively harder tasks must be designed by the experimenter, and can be a challenging problem in itself.

4.2 VISUALIZING SPECIATION

To understand how innovation takes place in NEAT, it is important to understand the dynamics of speciation. How many species form over the course of a run? How often do new species arise? How often do species die? How large do the species get? We answer these questions by depicting speciation visually over time.

Figure 5 depicts a typical run of the double pole balancing with velocities task. In this run, the task took 29 generations to complete, which is slightly above average. In the visualization, successive generations are shown from top to bottom. Species are depicted horizontally for each generation, with the width of each species proportional to its size during the corresponding generation. Species are divided

from each other by white lines, and new species always arrive on the right hand side. Gray-scale shading is used to indicate the fitness of each species. A species is colored dark grey if it has individuals that are more than one standard deviation above the mean fitness for the run, and light grey if they are two standard deviations above. These two tiers identify the most promising species and those that are very close to a solution. Thus, it is possible to follow any species from its inception to the end of the run.

Figure 5 shows that only one species existed in the population until the 5th generation, that is, all organisms were sufficiently compatible to be grouped into a single species. In successive generations, the initial species shrank dramatically in order to make room for the new species, and eventually became extinct in the 21st generation. Extinction is shown by a white triangle between the generation it expired and the next generation. The initial species with minimal structure was unable to compete with newer, more innovative species. The second species to appear in the population met a similar fate in the 19th generation.

In the 21st generation a structural mutation in the fourth species connected the long pole angle sensor to a hidden node that had previously only been connected to the cart position sensor. This innovation allowed the networks to combine these observations, leading to a significant boost in fitness (and brightening of the species in figure 5). This innovative species subsequently expanded, but did not take over the population. Nearly simultaneously, in the 22nd generation, a younger species also made its own useful connection, this time between the short pole velocity sensor and long pole angle sensor, leading to its own subsequent expansion. In the 28th generation, this same species made a pivotal connection between the cart position and its already established method for comparing short pole velocity to long pole angle. This innovation was enough to solve the problem within one generation of additional weight mutations. In the final generation, the winning species was 11 generations old and included 38 neural networks out of the population of 150.

Most of the species that did not come close to a solution survived the run even though they fell significantly behind around the 21st generation. This observation is important, because it visually demonstrates that innovation is indeed being protected. The winning species does not take over the entire population.

4.3 FUTURE WORK

NEAT strengthens the analogy between GAs and natural evolution by not only performing the optimizing function of evolution, but also a *complexifying* function, allowing solutions to become incrementally more complex at the same time as they become more optimal. This is potentially a very powerful extension, and will be further explored in future work.

One potential application of complexification is continual coevolution. In a companion paper (Stanley and Miikkulainen 2002) we demonstrate how NEAT can add new structure to an existing solution, achieving more complex behavior while maintaining previous capabilities. Thus, an arms race of increasingly more sophisticated solutions can take place. Strategies evolved with NEAT not only reached a higher level of sophistication than those evolved with fixed-topologies, but also continued to improve for significantly more generations.

Another direction of future work is to extend NEAT to tasks with a high number of inputs and outputs. For such networks, the minimal initial structure may have to be defined differently than for networks with few inputs and outputs. For example, a fully connected two-layer network with 30 inputs and 30 outputs would require 900 connections. On the other hand, the same network with a five-unit hidden layer would require only 300 connections. Thus, the three-layer network is actually simpler, implying that the minimal starting topology for such domains should include hidden nodes.

Finally, the NEAT method can potentially be extended to solution representations other than neural networks. In any domain where solutions can be represented with different levels of complexity, the search for solutions can begin with a minimal representation that is progressively augmented as evolution proceeds. For example, the NEAT method may be applied to the evolution of hardware (Miller et al. 200a,b), cellular automata (Mitchell et al. 1996), or genetic programs (Koza 1992). NEAT provides a principled methodology for implementing a complexifying search from a minimal starting point in any such structures.

5 CONCLUSION

The main conclusion is that evolving structure and connection weights in the style of NEAT leads to significant performance gains in reinforcement learning. NEAT exploits properties of both structure and history that have not been utilized before. Historical markings, protection of innovation through speciation, and incremental growth from minimal structure result in a system that is capable of evolving solutions of minimal complexity. NEAT is a unique TWEANN method in that its genomes can grow in complexity as necessary, yet no expensive topological analysis is necessary either to crossover or speciate the population. It forms a promising foundation on which to build reinforcement learning systems for complex real world tasks.

Acknowledgments

This research was supported in part by the NSF under grant IIS-0083776 and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001. Thanks to Faustino Gomez for providing pole balancing code.

References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37.

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1993). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5:54–65.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846.

Dasgupta, D., and McGregor, D. (1992). Designing application-specific neural networks using the structured genetic algorithm. In *Proceedings of the International Conference on Combinations of Genetic Algorithms and Neural Networks*, 87–96.

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, 148–154. San Francisco, CA: Morgan Kaufmann.

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

Gomez, F., and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Denver, CO: Morgan Kaufmann.

Gomez, F., and Miikkulainen, R. (2001). Learning robust nonlinear control with neuroevolution. Technical Report AI01-292, Department of Computer Sciences, The University of Texas at Austin.

Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

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

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence*.

Michie, D., and Chambers, R. A. (1968). BOXES: An experiment in adaptive control. In Dale, E., and Michie, D., editors, *Machine Intelligence*. Edinburgh, UK: Oliver and Boyd.

Miller, J. F., Job, D., and Vassilev, V. K. (200a). Principles in the evolutionary design of digital circuits – Part I. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35.

Miller, J. F., Job, D., and Vassilev, V. K. (200b). Principles in the evolutionary design of digital circuits – Part II. *Journal of Genetic Programming and Evolvable Machines*, 3(2):259–288.

Mitchell, M., Crutchfield, J. P., and Das, R. (1996). Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*. Russian Academy of Sciences.

Montana, D. J., and Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 762–767. San Francisco, CA: Morgan Kaufmann.

Moriarty, D. E., and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

Potter, M. A., and De Jong, K. A. (1995). Evolving neural networks with collaborative species. In *Proceedings of the 1995 Summer Computer Simulation Conference*.

Radcliffe, N. J. (1993). Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90.

Saravanan, N., and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, 23–27.

Schaffer, J. D., Whitley, D., and Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In Whitley, D., and Schaffer, J., editors, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, 1–37. IEEE Computer Society Press.

Stanley, K. O., and Miikkulainen, R. (2002). Continual coevolution through complexification. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann.

Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Whitley, D., Dominic, S., Das, R., and Anderson, C. W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284.

Wieland, A. (1991). Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), 667–673. Piscataway, NJ: IEEE.

Wieland, A. P. (1990). Evolving controls for unstable systems. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Connectionist Models: Proceedings of the 1990 Summer School*, 91–102. San Francisco, CA: Morgan Kaufmann.

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

Zhang, B.-T., and Muhlenbein, H. (1993). Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 7:199–220.

Exact Results from a Coarse Grained Formulation of the Dynamics of Variable-length Genetic Algorithms

Christopher R. Stephens
Instituto de Ciencias Nucleares
UNAM, Circuito Exterior
Mexico D.F. 04510
stephens@nuclecu.unam.mx

Riccardo Poli
Dept. of Computer Science
University of Essex,
Colchester, CO4 3SQ, UK
rpoli@essex.ac.uk

Alden H. Wright
Dept. of Computer Science
University of Montana, USA
wright@cs.umt.edu

Jonathan E. Rowe
School of Computer Science
University of Birmingham, UK
j.e.rowe@cs.bham.ac.uk

Abstract

We consider the dynamics of variable-length Genetic Algorithms (GAs) with strings of length $N \leq N_m$ using a recently developed exact, coarse-grained formulation where the relevant coarse-grained degrees of freedom are "building block" schemata. We derive an exact formal solution of the equations showing how a hierarchical structure in time and degree of coarse-graining emerges, the effect of recombination being to successively form more fine-grained objects from their more coarse-grained building blocks, where in this case the building blocks can come from strings of different lengths. We examine the limit distributions of the dynamics in the case of a flat fitness landscape, one-point homologous crossover and no mutation. By taking advantage of the existence of a set of conserved quantities in the dynamics we provide exact solutions for the cases $N_m = 2,\ 3$ and use these to investigate the phenomenon of inter-length-class allele diffusion. We also study the general case showing what exact results may be easily derived using our particular coarse-grained formulation.

1 Introduction

The dynamics engendered by a "canonical" GA and, indeed, genetic dynamics in general, is exceedingly complicated. This is true even in the case of what one might think of as "toy" fitness landscapes such as counting ones or needle-in-a-haystack. In fact, up until quite recently [1], to our knowledge, no solutions have been found for the dynamics in the presence of recombination for arbitrary string lengths even in the case of a flat fitness landscape, though there has been recent noteworthy progress in the special case of "genepool" recombination [2, 3], where for a given recombination event allele mixing is over the entire population not just between two parents. For binary strings of fixed length, N, the probability distribution that describes the dynamics is obtained by solving 2^N coupled, non-linear difference equations. Important results have been derived about this system of equations by viewing them as a dynamical system [4]. However, these coupled equations, in terms of the underlying string variables, are far removed from traditional elements of GA theory such as the Schema theorem and Building Block Hypothesis (BBH) [5, 6].

The underlying microscopic equations, however, can be rewritten naturally in a basis other than the string basis [7, 8, 9] yielding evolution equations that offer the benefit of a very intuitive interpretation, that illuminate the content of the Schema theorem and the BBH, that naturally coarse grain from string equations to schema equations, that yield an interpolation between the microscopic and the macroscopic and that offer new exact results or simpler proofs of known results. These equations lead to many insights into the dynamics of GAs offering an exact Schema theorem that naturally incorporates a form of the BBH, although it is important to emphasize here that the "building blocks" that naturally emerge in this formulation are dynamic and not necessarily short or even fit! However, creation events due to recombination can be precisely understood in terms of these BBs. Originally applied to a canonical GA (proportional selection, 1-point crossover and mutation) the basic elements have been extended to GAs with arbitrary selection schemes and any homologous crossover [1] and, importantly, have been extended to Genetic Programming (GP) by Poli and coworkers [10, 11].

There has been increasing interest in variable-length representations from different points of view [12, 13, 14] . In this paper we will use a coarse-grained BB formulation to investigate the dynamics of variable length GAs up to a maximum size N_m. We present formal solutions for an homologous crossover operator and arbitrary fitness landscape and mutation showing how the solution naturally admits an interpretation in terms of a hierarchy of BBs. We then consider the asymptotic behaviour of the dynamics for a flat fitness landscape, both at the formal level, discussing generalizations of Geiringer's theorem, and at the explicit level, deriving exact solutions for $N_m = 2$ and 3 and various exact results for arbitrary N_m.

This work is, of course, susceptible to the standard criticisms - what is the relevance of considering a small number of loci and flat fitness landscapes? There are several

ways of rebutting such criticism. Firstly, simple models can lead to intuitive insights that would be less transparent in a more complex model. An important example of that is the minimal two-bit deceptive problem [15]. Another example, is the work of Spears [16] where limit distributions for recombination and mutation for fixed length GAs in a flat fitness landscape were investigated in simple two and three-bit problems. Interestingly, even in this case he had to resort to numerical rather than analytical calculations. Additionally, understanding the structure of the dynamics in simple problems can lead to insight about how to construct results or proofs in more general problems and potentially lead to insights which may be of benefit for practitioners.

2 Coarse-Grained Evolution Equations

In this section we introduce the notion of coarse-grained evolution equations in a BB basis, discussing their interpretation and advantages at a formal level. We will not derive the coarse-grained exact evolution equations here but refer the reader to the original literature [7, 9, 10, 11]. Our interest here is variable-length GAs with homologous crossover. As homologous crossover operators conserve length classes [18] we will consider the corresponding evolution equation for strings or schemata within a given length class N, composed of strings of a fixed length, and consider arbitrary string length $N \leq N_m$, where $N_m \in [1, \infty]$. In this case, if one considers the evolution of length N strings then one of the parents in the crossover operation must be a length N string as well while the other parent may be of arbitrary size. The action of the homologous crossover we will use can be simply understood by aligning the two parents at the first loci then implementing a mask defined on the common region of the two strings. For example, with 1111 and 000000 the common region is associated with the first four loci. A one-point crossover between the second and third loci would yield 110000 and 0011 while a crossover between the fifth and sixth loci (of the second string) is not allowed. Hence, the total number of possible masks on the common region is 2^4.

Our primary object of interest will be the proportion of strings of a given type, c_i^N, $P(c_i^N, t)$, or of a given schema, ξ^N, $P(\xi^N, t) = \sum_{C_i^N \in \xi^N} P(c_i^N, t)$, within a length class N. Thus, we define a schema relative to a given length class. However, it is important to note that all proportions will be relative to the total population size summed over all length classes. In the infinite population limit, which we will generally assume throughout, $P(c_i^N, t)$ is simply the probability for finding the string c_i^N. For a string c_i^N we have

$$P(c_i^N, t+1) = \mathcal{P}(c_i^N) P_c(c_i^N, t) + \sum_{C_i \neq C_j} \mathcal{P}(c_j^N \to c_i^N) P_c(c_j^N, t) \quad (1)$$

where the sum is over all length-class N strings that differ by at least one bit from c_i^N. $\mathcal{P}(c_i^N) = (1-p_m)^N$ is the probability that c_i^N remains unmutated and $\mathcal{P}(c_j^N \to c_i^N) = p_m^{d_H(i,j)}(1-p_m)^{N-d_H(i,j)}$ is the probability that the string c_j^N mutate to the string c_i^N, $d_H(i,j)$ being the Hamming distance between the strings c_i^N and c_j^N. Note that mutation preserves the length class of a string or schema. $P_c(c_i^N, t)$ is the probability of finding a string c_i^N after selection and crossover and is given by

$$P_c(c_i^N, t) = (1-p_c) P'(c_i^N, t) + \sum_{j=1}^{N_m} \sum_{m=0, even}^{2^{min(j,N)}-1} (p_c(m) + p_c(\bar{m})) P'(c_i^j(m), t) P'(c_i^N(\bar{m}), t) \quad (2)$$

where $P'(c_i^N, t)$ is the probability for selecting the string c_i^N. $p_c(m)$ is the probability of implementing the mask m on the common region between the two strings and we sum over only even masks as this ensures that the tail comes from the second parent which, without loss of generality we assume to be of length N, and therefore that length is preserved. $\bar{m}$ is the mask conjugate to m. The total number of possible masks on the common region is $2^{min(j,N)}$. $c_i^j(m)$ for a given mask m represents the part of the string c_i^N inherited from the first parent, which we assume to be of length j, and $c_i^N(\bar{m})$ is that part inherited from the second. Both $c_i^j(m)$ and $c_i^N(\bar{m})$ are schemata. (2) has a form similar to that for the fixed length case and can be interpreted similarly, i.e. strings are created by BBs, the difference in this case being that one of the BBs can come from a parent of other than length N. Once again we emphasize that these BBs are dynamical not static schema averages and are neither necessarily small or even fit!

The microscopic equation (1) can be coarse-grained to an arbitrary schema of order $N_2 \leq N$ and defining length $(l-1)$ contained within strings of size N to find

$$P(\xi^N, t+1) = \mathcal{P}(\xi^N) P_c(\xi^N, t) + \sum_{\not{\xi}_i^N} \mathcal{P}(\not{\xi}_i^N \to \xi^N) P_c(\not{\xi}_i^N, t) \quad (3)$$

where the sum is over all schemata, $\not{\xi}_i^N$, that differ by at least one bit from ξ^N in one of the N_2 defining bits of ξ^N. In other words any schema competing with ξ^N and belonging to the same partition. $\mathcal{P}(\xi^N) = (1-p_m)^{N_2}$ is the probability that ξ^N remains unmutated and $\mathcal{P}(\not{\xi}_i^N \to \xi^N) = p_m^{d_H(\xi^N, \not{\xi}_i^N)}(1-p_m)^{N_2 - d_H(\xi^N, \not{\xi}_i^N)}$ is the probability that the schema $\not{\xi}_i^N$ mutate to the schema ξ^N with $d_H(\xi^N, \not{\xi}_i^N)$ being the Hamming distance between the schemata ξ^N and $\not{\xi}_i^N$. $P_c(\xi^N, t) = \sum_{C_i^N \in \xi^N} P_c(c_i^N, t)$ is the probability of finding a schema ξ^N of length class N after selection and crossover and is given by

$$P_c(\xi^N, t) = (1 - p_c A(\xi, t)) P'(\xi^N, t) + \sum_{j=1}^{N_m} \sum_{m \in \mathcal{M}_r(\xi^N)} (p_c(m) + p_c(\bar{m})) P'(\xi^j(m), t) P'(\xi^N(\bar{m}), t) \quad (4)$$

where $P'(\xi^N, t)$ is the probability for selecting a schema ξ^N from strings of length class N. $\xi^N(m)$ for a given mask m represents the part of the schema ξ^N inherited from the first parent and $\xi^N(\bar{m})$ is that part inherited from the second. Now, $\xi^N(m)$ and $\xi^N(\bar{m})$ are the BBs for the schema ξ^N. Thus, we see that BBs at one level are composed of more primitive (lower order) BBs which in their turn are composed of lower order blocks etc. thus leading to a hierarchical structure. $\mathcal{M}_r$ is the set of crossover masks that end in a 0 that affect ξ^N, i.e. the number of "allele mixing" masks, $N_{\mathcal{M}_r(\xi)}$ is their number. $A(\xi^N, t)$ determines the survival probability of the schema and depends on the properties of the particular schema, such as order and defining length, and, importantly, also depends on the length distribution of the strings and their corresponding fitnesses [18].

As with all coarse grained evolution equations the interpretation of (1) and (2) is very intuitive: (2) tells us how a particular string is selected and survives crossover, or alternatively how it is built up from its BBs. The novel element here compared to standard GAs is that the BBs come from strings of potentially different sizes. (1) then tells us how the string is preserved by mutation or formed by mutation from some other string of the same partition.

We can put the basic equation (1) into a yet more elegant form, the corresponding equation for schemata follows trivially, by introducing a 2^N-dimensional population vector for each length class, $\mathbf{P}^N(t)$, whose elements are $P(c_i^N, t)$, $i = 1, ..., 2^N$. Equation (1) takes the form

$$\mathbf{P}^N(t+1) = \overline{\mathbf{W}}^N \mathbf{P}_c^N(t) \tag{5}$$

where the $N \times N$-dimensional mutation matrix $\overline{\mathbf{W}}^N$ is real, symmetric and time independent and has elements $\overline{\mathbf{W}}^N_{ij} = p_m^{d^H(i,j)}(1-p_m)^{N-d^H(i,j)}$. For selection schemes linear in $P(c_i^N, t)$, $\mathbf{P}_c^N(t)$ can be written as

$$\mathbf{P}_c^N(t) = \overline{\mathbf{F}}^N(t)\mathbf{P}^N(t) + \sum_{j=1}^{N_m} \sum_{m=0,even}^{2^{min(j,N)}-1} (p_c(m) + p_c(\bar{m}))\mathbf{J}^j(m,t) \tag{6}$$

where the "cloning" matrix, $\overline{\mathbf{F}}^N(t)$, is diagonal and describes both selection and survival under crossover. Explicitly, for proportional selection $\overline{\mathbf{F}}^N{}_{ii}(t) = (f(c_i)/\bar{f}(t))(1-p_c)$. Finally, the components of the "source" vector are given by $J^j_{C_i^N}(m,t) = P'(c_i^j(m),t)P'(c_i^N(\bar{m}),t)$ which corresponds to the BB sources, from strings of length j and N respectively, for the string c_i^N. Defining the cloning-mutation matrix $\overline{\mathbf{W}}_s^N(t) = \overline{\mathbf{W}}^N\overline{\mathbf{F}}^N(t)$ we have

$$\mathbf{P}^N(t+1) = \overline{\mathbf{W}}_s^N(t)\mathbf{P}^N(t) + \sum_{j=1}^{N_m} \sum_{m=0,even}^{2^{min(j,N)}-1} (p_c(m) + p_c(\bar{m}))\overline{\mathbf{W}}^N\mathbf{J}^j(m,t) \tag{7}$$

The interpretation of this equation is that $J^j_{C_i^N}(m,t)$ is a source which creates strings c_i^N by bringing BBs from strings of length j and N together. The first term on the right hand side tells us how the strings themselves are propagated, or survive, into the next generation, the destructive effect of crossover renormalizing the fitness of the strings. Note that the equation is linear but for the presence of string creation. It is this division into a linear term and a source that allows for a natural formal solution which leads to further insight into the nature of GA dynamics while at the same time offering the possibility of exact, analytic calculations in certain circumstances.

Needless to say solutions of these dynamical equations are hard to come by. They represent, for binary alleles, $2(2^{N_m}-1)$ coupled non-linear difference equations, or in the continuous time limit - differential equations. Here, we consider the formal solution for the case of homologous crossover and mutation and for any selection scheme linear in $P(c_i^N, t)$. The equation (7) is always of the same form, i.e. a first order, linear, inhomogeneous difference (differential) equation. Its iterated solution is

$$\mathbf{P}^N(t) = \mathbf{D}(t,0)\mathbf{P}^N(0) + \sum_{j=1}^{N_m} \sum_{m=0,even}^{2^{min(j,N)}-1} (p_c(m) + p_c(\bar{m})) \sum_{n=0}^{t-1} \mathbf{D}(t,n)\overline{\mathbf{W}}^N\mathbf{J}^j(m,n) \tag{8}$$

where $\mathbf{D}(t,0) = \prod_{n=0}^{t-1} \overline{\mathbf{W}}_s^N(n)$. The interpretation of (8) follows naturally from that of (7). Considering first the case without mutation, the first term on the right hand side gives us the probability that a string survives from $t = 0$ to t without being destroyed by crossover. In other words $\mathbf{D}(t,0)$ is the Greens function or propagator for $\mathbf{P}^N$ [1]. In the case of a flat fitness landscape without mutation for instance $D_{ij}(t,0) = (1-p_c)^t\delta_{ij}$. In the second term, each element, $\mathbf{J}^j(m,n)$, is associated with the creation of a string c_i^N at time n via the juxtaposition of two BBs from strings of length j and N respectively and associated with a mask m. The component corresponding to c_i^N of the matrix $\mathbf{D}(t,n) = \prod_{i=n}^{t-1} \overline{\mathbf{W}}_s(i)$ is the probability that the resultant string survives from its creation at time n to t. The sum over masks, string lengths, j, and n is simply the sum over all possible creation events in the dynamics. In a more explicit notation we will denote the propagator for a string $h_1 \cdots h_N$ by $D_{h_1 \cdots h_N}(t,t')$.

This formal solution above has a very natural diagramatic interpretation both at the level of fixed length strings which can be extended to the present case.

3 Geiringer's Theorem

For any dynamical system fixed points and their stability are of particular interest. Hence, in this section we will discuss the fixed point distributions for fixed and

variable-length GAs. For a fixed-length GA evolving on a flat landscape in the absence of mutation the fixed point $P^*(h_1...h_N)$ of the dynamics for a string $c_i = h_1...h_N$ is

$$P^*(h_1...h_N) = \lim_{t\to\infty} P(c_i,t) = \prod_{i=1}^{N} P(*^{i-1}h_i*^{N-i},0) \quad (9)$$

where $*^i$ as a string argument means the symbol $*$ repeated i times. This result is the well known Geiringer's theorem [17] for a general crossover operator. Any population that factorizes in this manner is said to be in linkage equilbrium and the resulting allele frequencies are known as Robbins proportions. This result emerges naturally from equation (8), specialized to the case of a single length class, N, which yields for a flat landscape in the absence of mutation

$$\mathbf{P}^N(t) = (1-p_c)^t\mathbf{P}(0) + \sum_{m=0,even}^{2^N-1}(p_c(m)+p_c(\bar{m}))\sum_{n=0}^{t-1}(1-p_c)^{t-n-1}\mathbf{J}^N(m,n) \quad (10)$$

As $\lim_{t\to\infty}(1-p_c)^t = 0$, hence $\mathbf{P}^N(t) \to 0$ as $t\to\infty$ unless the summation over time leads to a cancellation of this damping factor. Given that the BB constituents of $\mathbf{J}^N(m,n)$ are associated with damping factors $(1 - p_c\frac{N_{\mathcal{M}_r}(C_i^j(m))}{N_{\mathcal{M}}})^t$ and $(1-p_c\frac{N_{\mathcal{M}_r}(C_i^N(\bar{m}))}{N_{\mathcal{M}}})^t$, where $N_{\mathcal{M}}$ is the total number of non-zero crossover masks, this can only occur if there is no damping of the consituent BBs and this only happens if they are 1-schemata as then $N_{\mathcal{M}_r} = 0$. Thus, the only term that survives in the hierarchical solution of (8) is the product of 1-schemata [9].

The type of recombination employed controls how fast the transient corrections to the limit distribution die out. The damping is controlled by $N_{\mathcal{M}_r}(\xi)$, hence the bigger it is the faster the corresponding transient dies out [1]).

The general approach to equilibrium is characterized by the exponential decay of linkage disequilibrium functions $\Delta_{h_1\cdots h_N} = \langle(h_1 - \langle h_1\rangle)\cdots(h_N - \langle h_N\rangle)\rangle$ where $\langle O\rangle$ denotes the population average of O. Thus, $\langle h_i\rangle = P(*^{i-1}h_i*^{N-i})$. These linkage disequilbrium functions will be seen to be natural variables in which to understand the dynamics and approach to equilbrium. In GAs a set of variables that have also been viewed as natural for considering the dynamics are "building blocks".

The generalization of Geiringer's theorem to the variable length case has recently been derived [18]

$$P^*(h_1\cdots h_N) = P(*^N)\prod_{i=1}^{N}\frac{P(*^{i-1}h_i\#,0)}{P(*^i\#,0)}, \quad (11)$$

where

$$P(*^{i-1}h_i\#,0) = \sum_{N\geq 0} P(*^{i-1}h_i*^N,0)$$

and

$$P(*^i\#,0) = \sum_{N\geq 0} P(*^{i+N},0).$$

Here, we see a generalization of the concept of Robbins proportions, the corresponding proportions in the variable length case being $\frac{P(*^{i-1}h_i\#,0)}{P(*^i\#,0)}$. We will see in the next section that there are natural analogs of the linkage disequilibrium functions as well.

4 Explicit Solutions - $N_m = 2,\ 3$

In [1] it was shown for fixed length strings in the continuous time limit how an exact explicit solution corresponding to (8) could be found for a flat fitness landscape. Even in this case however, the result is highly non-trivial due to the complicated combinatorics of the various BB creation events. In the case of variable length strings one would expect the combinatorics to be even more complicated. Before considering the general case we will therefore look at some relatively simple cases for $N_m = 2,\ 3$ with no mutation and using one-point crossover where we also include crossover before the first bit and immediately after the last bit of the shortest parent. For $N_m = 2$ we must solve:

$$P(h_1h_2,t+1) = (1-p_c)P(h_1h_2,t) + \sum_{j=1}^{2}\frac{p_c}{min(2,j)+1}\sum_{i=0}^{min(2,j)}P(h_1...h_i*^{j-i},t)P(*^ih_{i+1}...h_2,t) \quad (12)$$

for strings of length two and

$$P(h_1,t+1) = (1-p_c)P(h_1,t) + \frac{p_c}{2}\sum_{j=1}^{2}\sum_{i=0}^{1}P(h_1...h_i*^{j-i},t)P(*^ih_{i+1}...h_1,t) \quad (13)$$

for strings of length one. The corresponding "source" terms are respectively

$$J^j_{h_1h_2}(i,t) = P(h_1...h_i*^{j-i},t)P(*^ih_{i+1}...h_2,t) \quad (14)$$

$$J^j_{h_1}(i,t) = P(h_1...h_i*^{j-i},t)P(*^ih_{i+1}...h_1,t). \quad (15)$$

The explicit forms of the equations of motion are

$$P(h_1h_2,t+1) = (1-p_cA(h_1h_2))P(h_1h_2,t) + \frac{p_c}{2}P(h_1,t)P(*h_2,t) + \frac{p_c}{3}P(h_1*,t)P(*h_2,t) \quad (16)$$

where $A(h_1h_2) = \left(\frac{1}{2}P(*^1) + \frac{1}{3}P(*^2)\right)$ and

$$P(h_1,t+1) = (1-p_cA(h_1))P(h_1,t) + \frac{p_c}{2}P(*^1)P(h_1*,t) \quad (17)$$

where $A(h_1) = P(*^2)/2$. $P(*^1)$ and $P(*^2)$ are the probabilities to get any string of length one and length two respectively. Note that homologous crossover preserves the length distribution [18].

With this simple $N_m = 2$ problem equations (16) and (17) have an intuitive interpretation that allows us immediately to investigate the phenomenon of allele diffusion between different length classes that is an important characteristic of variable-length genetic dynamics. The factor $P_s(h_1 \cdots h_N) = (1 - p_c A(h_1 \cdots h_N))$ describes the survival probability per generation of a particular length-N string. For length-one strings $P_s(h_1) = (1-p_c P(*^2)/2)$ so it is only in the presence of length-two strings that that there is a non-zero decay probability. This probability grows as a function of $P(*^2)$ due to the fact that there are more decay channels open to the string. For length-one strings the only creation source is via the 2-schema h_1* which implies a diffusion of alleles of type h_1 from length-two to length-one strings. For length-two strings the two corresponding creation terms are associated with getting the first bit of the string from a parent of length one and the second bit from a 1-schema associated with strings of length two and the first and second bits from 1-schemata associated with strings of length two. This second term is exactly the same as would be found in a fixed-length GA. The novel element is to be able to construct the desired length-two string by interaction between a 1-schema associated with length-two strings and a length-one string. Thus, in order to solve for the dynamics for length-two strings one must first solve for the dynamics of the size one strings. As from (17) one can see that their dynamics depends on the dynamics of the 1-schemata it would seem that the dynamics of the length-one and two strings are inextricably interwined and must be solved for simultaneously. However, this is not so. The reason why not is that there exist constants of the motion that can be exploited. To see this consider $P(h_1\#, t) = P(h_1, t) + P(h_1*, t)$. The 1-schema probability $P(h_1*, t)$ may be determined from (16)

$$P(h_1*, t+1) = (1 - p_c A(h_1 h_2)) P(h_1*, t) + \frac{p_c}{2} P(h_1, t) P(*^2) + \frac{p_c}{3} P(h_1*, t) P(*^2, t) \quad (18)$$

thus adding this to (17) one finds

$$P(h_1\#, t+1) = P(h_1\#, t) \quad (19)$$

and hence $P(h_1\#)$ is an invariant of the motion. It basically expresses the conservation of the allele h_1 associated with the first bit position and in this sense is analogous to the conservation law $P(*^{k-1} h_k *^{N-k}, t) = P(*^{k-1} h_k *^{N-k}, 0)$ for any k associated with fixed length GAs. In the variable-length case however there is no conservation of alleles within a given length class due to the phenomenon of inter-length-class allele diffusion. With this conservation law in hand the equations (17) and (16) can be decoupled. We write (17) as

$$P(h_1, t+1) = D_{h_1} P(h_1, t) + \frac{p_c}{2} P(*^1) P(h_1\#, t) \quad (20)$$

where we now revert to the propagator notation used in section 2, $D_{h_1} = (1 - p_c/2)$ being the survival probability per generation. This equation can be simply solved using equation (8) to yield

$$P(h_1, t) = D^t_{h_1} P(h_1, 0) + (1 - (1 - D^t_{h_1})) P^*(h_1) \quad (21)$$

where $P^*(h_1) = P(*^1) P(h_1\#)/P(*\#)$ is the fixed point of the dynamics in agreement with the general fixed point of (11). We may expand $P(h_1\#) = P(h_1, 0) + P(h_1*, 0)$ to find

$$P(h_1, t) = \left((1 - \frac{p_c}{2})^t + (1 - (1 - \frac{p_c}{2})^t) P(*^1) \right) P(h_1, 0) + (1 - (1 - \frac{p_c}{2})^t) P(*^1) P(h_1*, 0) \quad (22)$$

Note that even if $P(h_1, t) = 0$ inter-length-class allele diffusion will generate alleles h_1 in length-one strings at some later time. Thus, unlike the fixed length case a particular allele in a given length class may be regenerated without the intervention of mutation. Note that at the fixed point the contributions to h_1 are determined solely by the $t = 0$ proportions of this allele from all possible length classes. Hence, recombination in the variable length case maximally mixes the alleles among all available length classes.

Having found the exact solution for strings of length one we may proceed to strings of length two. As can be seen from equation (16) we need to solve first for the dynamics of the two 1-schemata h_1* and $*h_2$. From (16), one notices that there are no source terms for $*h_2$ from length-one strings. Hence, one finds that

$$P(*h_2, t+1) = P(*h_2, t) \quad (23)$$

and notes that the allele h_2 is conserved in agreement with (11). The 1-schema $P(h_1*, t) = P(h_1\#) - P(h_1, t)$ can be simply solved for to yield

$$P(h_1*, t) = D^t_{h_1*} P(h_1*, 0) + (1 - D^t_{h_1*}) P^*(h_1*) \quad (24)$$

where the survival probability per generation for h_1* is $D_{h_1*} = (1 - \frac{p_c}{2})$ and the fixed point $P^*(h_1*)$ is given by $P^*(h_1*) = P(*^2) P(h_1\#)/P(*\#)$ once again in agreement with equation (11). Note that the exponential approach to this fixed point is the same as for $P(h_1, t)$.

Finally, using the explicit solutions (21), (23) and (24) we may deduce the solution of (16). $P(h_1*, t)$ and $P(h_1, t)$ are a time-dependent source of strings $P(h_1 h_2, t)$. Substituting in (16) the solutions (21), (23) and (24) one finds

$$P(h_1 h_2, t) = D^t_{h_1 h_2} (P(h_1 h_2, 0) - P(h_1\#) P(*h_2, 0)) + \frac{P(*h_2, 0)}{P(*^2)} (P(h_1, 0) - P(*^1) P(h_1\#)) (D^t_{h_1 h_2} - D^t_{h_1}) + P(h_1\#) P(*h_2, 0) \quad (25)$$

In the limit $t \to \infty$ $D_{C_i^N} \to 0$; thus, we see the fixed point $P^*(h_1h_2) = P(h_1\#)P(*h_2, 0)$ emerging in agreement with equation (11).

The solutions can be put into a more elegant and transparent form by introducing the notion of generalized linkage disequilibrium functions. We define $\Delta_{h_1}(t) = (P(h_1, t) - P(*^1)P(h_1\#))$ and $\Delta_{h_1h_2}(t) = (P(h_1h_2, t) - P(h_1\#)P(*h_2))$. Thus, both these functions characterize deviations from the corresponding fixed points. Immediately we see an important distinction from the fixed length case where a single bit cannot have BBs and linkage occurs between different bits. Here the "building blocks" of h_1 are any length-one string and any string of any length that contains h_1. Due to the phenomenon of inter-length-class allele diffusion there is a concept of linkage disequilibrium for a single bit. This is due to the fact that linkage disequilibrium can be generalized to take into account correlation between corresponding bits in different length classes. Similarly, for h_1h_2 the BBs are the length class two schema $*h_2$ and any string of any length that contains h_1. In both cases we see that one of the BBs is associated with a coarse graining over all possible length classes and hence is not a schema associated with a fixed length class. Explicitly,

$$P(h_1, t) = D^t_{h_1}\Delta_{h_1} + P^*(h_1) \tag{26}$$

and

$$P(h_1h_2, t) = D^t_{h_1h_2}(\Delta_{h_1h_2} + \frac{P(*h_2)}{P(*^2)}\Delta_{h_1}) - D^t_{h_1}\frac{P(*h_2)}{P(*^2)}\Delta_{h_1} + P^*(h_1h_2) \tag{27}$$

We now consider the solution for strings of length $N \leq 3$. For $N_m = 3$ we have

$$P(h_1h_2h_3, t+1) = (1 - p_cA(h_1h_2h_3))P(h_1h_2h_3, t) + \frac{p_c}{2}P(h_1, t)P(*h_2h_3, t) + \frac{p_c}{3}(P(h_1*, t)P(*h_2h_3, t) + P(h_1h_2, t)P(**h_3, t)) + \frac{p_c}{4}(P(h_1**, t)P(*h_2h_3, t)) + P(h_1h_2*, t)P(**h_3, t) \tag{28}$$

where $A(h_1h_2h_3) = (P(*^1)/2 + 2P(*^2)/3 + P(*^3)/2)$. Once again this is a linear equation in $P(h_1h_2h_3, t)$ but with sources for which we have to solve equations for length one and two strings and 1-schemata from two strings and 1- and 2-schemata from length-three strings. Analogously to the case $N_m = 2$ length-one strings satisfy an equation that is coupled to 1-schemata of different length, in this case $P(h_1*, t)$ and $P(h_1**, t)$. However, as in the length-two case using the conservation law $P(h_1\#, t) = P(h_1, t) + P(h_1*, t) + P(h_1h_2*, t) = P(h_1\#, 0)$ allows us to write the equation as

$$P(h_1, t+1) = D_{h_1}P(h_1, t) + \frac{p_c}{2}P(*^1)P(h_1\#, t) \tag{29}$$

The solution and associated fixed point are given by (26) as in the case $N_m = 2$ above. Length-two strings satisfy

$$P(h_1h_2, t+1) = (1 - p_cA(h_1h_2))P(h_1h_2, t) + \frac{p_c}{2}P(h_1, t)P(*h_2, t) + \frac{p_c}{3}P(h_1*, t)P(*h_2, t) + \frac{p_c}{3}P(h_1**, t)P(*h_2, t) + \frac{p_c}{3}P(h_1h_2*, t)P(*^2) \tag{30}$$

Thus we see a coupling to length-one and length-three sources. The 1-schemata equations for $P(h_1*, t)$ and $P(*h_2, t)$ however can be solved by eliminating length-three sources using the conservation law $P(*h_2\#, t) = P(*h_2, t) + P(*h_2*, t) = P(*h_2\#, 0)$. One obtains

$$P(h_1*, t) = D^t_{h_1*}(\Delta_{h_1*} + \frac{P(*^2)}{P(*\#)}\Delta_{h_1}) - \frac{P(*^2)}{P(*\#)}\Delta_{h_1} + P^*_{h_1*} \tag{31}$$

where Δ_{h_1*}, Δ_{h_1} and $P^*_{h_1*}$ are as above in the $N_m = 2$ case. To solve (28) we still require $P(h_1**, t)$, $P(*h_2*, t)$, $P(**h_3, t)$, $P(*h_2h_3, t)$ and $P(h_1h_2*, t)$. $P(**h_3, t)$ is conserved as the final bit of the longest string cannot mix with anything else and therefore is unaffected by inter-length-class allele diffusion. $P(*h_2*, t)$ can be solved for in terms of the solution of $P(*h_2, t)$. $P(h_1**, t)$ obeys

$$P(h_1**, t+1) = (1 - \frac{p_c}{2}P(*^1) - \frac{2p_c}{3}P(*^2))P(h_1**, t) + \frac{p_c}{2}P(*^3)P(h_1, t) + \frac{2p_c}{3}P(*^3)P(h_1*, t) \tag{32}$$

As we already have the solution for $P(h_1*, t)$ and $P(h_1, t)$ this can simply be solved for. $P(*h_2h_3, t)$ satisfies

$$P(*h_2h_3, t+1) = (1 - \frac{p_c}{3}P(*^2) - \frac{p_c}{4}P(*^3))P(*h_2h_3, t) + \frac{p_c}{3}P(**h_3)P(*h_2, t) + \frac{p_c}{4}P(**h_3)P(*h_2*, t) \tag{33}$$

Once again, given that we have the solutions for $P(*h_2, t)$ and $P(*h_2*, t)$ this can be simply solved. Finally, $P(h_1h_2*, t)$ satisifes

$$P(h_1h_2*, t+1) = (1 - \frac{p_c}{2}P(*^1) - \frac{2p_c}{3}P(*^2) - \frac{p_c}{4}P(*^3))P(h_1h_2*, t) + \frac{p_c}{3}P(*^3)P(h_1h_2, t) + \frac{p_c}{2}P(*h_2*, t)(P(h_1, t) + \frac{2}{3}P(h_1*, t) + \frac{1}{2}P(h_1**, t)) \tag{34}$$

This is the only non-trivial equation left to solve as it is coupled to $P(h_1h_2, t)$. Both equations are first order linear inhomogeneous difference equations and can be decoupled

by going to a second order linear inhomogeneous difference equation which can be readily solved. Due to length constraints we will present the results elsewhere. With these solutions in hand $P(h_1h_2h_3,t)$ may readily be solved for.

It is worth taking stock of what we have done here. In the case $N_m = 2$, in terms of the underlying string variables, there are six coupled equations to be solved. By going to a coarse-grained schema, or BB basis, one is able to implement the conservation laws most naturally, thereby decoupling the equations and finding an exact, explicit solution. For $N_m = 3$ there are fourteen coupled equations. The only extra complication relative to the $N_m = 2$ case however was the fact that after implementing the conservation laws two equations remained non-trivially coupled and had to be decoupled by going to a higher order difference equation.

5 Explicit Solutions - N_m arbitrary

In this section we wish to make some observations about the general case - N_m arbitrary. An important element, seen in the last section, is the existence of conservation laws which may be used to facilitate the solution of the dynamics. Generally, the conserved quantities are

$$P(*^{i-1}h_i\#,t) = P(*^{i-1}h_i\#,0) \tag{35}$$

of which there are N_m. Hence, from the dynamical equations one may eliminate N_m variables. As in the above cases of $N_m = 2,\ 3$ one may use this fact to obtain the exact dynamics of certain schemata. These conservation laws are more naturally expressed in terms of schemata rather than strings. For instance, the conservation law $P(1\#,t) = \text{constant}$ in terms of string variables is $P(1,t)+P(11,t)+P(10,t)+P(100,t)+P(101,t)+P(110,t)+P(111,t) = \text{constant}$. This is a difficult constraint to implement at the level of the string equations themselves.

As we have emphasized, with the coarse-grained BB approach advocated here dynamical solutions are built up hierarchically beginning with low order BBs and proceeding to higher ones. As the lowest order ones are 1-schemata it is of interest to investigate the general equation for a 1-schema from length class N. One finds that

$$P(*^{i-1}h_i*^{N-i},t+1) = A_1P(*^{i-1}h_i*^{N-i},t) + A_2\sum_{j\geq i;j\neq N}A_3(j)P(*^{i-1}h_i*^{j-i},t) \tag{36}$$

where

$$A_1 = i\left(\sum_{j>N}\frac{P(*^j)}{N+1}+\sum_{j=i}^{N}\frac{P(*^j)}{j+1}\right)+\sum_{j=1}^{i-1}P(*^j) + P(*^N)\left(\frac{N-i+1}{N+1}\right)$$

$$A_2 = P(*^N)$$

$$A_3 = \left(1-\delta(j>N)\frac{i}{N+1}-\delta(k\leq j\leq N)\frac{i}{j+1}\right)$$

Note that 1-schemata from other than length-class N strings act as sources for h_i, however, there are no more "primitive", i.e. lower order, sources. Hence, in the sense of section 2 this equation is really homogeneous with no BB sources and hence can be written as

$$\mathbf{P}(t+1) = \mathbf{A}\mathbf{P}(t) \tag{37}$$

where the elements of the matrix $\mathbf{A}$ can be read off from (36) and the values of the coefficients A_1, A_2 and A_3. The diagonalization of this matrix yields the decay rates of the various 1-schemata. With the 1-schemata solution in hand we may start to reconstruct the 2-schemata respecting the hierarchical structure outlined in section 2. We will not pursue this further in this paper restricting attention to some more specific results.

From (35) one immediately sees that the quantity $P(*^{N_m-1}h_{N_m},t)$ is conserved. Additionally, for the length-one strings all "sources" $P(h_1*^{j-1})$ for $P(h_1,t)$ appear with the same coefficient, $p_c/2$. Hence, $P(h_1,t)$ satisfies (26) the only difference now being that $P(h_1\#) = \sum_{j=2}^{N_m}P(h_1*^{j-1},t)$.

Using the conservation of the last bit of the longest string one may also determine the evolution of the last bit of the next longest string and the last bit of the string of length $N = N_m - 1$ by using the conservation law $P(*^{N_m-2}h_{N_m-1}\#) = \text{constant}$. For the next to last bit of the longest string the solution is

$$P(*^{N_m-2}h_{N_m-1}*,t) = D^t_{*^{N_m-2}h_{N_m-1}*}\Delta_{*^{N_m-2}h_{N_m-1}*}+P^*_{*^{N_m-2}h_{N_m-1}*} \tag{38}$$

where $D_{*^{N_m-2}h_{N_m-1}*} = (1-(1/N_m)(P(*^{N_m-1})+P(*^{N_m})))$ and $P^*_{*^{N_m-2}h_{N_m-1}*} = P(*^{N_m})P(*^{N_m-2}h_{N_m-1}\#)/P(*^{N_m-1}\#)$ which is the expected fixed point from (11).

6 Conclusions

We have investigated the dynamics of variable-length GAs using a coarse-grained BB representation of the dynamical equations. We showed that the formal solution of the equations could be interpreted in an analogous manner to that of the fixed length case, i.e. the hierarchical construction of more fine-grained schemata from their more coarse-grained BBs. The novel element here is that these BBs could come from strings of different lengths. We discussed briefly the fixed point distribution of the equations for a flat fitness landscape using a one-point homologous crossover operator and no mutation showing how a generalization of

Robbins proportions emerged that involved a generalized notion of a BB. We then turned to a more explicit construction of the entire dynamics and quantified the approach to the fixed point. For $N_m = 2,\ 3$ we were able to find explicit solutions utilizing the existence of conservation laws for certain quantities. This in itself shows the utility of the coarse grained BB representation, the $N_m = 3$ problem at the string level corresponding to 14 simultaneous first order difference equations which need to be solved.

From the resultant solutions we were able to investigate the phenomenon of inter-length-class allele diffusion. We saw that the diffusion rates, or mixing times, for different alleles or combination of alleles depended strongly on the length distribution of strings, which in the case of a flat fitness landscape is time independent. For instance, the diffusion rate for the allele h_1 in length-class-three strings is slower than that of the same allele in length-class-two or one strings if $P(*^1) + (4/3)P(*^2) > 1$ which is the case if the proportion of length-three strings is small. We also can see that the closer the string bit to the beginning of the string then typically the faster it mixes, simply because there are more things with which it can mix. In this sense in the variable length case the degree of exploration versus expolitation carried out by recombination is inhomogeneous depending on the bit's position in the string and the distribution of lengths, diversity being encouraged more at the beginning of strings than at the end. Another interesting aspect of inter-length-class allele diffusion is the fact that for a given length class a lost allele from a particular bit position can be recovered if the allele exists in the corresponding bit of another length class string.

Acknowledgements

CRS would like to thank the University of Birmingham for a visiting Professorship and DGAPA-PAPIIT grant IN100201. RP and CRS would like to thank the Royal Society and the University of Essex for their support. Alden Wright did this work while visiting the University of Birmingham, supported by EPSRC grant GR/R47394.

References

[1] Stephens, C.R. (2001) "Some Exact Results from a Coarse Grained Formulation of Genetic Dynamics". In L. Spector *et al* eds. *Proceedings of GECCO 2001*, 631-638 (Morgan Kaufmann, San Francisco).

[2] Wright, Rowe, J., Poli, R. and Stephens, C.R. (2002) "A Fixed Point Analysis of a Genepool GA with Mutation", accepted for publication (full paper) in *GECCO 2002*.

[3] Mähnig, T. and Mühlenbein, H. (2001) "Optimal Mutation Rate Using Bayesian Priors for Estimation of Dsitribution Algorithms", in *Stochastic Algorithms: Foundations and Applications*, ed. K. Steinhüfel, LNCS Springer-Verlag.

[4] Vose, M.D. (1999) *The Simple Genetic Algorithm: Foundations and Theory*, (MIT Press, Cambridge MA).

[5] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems* (MIT Press, Cambridge, MA).

[6] Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison Wesley, MA).

[7] Stephens, C.R. and Waelbroeck, H. (1997), "Effective Degrees of Freedom in Genetic Algorithms and the Block Hypothesis", *Proceedings of the ICGA7*, ed. T. Bäck, 34-41 (Morgan Kaufmann, San Mateo).

[8] Stephens, C.R. and Waelbroeck, H. (1998) "Analysis of the Effective Degrees of Freedom in Genetic Algorithms", *Physical Review* **D57** 3251-3264.

[9] Stephens, C.R. and Waelbroeck, H. (1999) "Schemata Evolution and Building Blocks", *Evol. Comp.* **7(2)** 109-124.

[10] Poli, R. (2000) "Exact Schema theorem and Effective Fitness for GP with one-point crossover", in *Proceedings of GECCO2000*, eds D. Whitley *et al* 469-476 (Morgan Kaufmann).

[11] Poli, R. and McPhee, N.F. (2000) "Exact Schema Theory for GP and Variable-length GAs with Homologous Crossover", *Proceedings of GECCO-2001*, ed. Lee Spector *et al* 104-111 (Morgan Kaufmann, San Mateo).

[12] Nordin, P. (1994) "A Compiling Genetic Programming System that Directly Manipulates the Machine Code", *Advances in Genetic Programming*, ed. K.E. Kinnear Jr., 311-331 (MIT Press).

[13] O' Neil, M. and Ryan, C. (2001) "Grammatical Evolution" *IEEE Transaction on Evolutionary Compuation*, in press.

[14] Wu, A.S. and Banzhaf, W. (1998) "Introduction to the Special Issue: Variable-Length Representation and Noncoding Segments for Evolutionary Algorithms" *Evolutionary Computation* **6(4)**, iii-iv.

[15] Goldberg, D.E. (1987) *"Simple Genetic Algorithms and the Minimal, Deceptive Problem"*, in *Genetic Algorithms and Simulated Annealing*, ed. L. Davis, 74-88 (Pitman, London).

[16] Spears, W.M. (2000) "Limiting distributions for mutation and recombination", in *Proceedings of FOGA 6*, eds. W.M. Spears and W. Martin, (Morgan Kaufmann, San Mateo).

[17] Geiringer, H. (1944) "On the Probability Theory of Linkage in Mendelian Heredity", *Annals of Mathematical Statistics* **15**, 25-27.

[18] Poli, R., Rowe, J., Stephens, C.R. and Wright, A. (2002) "On the Search Biases of Homologous Crossover in Linear Genetic Programming and Variable-length Genetic Algorithms", accepted for publication (full paper) in *GECCO 2002*; University of Essex Computer Science Technical Report TRCSM-352.

Strategy Parameter Variety in Self-Adaptation of Mutation Rates

Christopher Stone
Intelligent Computer Systems Centre
University of the West of England
Bristol, United Kingdom
christopher.stone@uwe.ac.uk

Jim Smith
Intelligent Computer Systems Centre
University of the West of England
Bristol, United Kingdom
james.smith@uwe.ac.uk

Abstract

Self-adaptation has been widely used in Evolution Strategies (ES) and Evolutionary Programming (EP), where it has proved useful in varying the mutation step size for continuous objective variables. To date, relatively little work has been performed on applying self-adaptation to the canonical Genetic Algorithm (GA). This research applies a simple discrete model of self-adaptation to test functions with differing characteristics. We show that the discrete model is able to provide more reliable problem solving than the classical lognormal self-adaptation scheme on the test problems examined. We find that although self-adaptation parameter choices representing conventional thinking perform best for unimodal functions, very different parameter settings are required for optimum performance on multimodal functions. These results are discussed in terms of the strategy parameter variety needed for self-adaptation to work effectively and we outline a self-adaptation mechanism designed to capitalize on these findings.

1 INTRODUCTION

In a self-adaptive Evolutionary Algorithm (EA), the representation for individuals in the population is extended to include strategy parameter information. The EA operates as normal, evolving the population according to the fitness of its members, with the additional step of stochastically varying the strategy parameters of individuals selected for reproduction. Self-adaptation of mutation rates is possibly the most common application of self-adaptation, largely stemming from its widespread use in ES (Schwefel, 1981) and EP (Fogel, Fogel & Atmar, 1991). For the purposes of self-adaptation, the main difference between GAs and ES/EP is that GAs usually employ a binary representation. With such a representation, a per-bit mutation rate is used to control the rate of bit-flipping mutations applied to an individual. For a non-adaptive GA, this parameter is fixed across the population and throughout the course of a run. However it is natural extension to encode the mutation rate into each individual, to allow it to vary across the population and in time. Bäck (1992) used these ideas and performed seminal work showing that self-adaptation in GAs is possible. Following Bäck's work, several authors have experimented with self-adaptation of mutation rates in GAs (see for example, Bäck & Schütz, 1996; Smith & Fogarty, 1996; Hinterding, 1997). Design decisions that must be addressed with this approach are the choice of representation for the strategy parameter and, related to this, the means by which the strategy parameter is itself varied to allow adaptation to occur. Bäck's early work remained close to the traditional interpretation of a GA and used a binary encoding of the strategy parameters with corresponding bitwise mutation. Current thinking is that a real-valued representation is preferable (Glickman & Sycara, 1998). This then allows the use of lognormal adaptation of strategy parameters as shown in (1) where the τ parameter controls the step size of σ_i, the individual's mutation rate.

$$\sigma_i' = \sigma_i \cdot \exp(\tau \cdot N(0,1)) \qquad (1)$$

Recent empirical (Liang et al. 1998; Glickman & Sycara 2000) and theoretical (Rudolph 1999) work has shown that self-adaptation schemes which adapt too quickly can lead to premature convergence to low step sizes, with the search getting 'stuck' at local optimum. This has lead to an interest in alternative variation schemes.

Smith (2001) introduces a dynamical systems model of a GA with self-adaptation of mutation rates. The model is

Table 1: Evaluation Functions

	Name	Formula	Length (bits)/ Encoding	Optimum	Allowed Generations	
					w/o xover	w/xover
f1	OneMax	$\sum_{i=1}^{l} x_i$	128	128	100	500
f2	Inverted Rastrigin's Function	$\frac{1}{\sum_{i=1}^{l/16}[x_i^2 - 10\cos(2\pi x_i) + 10] + 1}$ $-5.12 < x_i < 5.12$	64 4 dimensions each 16 bits	1	500	500
f3	Deb's Fully Deceptive Function	$\sum_{j=1}^{l/4} \begin{cases} 0.2\cdot(3-b_j), b_j<4 \\ 1, b_j=4 \end{cases}$ where $b_j = \sum_{i=1}^{4} x_i$	32	8	250	100
f4	Matching Bits	$\sum_{i=1}^{l-1} \begin{cases} 1, x_{i+1}=x_i \\ 0, x_{i+1}\neq x_i \end{cases}$	24	23	1000	100
f5	Royal Road R1	$\sum_{j=1}^{l/8} \begin{cases} 0, b_j<8 \\ 8, b_j=8 \end{cases}$ where $b_j = \sum_{i=1}^{8} x_i$	64	64	1000	1000

used to predict mean fitness of an evolving population over time. To make the mathematics computationally tractable, there are two key differences between the model and the self-adaptive GAs just described. Firstly, rather than using a binary or real-valued representation, strategy parameters are represented by a single allele of alphabet q, where q is small. Smith uses a value of 10 and this is also the value used in the present work. A consequence of this is that the mutation rate attached to an individual can only take on one of q possible values, as opposed to the large or effectively infinite number available with binary or real-valued representations. Secondly, because of the discrete nature of the strategy parameter representation, the lognormal scheme in (1) cannot be used to vary the strategy parameters. Although it would be possible to provide some discrete variant of this algorithm, for simplicity Smith uses a scheme that modifies every individual's strategy parameter with probability z and equal likelihood of changing to each of the q possible alleles. Because it is possible for the modified strategy parameter to retain the same allele as the original one, the probability P_a that the strategy parameter is altered is given by:

$$P_a = z \cdot (q-1)/q \qquad (2)$$

Control over the degree of change of the inherited strategy parameter is provided by z, an external parameter known as the *innovation rate*. This provides the variation needed for selection to choose preferable strategy parameters. The conventional view is that a non-uniform distribution is desirable, for example, the lognormal scheme in (1), to generate many small perturbations of strategy parameter value with larger variations being possible, but less likely. This is in order to provide occasional large changes in value to prevent the EA from getting stuck when searching rugged landscapes. As presented, Smith's model represents a considerable departure from this view since it uses a uniform distribution that provides equal opportunity for large or small perturbations[1]. In the present work we examine the implications of this variation scheme and the effects of varying the innovation rate on the performance of the GA. We do not attempt to demonstrate the need for self-adaptation, as this has been done elsewhere (see for example, Stephens et al, 1998).

2 EXPERIMENTAL SETUP

The GA used for the experiments provides a real-valued strategy parameter linked to each individual in the population. For discrete adaptation schemes, the strategy parameter encodes one of ten representative mutation rates in the range *minrate* to *maxrate* for the individual. Each strategy parameter is initialized to a random allele and varied according to (2). For continuous adaptation schemes, the strategy parameter encodes an arbitrary

[1] Although it should be noted that this is not a requirement of the model.

mutation rate for the individual, initialized to a random value in the range *minrate* to *maxrate* and variation of the strategy parameter is performed by the multiplicative lognormal function (1). To provide consistency with the discrete scheme, the resulting mutation rate is limited to the range *minrate* to *maxrate*.

The GA is generational because short-term regression is desirable with self-adaptation to allow adequate learning of strategy parameters (Schwefel, 1997). Selection is performed using either truncation (extinctive) selection or fitness proportionate (preservative) selection sampled using Baker's (1987) SUS algorithm. The differences between extinctive and preservative selection mechanism are discussed in (Bäck & Hoffmeister, 1991). A population of 500 individuals is maintained to reduce the variance of results and to allow adequate sampling of strategy parameters in the mating pool even with high selection pressure. (100,500) truncation selection is used, based on results from ES (Schwefel, 1981). Genetic operators are single point crossover applied with a rate of 0.7 or zero and bitwise mutation applied with the probability given by the individual's strategy parameter. All results presented are the mean of 50 runs unless otherwise noted.

Five functions with a broad range of characteristics were used for these experiments. These are detailed in Table 1.

Experiments were run on each of the functions with innovation rates of 0.01, 0.05 and 0.1 to 1 in steps of 0.1 using two selection pressures: (100,500) truncation selection and fitness proportionate selection. The ten standard mutation rates used for these experiments were 0.0005, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075 and 0.1.

For each set of experiments, two metrics were used to compare results. *Reliability* was measured using the number of times the global optimum was found out of 50 runs. *Time to optimum* was used to measure the ability of the GA to solve the problem. Our stance is that a self-adaptive GA must achieve reliability before time to optimum issues can be addressed. This is particularly important in online applications of self-adaptation, where the luxury of multiple runs is not feasible, for example in process control or robotics applications.

3 GA WITH NO RECOMBINATION

Figure 1 shows the time to optimum of the GA for each function using a high selection pressure. The graph is annotated with reliability data, where the optimum was not found in all 50 runs. It shows that high innovation rates provide the most reliable problem solving. These results can be separated according to whether the function is unimodal or multimodal. For the purposes of this work, we classify f5 (Royal Road) as multimodal, since its fitness landscape consists of a series of peaks connected by ridges[2], which produce similar effects to the local optima of multimodal functions. Under this classification the only unimodal function in the test suite, f1 (OneMax) achieves the best performance with an innovation rate of 0.05. Higher innovation rates tend to impact performance, although not excessively so. In contrast, the multimodal functions show a trend towards both faster and more reliable performance as the innovation rate increases.

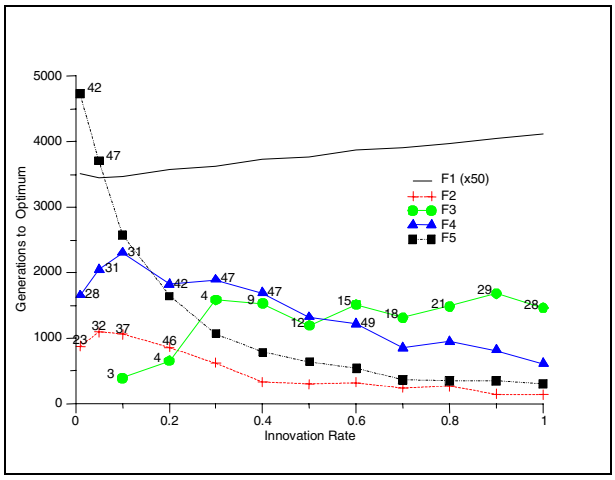


Figure 1 - Generations to Optimum against Innovation Rate for High Selection Pressure

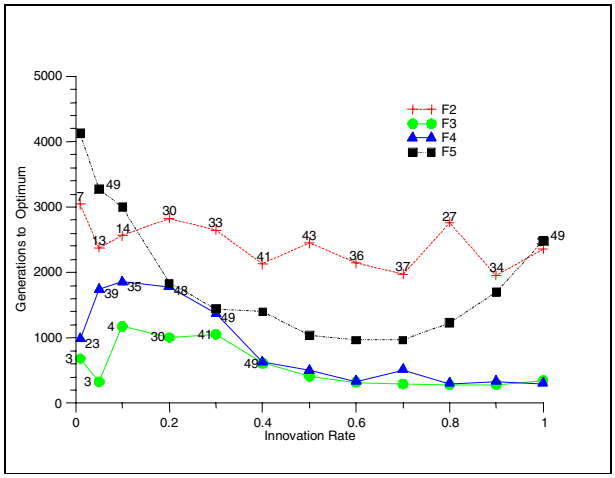


Figure 2 - Generations to Optimum against Innovation Rate for Low Selection Pressure

Results for the same set of experiments performed under low selection pressure are show in Figure 2. Results are

[2] The exact details of the fitness landscape depend on the operators employed.

not shown for f1 (OneMax) since fitness proportionate selection failed to find the optimum in any of the 50 runs. However, the low selection pressure is an advantage for f3 (Deb's Deceptive function) and reliability is much improved over high selection pressures. The graphs are somewhat noisier than their high selection pressure counterparts due to the lower number of runs in which the optimum was found for some functions and innovation rates. In general, reliability is still improved with high innovation rates, with the exception of f2 (Rastrigin's function), but unlike the case with high selection pressure, time to optimum appears to peak at an innovation rate lower than one.

4 GA WITH RECOMBINATION

Experiments on a GA with recombination were performed using an inheritance mechanism, which selects one of the two parental strategy parameters at random with equal probability for the offspring prior to innovation. This choice is discussed further in (Stone, 2001).

Figures 3 and 4 show the effects of innovation on reliability and time to optimum of the GA with crossover. With fitness proportionate selection the optimum for function f1 (OneMax) was never located in any of the 50 runs, so no results are shown for these experiments. With high selection pressure, reliability tends to improve as innovation rate increases whilst time to optimum remains relatively flat due to the effects of crossover. With low selection pressure, it seems that high innovation rates although still providing reliable operation are generally detrimental to time to optimum.

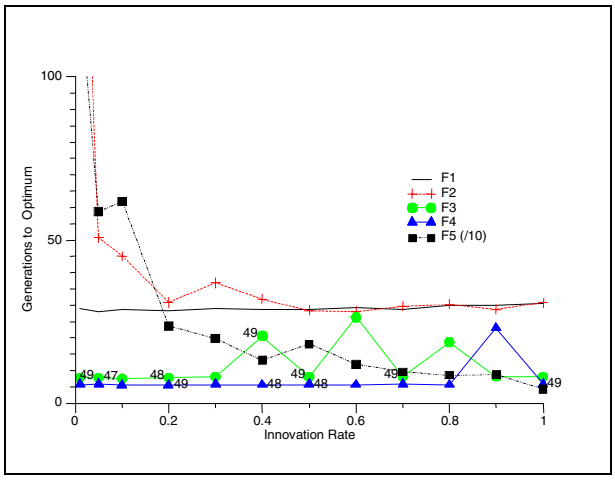


Figure 3 - Generations to Optimum against Innovation Rate for Random Inheritance with High Selection Pressure

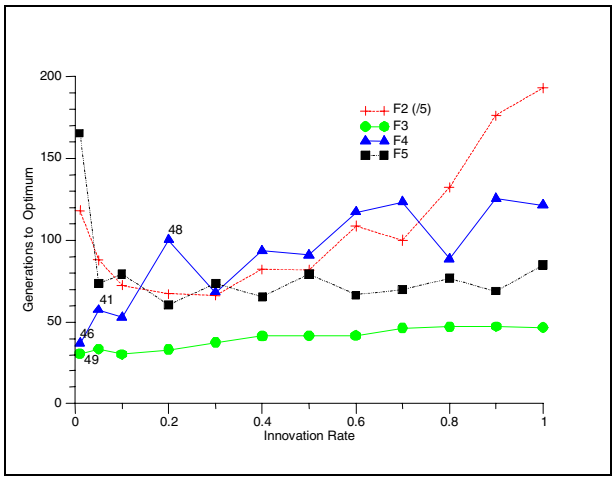


Figure 4 - Generations to Optimum against Innovation Rate for Random Inheritance with Low Selection Pressure

5 COMPARISON OF DISCRETE INNOVATION TO CONTINUOUS INNOVATION

To compare the performance of the discrete innovation scheme with the more mainstream continuous innovation scheme we used various settings for the continuous innovation scheme's τ parameter. The ES literature suggests the following formula for setting τ:

$$\tau = c / \sqrt{n} \qquad (3)$$

where n is the number of objective variables and c is a problem-specific constant. However, this rule of thumb has not, to our knowledge, been exhaustively tested in a GA environment. In ES, n is the number of objective variables in the representation, whereas in the case of a GA it is the length of the string in bits. It is not apparent that the rule can be directly mapped from ES to GA representations. Glickman & Sycara (1998) use a value of 0.1 for τ on a string of length 1000. This corresponds to a value of c of 3.16. In contrast, Hinterding, Michalewicz & Peachey (1996) use a fixed value of 0.013 in their self-adaptive GA. In the absence of other information, we tried values for c of 0.5, 1, 2, 3 and fixed rates of 0.013 and 0.02. For the former values the problem length (in bits) is taken into account when arriving at the actual rates used, whereas the latter two are fixed rates across all functions.

Table 2: Comparison of Best Results for Discrete and Continuous Self-Adaptation

	Discrete Self-Adaptation			Continuous Self-Adaptation			
	Time to Optimum	Successful Runs	z	Time to Optimum	Successful Runs	τ	c
f1 (OneMax)	68	50	0.05	66	50	0.063[1]	2.00
f2 (Rastrigin's)	137	50	1	84	34	0.013	0.10
f3 (Deb's Deceptive)	1680	29	0.9	20	8	0.020	0.11
f4 (Matching Bits)	608	50	1	231	40	0.020	0.10
f5 (Royal Road)	298	50	1	159	50	0.013	0.10

We ran the same set of mutation-only experiments as performed for the discrete scheme, using (100,500) truncation selection and limiting each run to the same number of generations as before (see Table 1). Table 2 compares the best result for each function from the discrete and continuous schemes. Here, best is defined as the result showing most reliability followed by smallest time to optimum for the function. The discrete scheme provides reliable results and acceptable time to optimum for all functions, whereas the continuous scheme, although apparently capable of providing superior time to optimum, displays poorer reliability. It is also interesting that the best results for all of the multimodal problems arise with a value of c of approximately 0.1. This suggests that the relationship in (3) may also hold for GAs with a bitstring representation. However, the range of problem lengths, l, tested is quite restricted. Further work is needed to determine if this pattern extends to other multimodal problems and values of l.

6 DISCUSSION

6.1 INTERPRETATION OF RESULTS

Results for the discrete model suggest that with a high selection pressure, a low innovation rate is appropriate for unimodal problems, whereas an innovation rate of one gives the best results for multimodal problems. The former represents conventional thinking whereas the latter result is novel and requires some explanation. For low selection pressures it appears that the optimal innovation rate to achieve reliability is lower and the best performance is worse than for high selection pressure. We conducted ANOVA analysis which showed that the effects of innovation rates were a statistically significant factor in the time to optimum, and post-hoc tests with a variety of measures confirmed the difference in performance between high and low innovation rates.

In the following discussions, we assume a causal relationship between mutation rate and any resulting mutation. That is, that the change in fitness of an individual resulting from a mutation is, in general, directly related to the mutation rate in force for that individual. The innovation scheme presently used mutates strategy parameters with a uniform distribution, so any resulting value is equally likely. However, when the innovation rate is less than one, there is an increased probability for the inherited strategy parameter to escape innovation and be transmitted to the next generation. This results in a non-uniform overall distribution for the innovation mechanism, with existing mutation rates being preferred. Thus, we may classify a low innovation rate as one that exploits existing strategy parameter information. In contrast, an innovation rate of one generates each strategy parameter allele with equal probability and is explorative.

We can therefore summarize these results by saying that unimodal problems require an exploitative algorithm, whilst multimodal problems perform best with an explorative algorithm. This finding supports that of Bäck (1992) who reached a similar conclusion regarding the results of using single versus multiple strategy parameters in a self-adaptive GA.

The continuous innovation scheme does not appear to provide the same degree of reliability as the discrete scheme in mutation-only experiments. This suggests that for difficult problems, the search gets stuck in local optima, from which it cannot escape in a reasonable amount of time. The lognormal scheme perturbs the inherited strategy parameter such that the perturbation is small with high probability. Unlike the discrete scheme, which has only a few possible mutation rates, the continuous scheme provides effectively an infinite choice of mutation rates. However since the strategies are encoding for bitwise mutation probabilities, rather than step sizes, many of these rates will have very similar effects in terms of the number of allele values changed in the representation. Thus although this scheme provides variety within the population that does not exist in the discrete scheme, when we consider the likely number of bits mutated, a specific point in the search can vary its

associated search strategy by a large amount with only low probability. This is comparable to the case of the discrete scheme with a low innovation rate and produces similar results.

6.2 PREMATURE CONVERGENCE

Multimodal problems have landscapes containing local optima and search information built up in previous generations may not be particularly useful, since the search can be attracted towards false optima. What is good for an individual at a certain stage of the search (i.e., a low mutation rate) may not necessarily be optimal for the overall search longer term, especially since the self-adaptive algorithm is inherently a greedy adaptive process, as evidenced by the need for non-elitist schemes. To counter this tendency, higher mutation rates must be available if the search is to escape from local optima.

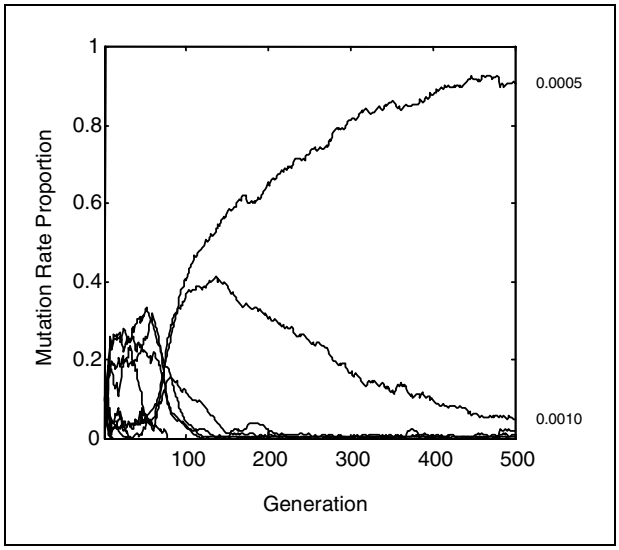


Figure 5 – Premature Convergence of Mutation Rates for f2 (Rastrigin's function)

For high selection pressure, graphs of the proportion of each strategy parameter allele present in the population with low innovation rates show that the population quickly assumes a small number of low mutation rates, typically the two or three lowest available rates. As an example, Figure 5 shows this for f2 (Rastrigin's function). Even though they are being introduced at a fixed rate by innovation, higher mutation rates exist in the population with very low, possibly zero, probability.

Liang et al (1998) and Glickman & Sycara (2000) observe premature convergence of strategy parameters with continuous self-adaptation schemes. Rudolph (1999) shows that convergence of the search to local optima can occur if the step size is reduced too rapidly[3]. At the level of the population, this appears to be the cause of poor reliability with the discrete scheme when using a low innovation rate. The fact that we see premature convergence of strategy parameters together with poor problem solving reliability using the discrete self-adaptation scheme suggests that this may be an effect common to all types of self-adaptation.

The operation of self-adaptation depends on variety of both the individual and its associated strategy parameter. Without adequate variety, self-adaptation will proceed only slowly. Variety is provided by the population and the self-adaptation algorithm to varying degrees, depending on the selection pressure and the nature of the self-adaptation scheme in use. High selection pressure has the characteristic of creating multiple copies of an individual, the strategy parameters of which are varied by innovation. With a low innovation rate and a GA operating without recombination, there are many copies of the individual produced with identical strategy parameters. Low innovation rates used with a high selection pressure are thus wasteful of function evaluations and do not represent a good approach. However, they may be a viable approach for low selection pressures, because fewer copies are produced of each individual and therefore relatively few function evaluations are wasted, especially as the nature of the low selection pressure is to preserve more of the population's variety. This view is supported by inspection of the best/mean/worst fitness (not shown) of low and high innovation rate experiments showing that mean fitness is roughly the same for both low and high innovation rates. However, the best individuals in the population are much more fit and the least fit individuals are much poorer with a high innovation rate. In short, the population shows a higher fitness variance with a high innovation rate.

6.3 METHODS FOR PROVIDING VARIETY

One approach is to assign a variety of strategy parameters to copies of the individual, for example by using an innovation rate of one. If the individuals are mutated and the results evaluated, now the relative fitness of the individual is determined solely by the strategy parameter (ignoring the stochastic effects of mutation). Selection is therefore evaluating the match between the individual and its assigned strategy parameter. This provides an emphasis on the appropriateness of the strategy parameter and hitchhiking of strategy parameters is discouraged. A somewhat similar approach is used in Improved Fast Evolutionary Programming (Yao, Lin & Liu, 1997). This variant of EP selects the best of two offspring generated from the same individual, one based on a step size generated from a Gaussian distribution, the other from a step size generated by a Cauchy distribution. Rudolph (1999) suggests the addition of a fixed step size in order to escape local optima. The discrete scheme with an

[3] We note that Rudolph's proof is based on an elitist EA and Rechenberg's 1/5 success rule.

innovation rate of one takes this one stage further and generates multiple offspring from the same individual with stochastically selected step sizes.

A compromise between retaining the inherited strategy parameter for testing, yet providing variety, is to pass through one copy of the current strategy parameter, together with several different choices of strategy parameter for selection to test. This scheme simultaneously allows exploration of new strategy parameters and exploitation of existing information. Based on the results of the experiments, these would seem to be desirable characteristics of any self-adaptation algorithm. Although this scheme cannot be achieved deterministically using the current discrete innovation algorithm, it is possible to calculate from (2) the probability, P_v, that exactly one of the n (in the present case, five) copies retains the inherited strategy parameter, with the other four having different random alleles:

$$P_v = n \cdot (1 - z + z/q)(z - z/q)^{(n-1)} \qquad (4)$$

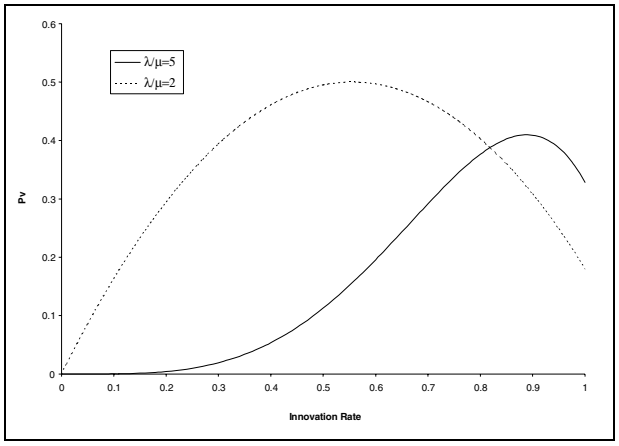


Figure 6 – Probability P_v against Innovation Rate

Figure 6, a plot of this probability against innovation rate, shows that for high selection pressure, P_v is negligible for low innovation rates, but increases with innovation rate, peaking at an innovation rate of approximately 0.89. An innovation rate of one, as used in the experiments gives a P_v value higher than any innovation rate below about 0.74.

This demonstrates that the discrete scheme, even with an innovation rate of one, provides good sampling of innovative strategy parameters *and* still allows transmission of inherited strategy parameter information with a respectable probability. In addition, we find that for low selection pressure, when fewer copies are made of each individual, the probability distribution is substantially different, peaking at a lower innovation rate than that of high selection pressure. This is again consistent with the results reported above. The probability of the inherited strategy parameter being passed through is also much higher than with the high selection pressure over a wide range of innovation rates, such that P_v is higher for low selection pressure than for high selection pressure for all innovation rates except those approaching one. This provides further reinforcement for the importance of selection pressure in the behaviour of self-adaptation.

As mentioned earlier, a deterministic version of this self-adaptation algorithm would provide for P_v=1 by passing through a single copy of the inherited strategy parameter, whilst testing other choices of strategy parameter. The selection pressure, λ/μ, effectively determines the number of copies of each individual made and in the present experiments this ratio is five. However, there are ten possible mutation rates that could be sampled, meaning that some rates will not be tested against each individual in a diverse population. Clearly, it would be possible to match the number of possible mutation rates to the selection pressure in use, especially for high selection pressures. This has the additional advantage that no inheritance mechanism is needed, since all mutation rates are tested against each individual. Further work is needed to experiment with such a scheme and determine the minimum number of strategy parameter alleles that may be successfully used.

7 CONCLUSIONS

We showed that Smith's discrete model is able to provide effective self-adaptation in a GA across a variety of problems with better problem-solving reliability than the typical lognormal self-adaptation scheme. Examination of results from the lognormal self-adaptation scheme showed that best results came from different values of the τ parameter depending on whether the problem had unimodal or multimodal characteristics. Multimodal results suggest tentatively that the relationship presented in (3) holds for GAs. Furthermore, we found that the best results on all the multimodal problems came from using (3) with a setting of $c = 0.1$.

A major finding of this work concerns the choice of an appropriate innovation rate for the self-adaptation mechanism. We found that although a low innovation rate provides the best performance for unimodal problems, in nearly all cases an innovation rate of one gives the best overall reliability and time to optimum figures for landscapes that have local optima or plateaus. Our hypothesis for the reasons for the benefits of a high innovation rate hinge on the relationship between selection pressure and the need for variety across strategy parameters in order for self-adaptation to function effectively. High selection pressure and self-adaptation algorithms favoring small step sizes have the effect of reducing population diversity and encouraging premature convergence. In this situation, self-adaptation schemes need to provide a means of stimulating diversity to counter the forces of selection. This is a side effect of high innovation rates, but it may be possible to design

parameterless self-adaptation schemes that deliver this benefit directly, whilst still providing the advantages of traditional low innovation rate schemes, namely the transmission of historically successful strategies when they are appropriate. A scheme having these features was outlined and it was shown that the discrete model with uniform stochastic innovation provides these characteristics to a certain extent. Further experimentation with purpose-designed algorithms along these lines is needed to determine whether any further performance benefits can accrue. In addition, we need to test the strategy parameter diversity hypothesis on other problems and self-adaptation schemes.

References

Bäck, T. (1992). Self adaptation in genetic algorithms. In F. Varela & P. Bourgine, (eds.), *Towards a Practice on Autonomous Systems: Proceedings of the First European Conference on Articial Life*, Cambridge, MA: MIT Press, pp. 263-271.

Bäck, T. & Hoffmeister, F. (1991). Extended selection mechanisms in genetic algorithms. In R. K. Belew & L. B. Booker, (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications,* San Mateo, CA: Morgan Kaufmann, pp. 92-99.

Bäck, T. & Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. In Z. Ras, (ed.), *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems, Lecture Notes in Computer Science,* Berlin: Springer, pp. 158-167.

Baker, J. E. (1987). Reducing bias and innefficiency in the selection algorithm. In J. J. Grefenstette (ed.), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms,* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 14-21.

Fogel, D. B., Fogel, L. J. & Atmar, J. W. (1991). Meta-evolutionary programming. In R. R. Chen (ed.), *Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers*, San Jose, CA: Maple Press, pp. 540-545.

Glickman, M., R. & Sycara, K. (1998). Evolutionary algorithms: Exploring the dynamics of self-adaptation. In J. R. Koza et al (eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference,* San Francisco, CA: Morgan Kaufmann.

Glickman, M. R & Sycara, K. (2000). Reasons for premature convergence of self-adapting mutation rates. In *CEC2000: Proceedings of the Congress on Evolutionary Computation,* Piscataway, NJ: IEEE Press.

Hinterding, R. (1997). Self-adaptation using multi-chromosomes. *In Proceedings of the Fourth IEEE International Conference on Evolutionary Computation,* Piscataway, NJ: IEEE Press, pp. 87-91.

Hinterding, R., Michalewicz, Z. & Peachey, T. C. (1996). Self-adaptive genetic algorithm for numeric functions. In H-M. Voigt, W. Ebeling, I. Rechenberg, & H-P. Schwefel, (eds.), *Parallel Problem Solving from Nature - PPSV IV, volume 1141 of Lecture Notes in Computer Science*, Berlin: Springer, pp. 420-429.

Liang, K-H., Yao, X., Liu, Y., Newton, C. & Hoffman, D. (1998). An experimental investigation of self-adaptation in evolutionary programming. In V. Porto, N. Saravanan, D. Waagen & A. Eiben (eds.), *Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, volume 1447 of Lecture Notes in Computer Science,* Berlin: Springer, pp. 291-300.

Rudolph, G. (1999). Self-adaptive mutations may lead to premature convergence. Technical Report CI-73/99, Department of Computer Science/XI, University of Dortmund.

Schwefel, H-P. (1997). Evolutionary computation - a study on collective learning. In N. Callaos, C. M. Khoong, E. Cohen (eds.), *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, volume. 2,* Orlando FL: International Institute of Informatics and Systemics, pp. 198-205.

Schwefel, H-P. (1981). *Numerical Optimisation of Computer Models*. New York, NY: Wiley.

Smith, J. E. (2001). Modelling GAs with self adaptive mutation rates. In L. E. Spector et al (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001,* San Francisco, CA: Morgan Kaufmann, pp. 599-606.

Smith, J. E. & Fogarty, T. C. (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of IEEE International Conference on Evolutionary Computing 1996,* Piscataway, NJ: IEEE Press, pp. 318-323.

Stephens C. R., García Olmedo I., Mora Vargas J. & Waelbroeck, H. (1998). Self-Adaptation in Evolving Systems. *Artificial Life* 4(2), 183-201.

Stone, C. N. (2001). *Discrete Self-Adaptive Mutation in Genetic Algorithms.* M.Sc. dissertation. University of the West of England.

Yao, X., Lin, G. & Liu, Y. (1997). An analysis of evolutionary algorithms based on neighbourhood and step sizes. In P. Angeline, R. Reynolds, J. McDonnell & R. Eberhart (eds.), *Evolutionary Programming VI: Proceedings of the Sixth Annual Conference on Evolutionary Programming, volume 1213 of Lecture Notes in Computer Science*, Berlin: Springer, pp. 297-307.

A Simple Method for Detecting Domino Convergence and Identifying Salient Genes Within a Genetic Algorithm

Hal Stringer
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816
stringer@cs.ucf.edu

Annie S. Wu
School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816
aswu@cs.ucf.edu

Abstract

Within a genetic algorithm, all genes may not be created equal. This concept is the central idea explored in this paper. A second and equally important idea is that this inequality in gene importance or salience can be detected and identified within a GA. To support these ideas, a technique for directly measuring genetic diversity within a GA population and thereby indirectly measuring gene-specific importance is provided. Diversity graphs are offered as a powerful technique for visualizing measurement results. Our theories, metrics and tools are tested on GAs for two problem classes and four different selection methods.

1 INTRODUCTION

Within a genetic algorithm (GA), all genes may not be created equal. Anecdotal evidence of this can be obtained from any student of genetic algorithms who has attempted to solve a symbolic regression problem. For example, consider a GA which finds the coefficients for the following equation:

$$y = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g + h\cos(x) \quad (1)$$

Intuitively, we expect genes representing the variables *a* through *h* to have varying impacts on fitness evaluation due to differences in the exponential order associated with each term. We would further expect the *a*-gene and *b*-gene to be the most important in determining fitness of an individual. The values for genes *c* through *h* would be largely irrelevant in terms of raw fitness until these first two genes converged to some local optimum.

The idea of a gene's importance or temporal-salience has already been described in (Thierens, Goldberg & Pereira, 1998). A side effect of this property is the phenomenon of "domino" convergence introduced in (Rudnick, 1992). A GA with non-uniformly salient genes converges serially over time starting with the more important genes and moving to less salient genes similar to the way a row dominos falls. Domino convergence and variations in gene importance have been shown to occur in genetic algorithms attempting to solve exponentially scaled fitness problems.

In subsequent works, (Goldberg, 1999) and (Srivastava & Goldberg 2001) explored how gene salience and domino convergence can be used to develop GAs with a serial mode of processing. A serial GA consists of small populations and short epochal runs. During each epoch different salient genes converge to their respective optimal values. Between each epoch, a continuation operator is activated to rejuvenate the diversity of less salient genes while leaving more important (and previously converged) genes alone.

Without continuation operators, GAs for exponentially scaled problems tend to converge around highly salient genes. The GA may then drift and stall at a less than optimal solution due to lack of diversity in less salient genes.

The use of epochs and continuation operators was found unproductive for problems with uniformly salient genes (e.g., OneMax). For these types of problems, the traditional GA's implicit parallelism, larger populations, and single long epoch were found to still be the most productive method of processing.

We believe that the idea of gene-specific temporal salience provides a valuable insight into how a GA functions. In the case of exponentially scaled problems, the concept opens up new opportunities for developing continuation operators to fine tune GA performance. But in order to use this approach, we must first find an effective method to determine if a problem includes genes with non-uniform salience and if so, a method for identifying those genes that are more important than others. This is particularly important in problems with very large numbers of genes where *a priori* knowledge of gene salience is less likely. In this work, we present a simple method for detecting domino convergence and identifying genes with high levels of importance. We show how tracking gene diversity within a GA population

can provide the information we need to obtain a measurement of gene salience.

Our measurement technique focuses on two metrics. The first is the variation in unique alleles associated with each gene in a population. Unique allele counts plotted over time (generations) constitutes a convergence profile for a given problem and selection method. This profile clearly indicates the presence or absence of domino convergence. Our second metric consists of a ratio of unique sub-genotypes to alleles and assigns a numeric salience value to each gene. Graphically presented, this ratio gives us a salience profile identifying genes of higher importance.

In this work we describe the general nature of the experiments we performed to test the use of diversity as a salience indicator. Experiments include GAs for different problem classes and various selection methods. This range of problem classes and selection methods allow us to validate our method against previous theoretical work performed by other researchers.

2 MEASURING GENE SALIENCE THROUGH GENETIC DIVERSITY

The concept of gene salience or importance is all around us. For example, normal human beings are born with two eyes. Yet there exist numerous variations in eye color within the population. On a simplistic level, we can assume that the genes which affect the number of eyes in one's head are more important than those affecting eye color. We can also assume that a lack of diversity in the number_of_eyes genes relative to eye_color genes indicates that the first is more salient than the others.

The same concept applies to genetic algorithms with non-uniform gene importance. Over time, diversity of salient genes diminishes faster than that of non-salient genes. Less salient genes are not subject to the same selection pressures due to their low fitness impact. The diversity of alleles for each gene in a population relative to other genes provides a good indication of gene salience. The less diverse, the more important.

Using this idea we began investigating various ways to measure genetic diversity (or lack there of) within a GA. Initial experiments looked at uniqueness of entire chromosomes within a population. It was assumed that this method would provide a good showing of genetic diversity and illustrate how a population converges toward a small number of similar individuals over the course of multiple generations. This method was tested but found to be unsatisfactory. Looking at entire chromosomes did not single out specific genes nor indicate their specific importance. Nor did this method clearly show the presence or absence of domino convergence. We also investigated convergence to fitness values as a way of tracking convergence and diversity. This also proved to be less than satisfactory in identifying salient genes.

Throughout these initial experiments, we notice that there appeared to be a strong correlation between gene-salience and diversity of alleles within a single gene and also within partial chromosomes ("sub-genotypes"). The final version of our measurement methods used this idea and are described below.

2.1 UNIQUE ALLELES

The starting point for our method of determining genetic diversity within a GA is to count the number of unique alleles for each gene within a population at a given time. An allele can be thought of as a single representation instance of a gene. For example, using bit strings to represent a 9-bit gene allows for 2^9 different alleles.

For notational purposes a single gene location within a chromosome will be identified as G_i where $1 \le i \le n$ and n equals the total number of genes which make up a single chromosome. Two additional subscripts t and j are added to further specify a gene. t indicates a specific time or generation. j identifies an individual chromosome where $1 \le j \le p$ and p equals the population size. For example, $G_{3,\ 100,\ 12}$ denotes the third gene located on individual 12's chromosome at generation 100.

$U(G_{i,t})$ will be used to denote the count of unique alleles for G_i within the total population at the start of generation t.:

$$U(G_{i,t}) = \left| \{ G_{i,t,\,j} \mid G_{i,t,\,j} \neq G_{i,t,k} \text{ where } 1 \le j,k \le p \} \right|$$

To illustrate, assume that at the start of generation 54 during a GA's run, the third gene on all chromosomes contained bit representations (genotypes) for one of the following numbers: 12, -47, 178 or 3 (phenotypes). The population has evolved to contain chromosomes with only four unique alleles in the third position. In this example, $U(G_{3,54}) = 4$. Note that we are not concerned with how many genes contain a given allele, only the number of unique alleles within the population. $U(G_{i,t})$ provides a measure of the diversity of G_i within the population at the start of generation t.

Interesting results were obtained by following the behavior of a population using this measure. A low $U(G_{i,t})$ for a given gene relative to other genes in a chromosome indicates that the population is converging towards a few select alleles thus towards some local optimum. Unfortunately, the difference between $U(G_{i,t})$ for all genes within a GA was sometimes very small. This limited our ability to draw any firm conclusions regarding a specific gene's level of importance. Nor did this single statistic provide a total picture of what was occurring within the GA as a whole. Additional information was required.

2.2 UNIQUE SUB-GENOTYPES

Counting unique alleles gave us a way to track convergence of a given gene. But what about the rest of the genetic material within a chromosome?

To answer this question, we have developed the idea of a partial chromosome or "sub-genotype". A sub-genotype

is the entire chromosome excluding a single gene. For notational purposes, $S_{i,t}$ will represent a chromosome's sub-genotype with respect to G_i at the start of generation *t*. The sub-genotype for a specific gene consists of the concatenation of all genetic material in the chromosome excluding the gene itself.

The example below illustrates how allele representations and sub-genotypes are derived from a hypothetical five-gene chromosome associated with individual 9 at generation 60:

Original Chromosome #9 at Start of Generation 60:

Gene:	#1	#2	#3	#4	#5
Value:	1010	1111	0011	0000	1101

Derived Gene Values and Sub-Genotypes:

$G_{1,60,9} = 1010$, $S_{1,60,9} = 1111\ 0011\ 0000\ 1101$
$G_{2,60,9} = 1111$, $S_{2,60,9} = 1010\ 0011\ 0000\ 1101$
$G_{3,60,9} = 0011$, $S_{3,60,9} = 1010\ 1111\ 0000\ 1101$
$G_{4,60,9} = 0000$, $S_{4,60,9} = 1010\ 1111\ 0011\ 1101$
$G_{5,60,9} = 1101$, $S_{5,60,9} = 1010\ 1111\ 0011\ 0000$

$U(S_{i,t})$ will be used to denote the count of unique sub-genotypes within the total population at generation *t*:

$$U(S_{i,t}) = \left| \{ S_{i,t,j} \mid S_{i,t,j} \neq S_{i,t,k} \text{ where } 1 \leq j,k \leq p \} \right|$$

2.3 RATIO OF SUB-GENOTYPES TO ALLELES

As a final measure of diversity, we also looked at the ratio of sub-genotype counts to the count of unique alleles. This ratio (R) is equal to the sub-genotype count divided by the unique allele count and can be shown as follows:

$$R_{i,t} = U(S_{i,t}) / U(G_{i,t})$$

Examples illustrating the importance of this relationship will be given later. For now, it is sufficient to say that this ratio "amplifies" the measurement of gene-specific salience and provides an better indicator of this important characteristic.

3 EXPERIMENT DESIGN

Many experiments were performed to capture the metrics described in Section 2. The purpose of these experiments was to test our ability to detect non-uniform salience and identify the salient order of genes within a chromosome. Experiments involved calculating and then graphing $U(G_{i,t})$, $U(S_{i,t})$ and $R_{i,t}$ for a variety of problem classes and selection methods. An analysis of the data obtained from the experiments supports our proposal that genetic diversity can reveal gene-specific salience in a GA.

Two different problem classes were tested and included in this paper: Symbolic Regression and OneMax. It was our expectation that gene-specific salience would be found in the symbolic regression problem. Based on the work researchers previously cited, we should find no important genes in the OneMax problem.

Experiments were conducted as follows:

1. A GA was executed for 50 runs of 100 generations each. All runs were initialized with a different random number seed.
2. All unique alleles and associated sub-genotypes were counted for each gene during each generation.
3. The allele and sub-genotype counts from step 2 were averaged across all 50 runs.
4. A ratio of the values from step 3 was calculated for each generation. Ratios were summed and divided by 100 for an average ratio across all generations.
5. The results from 3 and 4 were plotted for each problem as a set of six diversity graphs.

3.1 GA PARAMETERS AND SETTINGS

Our experiments used one of four selection methods: Fitness Proportional, Tournament, Rank Proportional and Random. Features and parameters incorporated into our GA for all experiments included the following: Population Size = 200 Individuals, Representation Method = Bit String, Number of Genes per Chromosome = 8, Number of Bits per Gene = 9, Crossover Type = 2-Point, and Crossover Rate = 100%.

With the exception of one experiment, mutation was not employed in any of our experiments. Our diversity metrics are based on counts of unique alleles and sub-genotypes. Mutation has the effect of increasing overall diversity in a population and tended to obscure though not hide our results. Leaving out mutation allows us to remove its effects from our measurements and focus on the evolution of individuals using only genetic material available from the initial population. One can think of the results of our mutation-less experiments as providing a baseline measure of gene salience and selection pressure within a GA.

3.2 COUNTING UNIQUE ALLELES AND SUB-GENOTYPES

The method proposed in this paper for identifying gene-specific salience requires that the number of unique alleles and sub-genotypes be determined for each gene in each generation. There are many different methods that can be used for such a counting function, some more efficient than others. The method employed for our experiments was simple though not necessarily the most efficient computationally.

All genes consisted of 9-bit binary strings representing integer values from –255 to +255. During fitness evaluation, these genotypic strings were converted to their phenotypic decimal equivalents. Genes were left in their original string format for counting purposes.

At the beginning of each experiment an m x n array (*count*) was constructed for storing unique allele counts where m = 100 was the number of generations in a run and n = 9 was the number of genes in each chromosome. All array elements were initialized to 0.

A hash table was used to keep track of unique alleles. The table was queried for the existence of each allele during the counting process. A gene value not found in the hash table was considered to be a new unique allele – the first of its kind. The corresponding element in *count* was incremented by 1 and the gene was added into the hashtable. If an allele was found to already exist in the hash table, no action was taken. The uniqueness of the allele had already been noted and added to the count for that gene during that generation. The following pseudo-code further illustrates this process:

```
for (i=1; i<=number_of_genes; i++){
  clear hash table;
  for (j=1; j<=population_size; j++){
     extract gene Gi from chromosome;
     if (Gi not in hashtable){
        add 1 to count[generation][i];
        add Gi to hashtable;
     }
     else no action necessary;
  }
}
```

A similar process was utilized to count unique sub-genotypes associated with each gene. It should be noted that the counting method described here is based primarily on the number of genes in a chromosome and is therefore usable with both small scale GAs and GAs with larger gene sizes (number of bits) or populations.

3.3 VISUALIZATION OF DIVERSITY

Results were written from the *count* array to a comma-delimited text file at the end of each experiment. The file contained the count of unique gene values and sub-genotypes for all 100 generations. Using data from this file, two graphs were plotted for each term described previously ($U(G_{i,t})$, $U(S_{i,t})$ and $R_{i,t}$.) One graph shows the change in the term over time (by generation) giving us an online view. The second graph shows an offline average value for each term for the entire GA run. Thus for each experiment, a suite of six graphs was prepared which, when viewed as a set, provided an excellent picture of the changing genetic diversity within a GA. Examples of these diversity graphs are provided throughout the remainder of this work along with our analysis.

4 FINDING TEMPORAL SALIENCE

Given the introduction to this work, it is fitting that symbolic regression be the first problem used to test our diversity measurement technique. Predetermined x and y values were provided as input to the GA's fitness function. The GA's task was to find optimal values for coefficients a through h. Positionally, these coefficients corresponded to genes 1 through 8 on a chromosome.

Intuition and knowledge of the problem lets us know that the first gene (G_1) will be the most salient and have the greatest impact on fitness evaluations due to its association with the term ax^6. The population should converge around this one gene before all others. G_2 representing the coefficient for bx^5 would be next in importance followed by G_3, G_4,G_5 and so on.

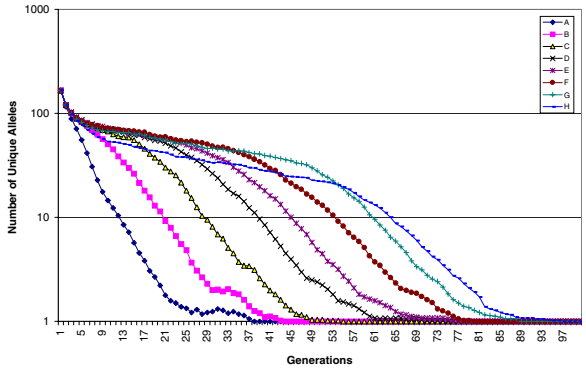
Experiments were run per the design in Section 3 using tournament selection. Unique alleles and sub-genotypes for all runs were counted, averaged and plotted on a set of diversity graphs (see Figure 1).

Figure 1(a) shows the convergence profile for this problem/selection method combination. Allele diversity for the gene associated with the a coefficient - $U(G_1)$ - drops fastest followed by $U(G_2)$ and the other genes. By generation 41, only one allele for G_1 exists in the population. The gene's salience caused a single value to quickly take over this gene in the entire population. This graph also shows a similar but delayed behavior for G_2 though G_8 over the course of 100 generations. The result is a staggered look to the graph clearly indicating the domino convergence occurring in this experiment.

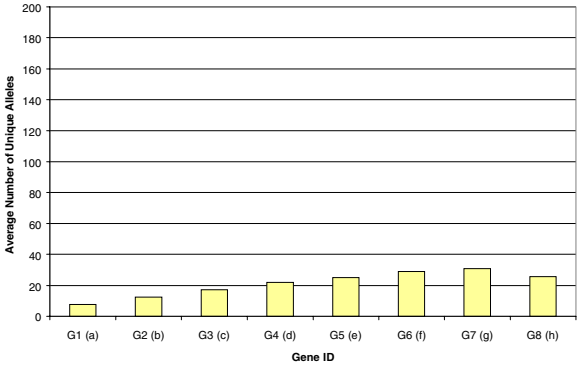
Figure 1(b) shows the diversity of sub-genotypes for this problem. Diversity for all sub-genotypes drops decreases over time as the GA converges to a single result.

Figures 1(d) and 1(e) provide an offline view of allele and sub-genotype counts. Both of these graphs contain the average number of unique alleles or sub-genotypes over 100 generations. For our test problem, G_1 and G_2 have the lowest average unique number of alleles. On average, only 7.7 different values for G_1 existed during each generation due to this gene's salience. Although hard to tell from the graph, G_1 has the greatest sub-genotype diversity. On average, any given individual in the population will include one of 105 G_2-through-G_8 gene combinations regardless of G_1's value. It was found that generally the lower the unique allele count, the higher the sub-genotype count within the GA.

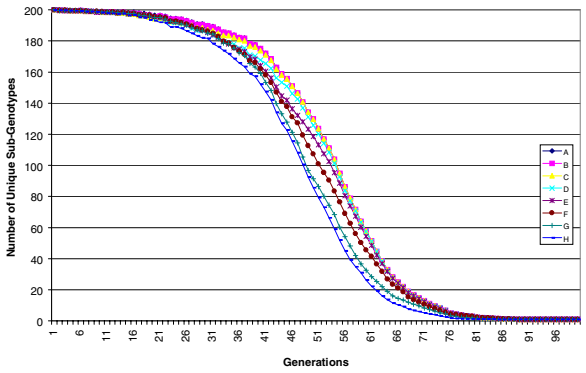
From these first four graphs, we can see that G_1 and G_2 or coefficients a and b respectively, are the more important genes and exert a higher degree of selection pressure than the other genes in this GA. But these graphs alone may not be enough to clearly indicate gene salience. A more reliable indicator has proven to be the ratio of sub-genotypes to gene values ($U(S_{i,t})$ / $U(G_{i,t})$). These ratios are plotted for our same test problem in Figures 1(c) and 1(f).



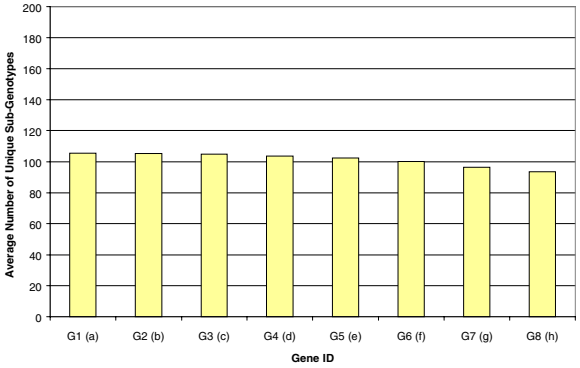
(a) Unique Allele Counts U(G_i) over Time (Convergence Profile)



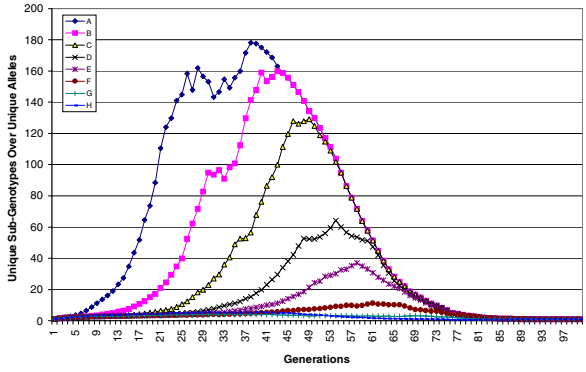
(d) Avg. Unique Alleles per Gene



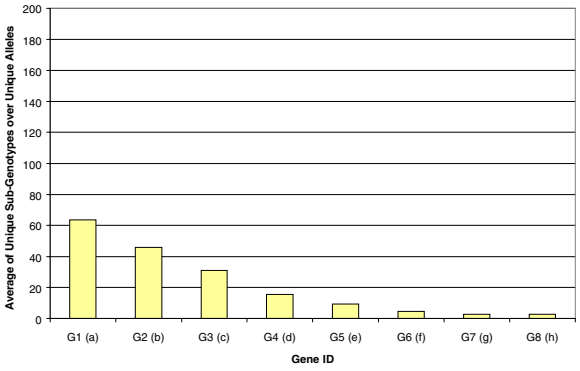
(b) Unique Sub-Genotype Counts U(S_i) over Time



(e) Avg. Unique Sub-Genotypes per Gene



(c) Ratio (R_i) of Sub-Genotypes to Alleles over Time



(f) Avg Ratio (R_i) of Sub-Genotypes to Alleles (Salience Profile)

Figure 1: Diversity Graphs for 8-Term Symbolic Regression Problem Using Tournament Selection

As mentioned earlier, this ratio $R(G_{i,t})$ tends to amplify our ability to detect gene-specific importance and make it easier to pick out the genes with greatest salience. Figure 1(f) is most important to us and we have called this type of plot a "salience profile." From Figure 1(f) it is very clear which genes in our GA are more salient than others.

A variation on the preceding symbolic regression problem was developed to check the previous results. In this second problem, the positional order of terms was mixed. The resulting equation is:

$$y = ax + bx^4 + c\ cos(x) + dx^5 + e + fx^2 + gx^6 + hx^3$$

Assuming our proposal is correct, G_7, G_4, and G_2 should exhibit behavior that typifies genes of higher importance. Figure 2 shows the salience profile for this reordered problem. As expected, G_7, G_4 and G_2 had the highest average ratio of unique sub-genotypes to alleles $R(G_{i,t})$ of the eight genes.

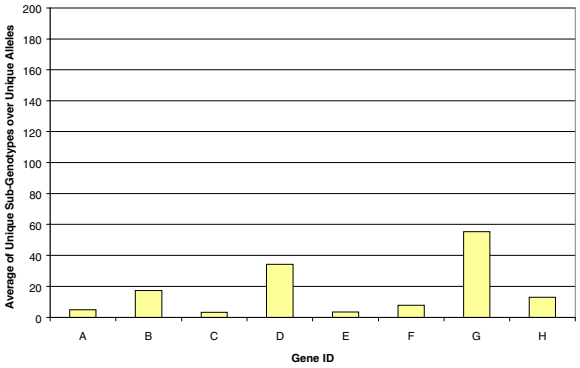


Figure 2: Saleince Profile for Symbolic Regression Problem with Reordered Terms

This second experiment confirms that our measurement technique can identify salient genes regardless of their position within a chromosome.

As mentioned in Section 3, most of our experiments were run without mutation. For sake of completeness, we incorporated bit mutation at a rate of 0.01 in a third experiment using equation (1). Figure 3 shows the salience profile for this GA. A comparison of Figure 3 with Figure 1(f) shows that mutation reduced but did not eliminate the indication of gene-specific salience calculated from $R(G_i)$. Using a magnified *y*-axis, the stair step pattern indicating the presence of domino convergence is still apparent.

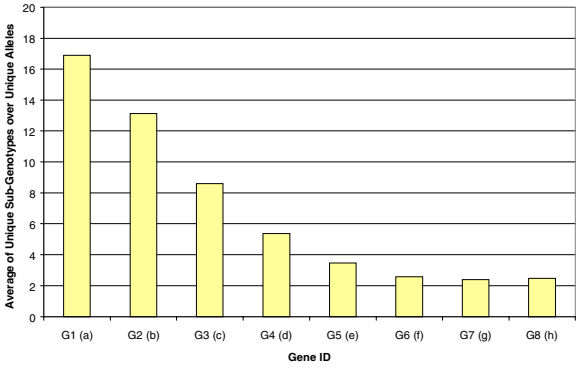


Figure 3: Salience Profile for Symbolic Regression Problem with Mutation = .01

5 OTHER SELECTION METHODS

Graphs in Figures 1, 2 and 3 were associated with GAs using tournament selection. How well does our measurement technique work with other selection methods? To answer this question we present convergence and diversity profiles for GAs solving equation (1) using random and fitness proportional selection (Figures 4 and 5 respectively). Space does not allow a detailed description of the results. However, a few points should be noted.

The plots for random selection show that lack of directed selection pressure leads only to drift in gene diversity.

Gene-specific salience also appears in GAs run with fitness proportional selection. The exponential effect of the selection method causes the GA to converge very rapidly around highly salient genes. As a result we do not see the stair stepped or staggered type of convergence profile found in Figure 1. The salience profile is stronger for genes of higher importance.

The important concept to be seen in these graphs is the impact of selection method. Rank or tournament selection is best for detecting domino convergence and identifying the gene order in terms of salience. However, fitness proportional selection provides a very clear indication of the importance of the most salient genes in a chromosome. As a result, fitness proportional selection may be most useful when results for other selection methods are less clear.

Experiments testing our method on rank fitness were also performed. We do not include the diversity and salience profiles for these experiments as they were very similar to those of tournament selection. When combined with a OneMax problem, profiles for binary tournament and rank selection were nearly identical as was expected based on showings in (Blickle & Thiele, 1995).

6 OTHER PROBLEM CLASSES

It appears that we have found a simple method for identifying domino convergence and gene-specific salience in a GA. But what about detecting a lack of gene importance? Random selection results in the elimination of gene salience regardless of the problem type. Can we also show that a problem class in and of itself lacks salient genes. To further test our technique, we ran a GA using a OneMax problem. The fitness function merely counts the number of ones in the entire chromosome. This problem has been shown to have uniform salience across all genes. Figure 6(a) and (b) contain convergence and salience profiles for this problem using tournament selection. We can see from these plots that no gene is more salient than another.

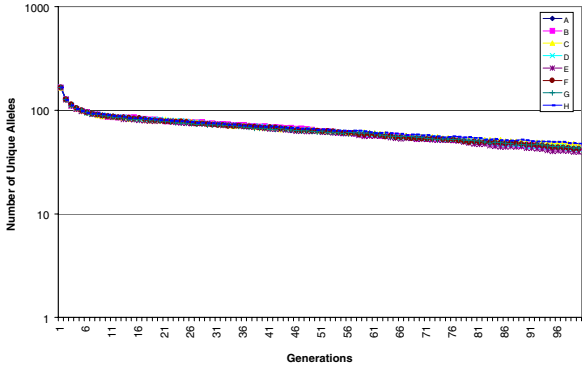


Figure 4(a): Convergence Profile for Symbolic Regression Problem with Random Selection

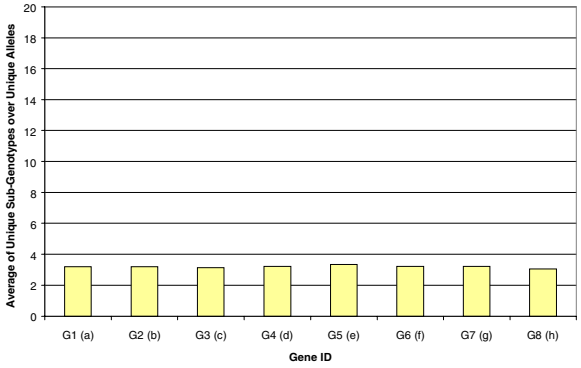


Figure 4(b): Salience Profile for Symbolic Regression Problem with Random Selection

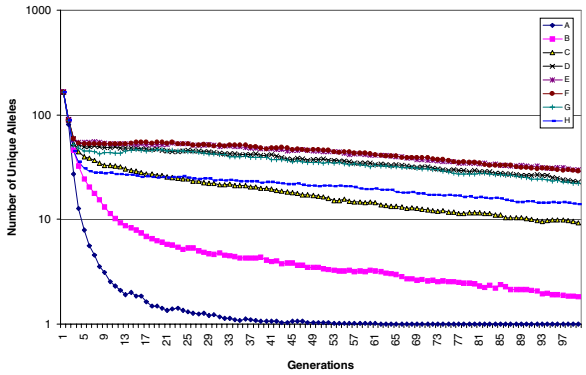


Figure 5(a): Convergence Profile for Symbolic Regression with Fitness Proportional Selection

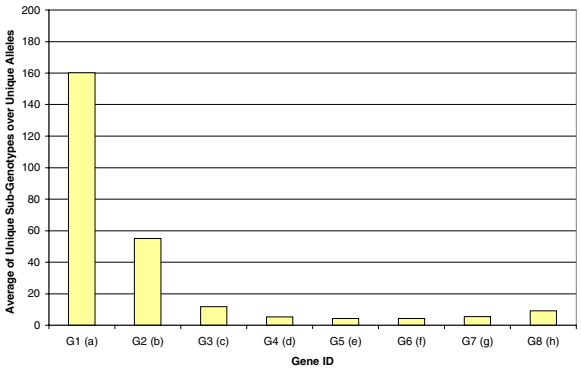


Figure 5(b): Salience Profile for Symbolic Regression with Fitness Proportional Selection

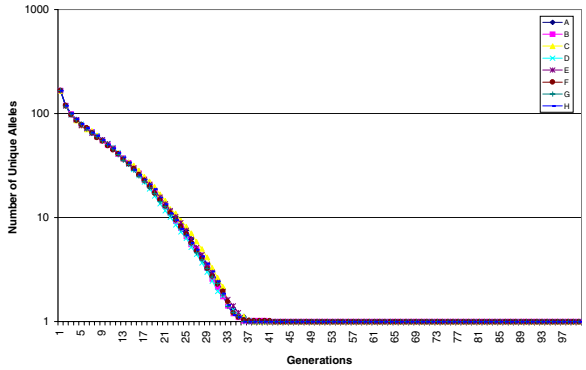


Figure 6(a): Convergence Profile for One-Max Problem with Tournament Selection

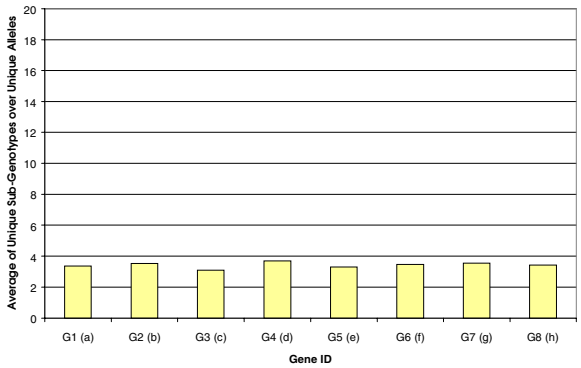


Figure 6(b): Salience Profile for OneMax Problem with Tournament Selection

7 CONCLUSIONS

In this work we have presented a simple but useful method for detecting domino convergence and gene-specific salience within a given problem. It is not uncommon for certain regions of GA individuals to consistently converge early. Those regions are typically expected to be regions that have high impact on the fitness function. The ability to detect high impact regions would allow practitioners to potentially develop operators that may improve GA performance on a particular problem. Our detection method is based on a count of the unique gene alleles and unique sub-genotypes that occur within a short run. While both of these counts provide some indication of gene salience, it is the ratio of the sub-genotype count to the unique allele count that appears to give the clearest picture as to which genes have the strongest impact on the GA search process.

We tested our method for detecting salient genes on problems in which genes are and are not expected to have varying impact. From results we are clearly able to detect salient genes when they exist, regardless of their position within a chromosome.

A comparison of salience profiles for varying selection methods indicate that choice of selection method can enhance or diminish gene-specific salience depending on the desires of the GA researcher/developer. Our experimental evidence shows that fitness proportional selection magnifies a gene's selection pressure. Tournament or rank fitness selection methods reduce that pressure and allow the temporal salient nature of more genes to shine through.

Choice of selection method is an example of how gene-salience can be manipulated on a chromosomal- or problem-wide scale. But can we manipulate selection pressure at the gene level? The use of continuation operators is a step in that direction. We believe that the ability to identify salient genes within a GA will help researchers in those development efforts.

While the information presented here is of value we do recognize that our methods have their limitations. Specifically, our methods were designed for GAs with fixed gene positions and would not be directly applicable to locus-variable situations such as messy GAs or GAs with variable length chromosomes. We believe the development of methods for detecting gene salience in these other GA categories will be a productive area for future research.

In addition, our research focused on gross numerical counts of unique allele values and sub-genotypes This approach can suffer from scalability issues which may be addressed by taking measurements on restricted GA runs (e.g., short duration or small populations.) These restricted runs can reduce processing time while still providing information about the problem. Such gross numerical counts also make no attempt to evaluate genes or sub-genotypes qualitatively. Further research in these areas are expected to provide a greater understanding of genetic diversity within a GA.

Despite these limitations, we feel the knowledge gained from our research has immediate value. We can now detect domino convergence within a GA and thus non-uniform gene salience. In addition, we can identify important genes within these GAs and begin to use this knowledge towards development of better control mechanisms.

In terms of immediate applications, our method may be helpful in a number of ways. The programming effort required to extract our measurements (V(Gi,t), S(Gi,t) and R(Gi,t)) is relatively small. A few lines of code added to any fixed position GA would allow a quick view of any gene-specific temporal salience that the GA might encounter.

We feel our diversity graphs will be useful in development and evaluation of new genetic operators and selection methods. We have already mentioned the strikingly similar results found for rank and tournament selection which concur with theoretical studies. Diversity graphs would show where new selection methods are similar to existing methods and where they differ.

References

Blickle, T. & Thiele, L. (1995). A Comparison of Selection Schemes Used in Genetic Algorithms. *TIK-Report No. 11*, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Goldberg, D. E. (1999). Using Time Efficiently: Genetic-Evolutionary Algorithms and the Continuation Problem, In *Proceedings of the Genetic and Evolutionary Computation Conference: GECCO 1999,* Volume 1 pp. 212-219, Morgan Kaufman Publishers: San Francisco.

Rudnick, W. Michael (1992). *Genetic Algorithms and Fitness Variance with an Application to the Automated Design of Artificial Neural Networks*. Unpublished doctoral dissertation, Oregon Graduate Institute of Science and Technology.

Srivastava, R.P. & Goldberg D.E. (2001). Verification of the Theory of Genetic and Evolutionary Continuation, *In Proceedings of the Genetic and Evolutionary Computation Conference: GECCO 2001*, Morgan Kaufman Publishers: San Francisco.

Thierens, D., Goldberg, D. E., & Pereira, A. B. (1998). Domino Convergence, Drift and the Temporal-Salience Structure of Problems, In *The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pp. 535-540, IEEE Press: New York, NY.

Variable Dependence Interaction and Multi-objective Optimisation

Ashutosh Tiwari and Rajkumar Roy

Department of Enterprise Integration,
School of Industrial and Manufacturing Science,
Cranfield University, Cranfield, Bedford,
MK43 OAL, United Kingdom (UK).
E-mail: {a.tiwari, r.roy}@cranfield.ac.uk
Tel: +44 (0) 1234 754072, Fax: +44 (0) 1234 750852

Abstract

Interaction among decision variables is inherent to a number of real-life engineering design optimisation problems. There are two types of interaction that can exist among decision variables: inseparable function interaction and variable dependence. The aim of this paper is to propose an Evolutionary Computing (EC) technique for handling variable dependence in multi-objective optimisation problems. In spite of its immense potential for real-life problems, lack of systematic research has plagued this field for a long time. The paper attempts to fill this gap by devising a definition of variable dependence. It then uses this analysis as a background for identifying the challenges that variable dependence poses for optimisation algorithms. The paper further presents a brief review of techniques for handling variable dependence in optimisation problems. Based on this analysis, it devises a solution strategy and proposes an algorithm that is capable of handling variable dependence in multi-objective optimisation problems. The working of the proposed algorithm is demonstrated, and its performance is compared to that of two high performing evolutionary-based multi-objective optimisation algorithms, NSGA-II and GRGA, using two test problems extracted from literature. The paper concludes by giving the current limitations of the proposed algorithm and the future research directions.

1 INTRODUCTION

Real-life engineering design optimisation problems, as opposed to the theoretical problems (test cases), are those that are encountered in industry. Some examples of these problems are the design of aerospace structures for minimum weight, the surface design of automobiles for improved aesthetics and the design of civil engineering structures for minimum cost (Rao, 1996). A survey of industry and literature reveals that along with multiple objectives, constraints, qualitative issues and lack of prior knowledge, most real-life design optimisation problems also involve interaction among decision variables (Roy et al., 2000). However, lack of systematic research has plagued the field of interaction for a long time. This can mainly be attributed to the lack of sophisticated techniques, and inadequate hardware and software technologies. However, in the last two decades, with the improvements in hardware and software technologies some research has been carried out in this area especially in the field of statistical data analysis (Draper and Smith, 1998). This has been further augmented in the recent past with the growth of computational intelligence techniques like Evolutionary Computing (EC), Neural Networks (NNs) and Fuzzy Logic (FL) (Pedrycz, 1998). This paper focuses on the development of an evolutionary-based algorithm for handling variable interaction in multi-objective optimisation problems.

2 TYPES OF VARIABLE INTERACTION

In an ideal situation, desired results could be obtained by varying the decision variables of a given problem in a random fashion independent of each other. However, due to interaction this is not possible in a number of cases, implying that if the value of a given variable changes, the values of others should be changed in a unique way to get the required results. The two types of interaction that can exist among decision variables are discussed below.

2.1 INSEPARABLE FUNCTION INTERACTION

The first type of interaction among decision variables, known as inseparable function interaction, is discussed in detail by Tiwari et al. (2001). This interaction occurs when the effect that a variable has on the objective function depends on the values of other variables in the

function (Taguchi, 1987). This concept of interaction can be understood from Figure 1.

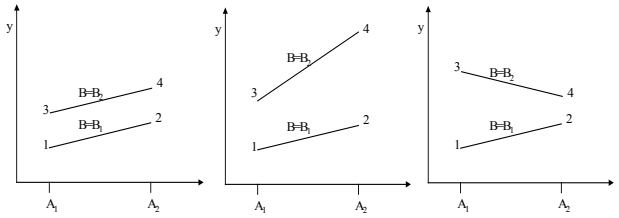


Figure 1: Examples of Interaction (a) No Interaction (b) Synergistic Interaction (c) Anti-synergistic Interaction (Phadke, 1989)

In GA literature, the inseparable function interaction, as defined above, is termed as epistasis. The GA community defines epistasis as the interaction between different genes in a chromosome (Beasley et al., 1993). A review of literature reveals that the evolutionary-based techniques for handling inseparable function interaction can be classified into two broad categories based on the approach used for the prevention of building block disruption. These categories involve managing the race between linkage evolution and allele selction (Harik, 1997), and modelling the promising solutions (Muhlenbein and Mahnig, 1999).

A number of real-life examples can be found in literature that involve this type of interaction. For example, the temperature (T) of an ideal gas varies with its pressure (P) and volume (V) as T=kPV, where k is the constant of proportionality. This equation has cross-product term PV clearly demonstrating the interaction between P and V in the definition of T.

2.2 VARIABLE DEPENDENCE

The second type of interaction among decision variables, known as variable dependence, is the main focus of this paper. This interaction occurs when the variables are functions of each other, and hence cannot be varied independently. Here, change in one variable has an impact on the value of the other. A typical example of this type of interaction is the case when the function y is A^2+B^2, where A and B are as defined below.

$$A = Random(a, b)$$
$$B = f(A) + Random(c, d)$$

As can be seen, variable A is fully independent and can take any random value between a and b. On the other hand, variable B is not fully independent and has two components. The first component, which is a function of variable A, takes values depending on the values of A. The second component is a random number lying between c and d. It should be noted that in case of no dependence among decision variables, a and b define the range of variable A, and c and d define the range of variable B.

The above example reveals that the presence of dependence among decision variables has the following effects.

- Both variables A and B cannot simultaneously take random values in their respective ranges. If variable A takes a value A_1, variable B can take only those random values that lie between $[f(A_1)+c]$ and $[f(A_1)+d]$. With the change in value of A, the range of random values that B can take also changes. So, the variables cannot be varied independently of each other.
- The above discussion implies that the presence of dependence among decision variables modifies the shape and location of variable search space. In case of no dependence among decision variables, both variables A and B can independently take random values in their respective ranges making the A-B search space rectangular in shape. However, the presence of dependence makes the search space take the shape and location based on the nature of function $f(A)$.

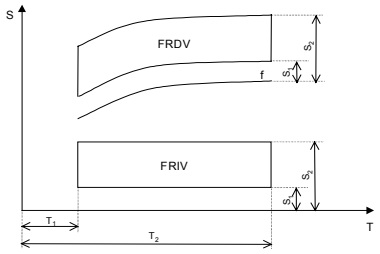


Figure 2: Relationship between Stress(S) and Temperature(T) (FRIV: Feasible Region with Independent Variables and FRDV: Feasible Region with Dependent Variables)

The dependence among decision variables is frequently observed in real-life problems. As an example, the resistance (R) of a wire is defined in terms of two variables, namely Temperature (T) and Stress (S), where T and S are as defined below.

$$R = F(S, T)$$
$$T = Random(T_1, T_2)$$
$$S = f(T) + Random(S_1, S_2)$$

This real-life problem is analogous to the example discussed earlier. As illustrated in Figure 2, the presence of dependence among decision variables modifies the variable search space. In case of no dependence among decision variables, T-S search space is rectangular in shape. It is shown as FRIV (Feasible Region with Independent Variables) in Figure 2. In presence of dependence among variables, the modified search space is shown as FRDV (Feasible Region with Dependent Variables) in Figure 2.

3 CHALLENGES POSED BY VARIABLE DEPENDENCE

Complex variable dependence poses a number of challenges for multi-objective optimisation algorithms. In the presence of variable dependence, the decision variables cannot be varied independently of each other. Also, the search space gets modified creating a new feasible region based on the dependence among decision variables. This is demonstrated in Figure 2. Depending upon the nature of variable dependency, additional features (such as bias (non-linearity), multi-modality, deception and discontinuity) may also be introduced in the problem. A generic Genetic Algorithm (GA) independently varies the decision variables and works in the feasible region that does not take variable dependence into account. So, it creates solutions that have limited practical significance since they do not lie in the actual feasible region of the search space. Therefore, there is a need to develop GAs that have mechanisms for handling variable dependence in their search processes.

4 TECHNIQUES FOR HANDLING VARIABLE DEPENDENCE

Most of the dependent-variable optimisation problems do not have known dependency relationships. In these problems, multiple sets of variable values are available from which the dependency relationships need to be inferred. An optimisation algorithm that is capable of handling variable dependence should be able to infer these relationships from the given data, identify the independent variables and manage the search process accordingly. Due to the lack of systematic research in this area, the literature in the field of optimisation does not report any dedicated technique that can deal with variable dependence. However, as shown in Table 1, the survey of literature in related areas of research reveals some techniques that can be used for inferring dependency relationships among decision variables and identifying independent variables.

Table 1: Techniques for Identification of Dependency Relationships and Independent Variables

Identification of Dependency Relationships	• Neural Networks (NNs) (Hertz et al., 1991; Richards, 1998; Gershenfeld, 1999) • Probabilistic Modelling (PM) (Pelikan et al., 1998; Larranaga et al., 1999; Evans and Olson, 2000; Muhlenbein and Mahnig, 1999) • Regression Analysis (RA) (Frees, 1996; Draper and Smith, 1998; Evans and Olson, 2000)
Identification of Independent Variables	• Tree Diagrams (TDs) (Banzhaf et al., 1998; Richards, 1998; Larranaga et al., 1999) • Direct Analysis (DA) (Gershenfeld, 1999)

4.1 IDENTIFICATION OF DEPENDENCY RELATIONSHIPS

Table 2 presents an analysis of the techniques that can be used for inferring dependency relationships from the avaiable sets of variable values. This table highlights the following.

- NNs: As can be seen from Table 2, the NNs require a priori knowledge regarding the classification of variables as dependent and independent (Hertz et al., 1991). Since this information is rarely available in real-life problems, the choice of the NNs is ruled out in spite of their other attractive features.

Table 2: Analysis of Techniques for Identification of Dependency Relationships

Comparative Analysis		**Techniques for Identification of Dependency Relationships**		
		Regression Analysis (RA)	**Neural Networks (NNs)**	**Probabilistic Modelling (PM)**
Features	**Difficulty of Implementation**	Medium	High	Very high (due to many open issues)
	Accuracy	Dependent on degree of RA equation	Dependent on number of hidden units	Dependent on choice of modelling method
	Computational Expense	Low	High	Medium
	Nature of Dependency Relationships	Explicit	Explicit (for given dependent variables)	Purely implicit
	Identification of Multiple Dependency Relationships	Multiple RA equations	Built-in multiple relationships (based on choice of NN structure)	Built-in multiple relationships
	Identification of Independent Variables	Through multiple repetitions of RA	Not possible	Not required
	Difficulty of Data Addition	Medium (repetition required)	Medium (repetition required by most NNs)	Low (updating required)

- PM: PM is also a very powerful technique, requiring little information regarding the nature of variables. As shown in Table 2, it also has a number of other features that are required for dealing with real-life problems. However, the application of PM to model multiple interacting decision variables is a relatively new area of research, and a number of research issues need to be addressed before it could be chosen for handling real-life problems having multiple real variables (Evans and Olson, 2000).
- RA: Table 2 reveals that the multiple explicit equations that are identified by the RA give good insight to the designer regarding the relationships among decision variables. RA is also easy to implement and maintain (Frees, 1996). Further, it addresses most of the above-mentioned limitations of NNs and PM. However, the accuracy of RA is dependent on its degree.

4.2 IDENTIFICATION OF INDEPENDENT VARIABLES

The main strengths and weaknesses of the techniques used for the identification of independent variables are the following.

- TDs: The dependence among decision variables can be graphically represented using TDs, in which each node represents a variable in the problem. TDs are easy to use and have good visualisation capabilities, but they are difficult to be encoded in a computer language.
- DA: DA involves the analysis of dependency equations to identify the independent variables. This method is easy to be encoded in a computer language but is difficult to visualise.

5 PROPOSED GA FOR VARIABLE DEPENDENCE (GAVD)

This section proposes a novel algorithm 'GA for Variable Dependence (GAVD)', described in Figure 3. Based on the discussion in Section 4, the RA is chosen in GAVD to identify variable dependency equations using the data provided. Furthermore, GAVD uses TDs for visualisation of dependency relationships, and DA to automate the identification of independent variables. The steps involved in GAVD are described below.

5.1 STEP 1: IDENTIFICATION OF DEPENDENCY RELATIONSHIPS

This step is omitted in those cases in which the dependency relationships are known. In the other cases, this step analyses the given data for identifying multiple dependency equations, while keeping the computational expense as low as possible. GAVD uses RA in such a way that it not only identifies all non-decomposable relationships among decision variables but also removes any cyclic dependency in those relationships. To attain this, a strategy that ensures good 'book keeping' is adopted. The salient features of this strategy are discussed below.

- The RA that is used in GAVD breaks down a regression equation until it becomes non-decomposable. In this way, all the underlying relationships among the decision variables are identified.
- A Dependency Chart (DC), which is a tool for DA, is maintained to keep track of the variables that are identified as dependent (D) and independent (I) in the regression process. In this way, unnecessary repetitions of RA are avoided for the variables that have already been identified as 'D' or 'I'. This also ensures that the regression equations do not involve any cyclic dependency.
- When determining the regression equation for a given variable, only those variables that are marked as 'I' or are unmarked in DC are considered as independent. This guarantees that the variables that are identified as 'D' are not considered as independent in subsequent stages of the RA, thereby ensuring that the regression equations obtained are as non-decomposable as possible. This also reduces the number of variables that are considered at each stage of the RA.

5.2 STEP 2: IDENTIFICATION OF INDEPENDENT VARIABLES

TDs are used here for visual representation of relationships among decision variables. A TD is constructed here to give a visual representation of the dependency relationships to the user. The end nodes of this tree are the independent variables. The TD also aids in the identification of cyclic dependencies that may be present in the given dependency equations. Since TDs are difficult to be encoded in a computer language, the DC is used to automate the process of identification of independent variables and remove any cyclic dependency. Here, the DC is used to identify the independent variables as those that are marked as 'I'. The construction of this chart also aids the identification and removal of cyclic dependencies from the dependency equations.

5.3 STEP 3: OPTIMISATION

Being a high-performing latest algorithm, Generalised Regression GA (GRGA) has been chosen as the optimisation engine for GAVD. GRGA is a multi-objective optimisation algorithm that uses RA for handling complex inseparable function interaction (Tiwari et al., 2001). Here, the independent variables, identified in the previous step, define the GA chromosome. For each alternative solution generated by the GA, the dependency equations are used to calculate the values of the dependent variables. It should be noted here that the bounds on independent variables are treated as variable

limits and those on dependent variables are treated as constraints.

Since GAVD uses GRGA as its optimisation engine, the basic operations of GRGA also form part of GAVD. In addition, it uses the RA to model the relationship among decision variables. Therefore, the overall computational complexity of GAVD is the complexity of GRGA increased with the complexity of the RA, where in most cases the latter is much smaller than the former.

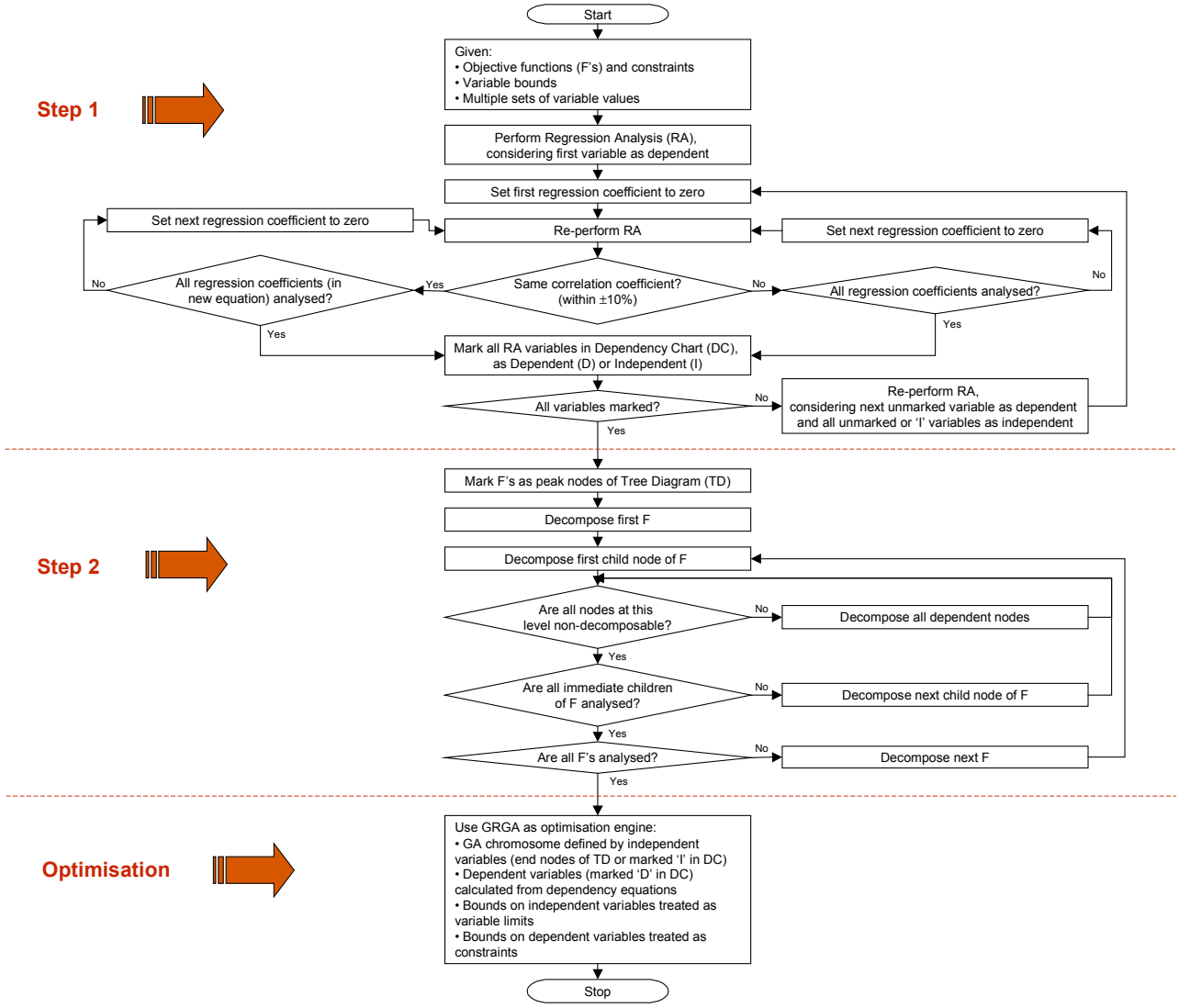


Figure 3: GA for Variable Dependence (GAVD)

5.4 A WORKED EXAMPLE

This worked example demonstrates the application of GAVD to a problem that has dependence among its decision variables. This problem is given below.

$$Objective\_Function: F = F(x_1, x_2, x_3, x_4, x_5)$$
$$\forall x_i^{(L)} \le x_i \le x_i^{(U)}, i = 1...5$$
$$Given: Multiple\_Sets\_of\_Variable\_Values$$

Suppose the underlying relationships among decision variables that need to be identified are as follows.

$$x_1 = f_1(x_2, x_3)$$
$$x_3 = f_2(x_2, x_4, x_5)$$

The flowchart of Figure 3 identifies the following steps for solving this problem.

- Determine the following equation for x_1.

  $$x_1 = v_1(x_2, x_3, x_4, x_5)$$

- No change is observed in correlation coefficient, when the RA is performed with the regression coefficient of x_2 set to zero. The new equation is as follows.

$$x_1 = v_1'(x_3, x_4, x_5)$$

- Correlation coefficients reduce, when the RA is performed with the regression coefficients of x_3, x_4 and x_5 set to zero in steps.
- Mark x_1 as 'D' and x_3, x_4 and x_5 as 'I' in the DC (Table 3).
- Determine the following equation for x_2 in terms of those variables that are so far identified as 'I' or are so far unmarked in the DC.

$$x_2 = v_2(x_3, x_4, x_5)$$

- Correlation coefficients reduce, when the RA is performed with the regression coefficients of x_3, x_4 and x_5 set to zero in steps.
- Mark x_2 as 'D' and x_3, x_4 and x_5 as 'I' in the DC (Table 3).
- The variables marked 'I' in the DC are independent whereas those marked 'D' are dependent.
- Use the dependency equations determined above for drawing the TD for the problem (Figure 4). The nodes that are encircled in this figure represent the independent variables. All other variables are treated as dependent.
- Use GRGA as the optimisation engine.

- *x_3, x_4 and x_5 constitute the GA chromosome.*
- *x_1 and x_2 are determined from the dependency equations.*
- *Bounds on x_3, x_4 and x_5 are treated as variable limits.*
- *Bounds on x_1 and x_2 are treated as constraints.*

Table 3: Dependency Chart (DC) for Worked Example

Dependency Chart (DC)		Variables				
		X_1	X_2	X_3	X_4	X_5
Regression Equations	X_1	D		I	I	I
	X_2		D	I	I	I
	X_3					
	X_4					
	X_5					

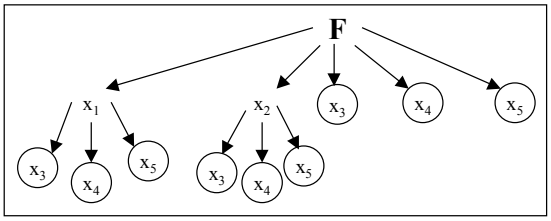


Figure 4: Tree Diagram (TD) for Worked Example

Table 4: Test Problems for Performance Analysis of GAVD

Problem	Objective Functions (Minimisation)	Dependency Equations
Problem-1	$D(\vec{x}') = \frac{1}{\left(1-\exp(-4)\right)}\left[1-\exp\left(-4x_1\right)\right], \forall 0 \le x_1 \le 1$ $I(\vec{x}'') = 2-\exp(-2x_2)\cos\left(8\pi x_2\right), \forall 0 \le x_2 \le 1$ $s(f_1, I) = 2-(f_1/I)^{0.6}$ $f_1 = D(\vec{x}')$ $f_2 = s(f_1, I) \times I(\vec{x}'')$	$x_2 = 1-0.1x_3-0.2x_3^2-0.3x_4-0.1x_4^2-0.3x_3x_4$ $\forall 0 \le x_3 \le 1, \forall 0 \le x_4 \le 1$ $Data\_Generation: x_2' = x_2 + Normal(0,0.05)$ (Figure 5(a))
Problem-2	$D(\vec{x}') = \frac{1}{\left(1-\exp(-3)\right)}\left[1-\exp\left(-3x_1\right)\right], \forall 0 \le x_1 \le 1$ $I(\vec{x}'') = 3-\exp(-x_2)\cos\left(2\pi x_2\right)-\exp(-x_3)\cos\left(4\pi x_3\right), \forall 0 \le x_2, x_3 \le 1$ $s(f_1, I) = 2-(f_1/I)^{0.4}-(f_1/I)\cos(8\pi f_1^2)$ $f_1 = D(\vec{x}')$ $f_2 = s(f_1, I) \times I(\vec{x}'')$	$x_2 = 0.2+0.2x_3+0.6x_3^2, \forall 0 \le x_3 \le 1$ $Data\_Generation: x_2' = x_2 + Normal(0,0.05)$ (Figure 5(b))

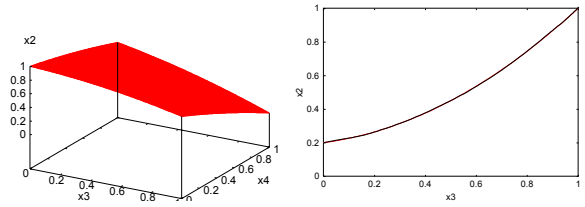
6 PERFORMANCE ANALYSIS

In this section, GAVD is tested using two multi-objective optimisation test problems that have dependence among their decision variables (Table 4). The features of these test problems make them particularly difficult for multi-objective optimisation algorithms. In the absence of any dedicated technique for handling variable dependence, this section compares the performance of GAVD with two high-performing multi-objective optimisation algorithms: NSGA-II and GRGA. However, unlike GAVD, both these algorithms do not take variable dependency into account.

6.1 EXPERIMENTAL RESULTS

All the tests reported here correspond to 100 population size, 500 generations, 0.8 crossover probability, 0.05 mutation probability, and simulated binary crossover with 10 crossover distribution index and 50 mutation

distribution index. The results obtained from these tests are shown in Figure 6 for Problem-1 and Figure 7 for Problem-2. The γ (convergence metric) and Δ (diversity metric) values corresponding to these results are shown in Table 5 (Deb et al., 2000). These results form the typical set obtained from 10 runs with different random number seed values. No major variation was observed in the results with the change in the seed values.



(a) (b)

Figure 5: Dependency Relationships (a) Problem-1 (b) Problem-2

Table 5: Performance Metrics in Problems 1 and 2

Performance Metrics		Problem-1		Problem-2	
		γ	Δ	γ	Δ
Optimisation Algorithms	NSGA-II	1.209567	0.090002	0.986345	0.083956
	GRGA	0.009143	0.080121	1.654703	0.045431
	GAVD	0.008221	0.081124	0.001373	0.014564

6.2 DISCUSSION OF RESULTS

The following observations can be made from the results obtained from Problem-1 (Figure 6, Table 5).

- Since the dependency equation covers the full range of x_2, it does not alter the Pareto front. Therefore, the Pareto fronts for the original problem (with no dependence) and the dependent-variable problem coincide with each other.
- GRGA and NSGA-II do not incorporate variable dependence in their solution strategies. However, since the original and the new Pareto fronts are coincident in this case, the GRGA is able to locate the Pareto front. However, NSGA-II gets trapped in one of the local fronts.
- The dependency equation is quadratic, making it possible for the GAVD (that uses quadratic RA) to exactly model the dependence. Hence, the Pareto front that the GAVD sees coincides with the true Pareto front. Furthermore, since GAVD uses GRGA as its optimisation engine, it is able to converge to the Pareto front and distribute the solutions uniformly across the front.

The following observations can be made from the results obtained from Problem-2 (Figure 7, Table 5).

- In this problem, the original Pareto front occurs when both x_2 and x_3 are equal to 0. Due to the given dependency among these variables, this is no longer possible. This causes modifications in the search space and the Pareto front.
- GRGA converges to the global Pareto front of the original problem (with no dependence among its decision variables). However, since the new Pareto front does not coincide with the original one, the results from GRGA are not feasible in this case. Similar to the previous case, NSGA-II gets trapped on a local front, which in this case coincidentally lies in the new search space.
- Also, since GAVD uses quadratic RA, it is able to exactly determine the dependency equation in this case. Hence, the Pareto front seen by GAVD is the same as that of the actual dependent-variable problem. Therefore, GAVD converges to the Pareto front and distributes the solutions uniformly across the front.

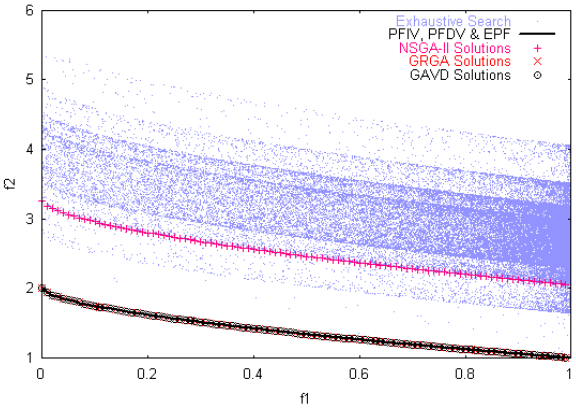


Figure 6: GAVD Performance in Problem-1 (PFIV: Pareto Front for Independent Variables, PFDV: Pareto Front for Dependent Variables, EPF: Estimated Pareto Front)

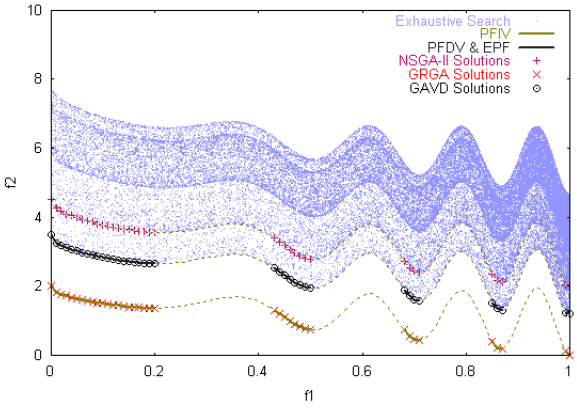


Figure 7: GAVD Performance in Problem-2 (PFIV: Pareto Front for Independent Variables, PFDV: Pareto Front for Dependent Variables, EPF: Estimated Pareto Front)

7 FUTURE RESEARCH ACTIVITIES

The current limitations of GAVD and the corresponding future research activities are as follows.

- The performance of this algorithm in identifying the dependence among decision variables is limited by the degree of RA that it uses. Hence, in dealing with complex dependence, higher order RAs are required. This implies that the use of more sophisticated non-linear modelling tools, such as Neural Networks, have the potential of improving its performance, especially in modelling deceptive and complex non-linear functions.
- GAVD also needs to be fitted with a mechanism that can learn the dependency relationships, and update it each time a new data is added, without having to repeat the whole process.
- GAVD also needs enhancements to deal with noisy data and qualitative issues in real-life problems.

8 CONCLUSIONS

There is currently a lack of systematic research in the field of variable dependence. This paper proposes an algorithm capable of handling variable dependence in multi-objective optimisation problems. The performance of proposed algorithm is compared to that of two state-of-the-art optimisation algorithms (NSGA-II and GRGA) using two dependent-variable test problems. It is observed that the proposed algorithm GAVD enables its optimisation engine (GRGA) to handle variable dependence in optimisation problems.

Acknowledgements

The authors wish to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) – Grant No. GR/M 71473, Nissan Technical Centre – Europe (NTC-E) and Structural Dynamics Research Corporation (SDRC) UK.

References

Banzhaf, W., Nordin, P., Keller, R.E., and Francone, F.D. (1998). *Genetic programming: An introduction.* Morgan Kaufmann Publishers, Inc., San Francisco, California (USA).

Beasley, D., Bull, D. and Martin, R. (1993). An overview of genetic algorithms: Part 2, research topics. *University computing*, vol. 15, no. 4, 170-181.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2000). *A fast and elitist multi-objective genetic algorithm: NSGA-II.* KanGAL Report No. 200002, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology (IIT), Kanpur (India).

Draper, N.R. and Smith, H. (1998). *Applied regression analysis.* John Wiley and Sons, Inc., New York (USA).

Evans, J.R. and Olson, D.L. (2000). *Statistics data analysis, and decision modelling.* Prentice Hall (USA).

Frees, E.W. (1996). *Data analysis using regression models: The business perspective.* Prentice Hall (USA).

Gershenfeld, N. (1999). *The nature of mathematical modelling,* Cambridge University Press, Cambridge (UK).

Harik, G.R. (1997). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms.* PhD. Thesis, Computer Science and Engineering, University of Michigan (USA).

Hertz, J.A., Krogh, A.S., and Palmer, R.G. (1991). *Introduction to the theory of neural computation.* Addison-Wesley, Redwood City, CA (USA).

Larranaga, P., Etxeberria, R., Lozano, J.A., and Pena, J.M. (1999). *Optimization by learning and simulation of Bayesian and Gaussian networks.* Technical Report No. EHU-KZAA-IK-4/99, Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country (Spain).

Muhlenbein, H. and Mahnig. T. (1999). FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary computation,* vol. 7, no. 4, 353-376.

Pedrycz, W. (1998). *Computational intelligence – an introduction.* CRC Press, New York (USA).

Pelikan, M., Goldberg, D.E., and Cantu-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks.* IlliGAL Report No. 98013, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (USA).

Phadke, M.S. (1989). *Quality engineering using robust design.* Prentice-Hall International Inc., London (UK).

Rao, S.S. (1996). *Engineering optimization – theory and practice.* Wiley-Interscience, USA.

Richards, W. (1998). *Natural computation.* MIT Press, Cambridge, MA (USA).

Roy, R., Tiwari, A., Munaux, O. and Jared. G. (2000). Real-life engineering design optimisation: features and techniques. In: Martikainen, J. and Tanskanen, J. (eds.). *CDROM Proceedings of the 5th online world conference on soft computing in industrial applications (WSC5)* – ISBN 951-22-5205-8, IEEE (Finland).

Taguchi, G. (1987). *System of experimental design.* Clausing, D. (ed.), UNIPUB/Kraus International Publications, vol. 1 and 2, New York (USA).

Tiwari, A., Roy, R., Jared, G. and Munaux, O. (2001). Interaction and multi-objective optimisation. In: Spector, L., Goodman, E., Wu, A., Langdon, W.B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. and Burke, E. (eds.). *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001),* 671-678, Morgan Kaufmann Publishers, San Francisco (USA).

Applying Genetic Algorithms to Finding the Optimal Gene Order in Displaying the Microarray Data

Huai-Kuang Tsai
Dept. of Computer Science and Information Engineering National Taiwan University, Taipei, Taiwan
d7526010@csie.ntu.edu.tw

Jinn-Moon Yang
Dept. of Biological Science and Technology & Institute of Bioinformatics, National Chiao Tung University, Hsinchu, Taiwan
moon@cc.nctu.edu.tw

Cheng-Yan Kao[1,2]
[1] Dept. of Computer Science and Information Engineering National Taiwan University, Taipei, Taiwan
[2] Bioinformatics Center, National Taiwan University, Taipei, Taiwan
cykao@csie.ntu.edu.tw

Abstract

In this paper the *Family Competition Genetic Algorithm* (FCGA) is applied to analyze DNA-microarray data. DNA Microarray technology is a significant impact on genomics study. The proposed approach consists of global and local strategies by integrating the family competition, edge assembly crossover, and neighbor-join mutation. Experiments are performed to compare the FCGA with several methods in some real-world biological data sets. Numerical results indicate that FCGA performs very robustly and is very competitive with other approaches. Using FCGA, we are able to find a gene order to display the microarray data in a meaningful way.

1 INTRODUCTION

DNA microarray technology can be applied to many biological domains, such as drug discovery, molecular diagnosis, and toxicological research. During the past few years, the development of DNA-microarray technology had provided the means to monitor the expression levels of a large number of genes simultaneously.

In the microarray experiments, messenger RNAs (mRNA) are extracted from the cell culture. Complementary DNAs (cDNA) are generated from the RNAs, amplified, labeled and then hybridized to a large array of DNA probes immobilized on a solid surface. The array is then scanned by a laser to obtain the signal for each probe region. From the signal strengths of the probes from a particular gene, one can infer the expression level of the gene in the cell type under study. Fig. 1 is the schematic procedures for monitoring gene expression using DNA microarray. With many chips, the expression data can be represented by a real-valued expression matrix X where X_{ij} is the measured expression level of gene i in experiment j.

However with thousands of genes and hundreds of experiments, it is difficult to evaluate the immense amount of gene expression profiles. A large number of approaches have been developed for analyzing the huge microarray data. For examples, clustering, classification, and genetic network analysis are usually adapted for analyzing these data. In any case, it is important to display microarry data in a meaningful way to best illustrate trends in gene expression.

An intuitive way to display microarray data is to find an optimal order of genes such that genes with similar expression profiles are blocked together. However, it is NP-complete to find an optimal order of genes [1]. Several approaches have been proposed for solving this problem. For example, the hierarchical clustering approach, a widely used tool [2-6], has been used to approximate the solution. Since the constructing process of the hierarchical tree is greedy, this approach may get stuck at local minima. Some approaches have been proposed to improve the solution quality of hierarchical clustering approach, such as flipping the internal nodes in the tree [7] and neural networks [8]. In this paper, finding an optimal order of genes is formulated as a travel salesman problem (TSP). Evolutionary approaches (EAs) are one of promising directions for solving TSPs.

Evolutionary approaches have been successfully applied to optimization problems that are inherently computationally complex [9-11]. EAs are an adaptable concept for problem solving and especially well suited for solving difficult optimization problems. They have been used to solve problems involving large search spaces, where traditional optimization methods are less efficient.

In this paper, we propose the *family competition genetic algorithm* (FCGA) to find the optimal order of genes with expression profiles. The FCGA combines a family competition, the neighbor-join mutation (NJ), and the edge assembly crossover (EAX) [12]. The family competition, derived from (1+λ)-ES and Lin-Kernigan heuristic, had been successfully applied to several continuous parameter optimization problems, such as protein docking [13] and thin-film coatings [14]. In our pervious studies [15], we had successfully integrated the family competition and EAX for solving traveling salesman problems (TSPs). In order to balance exploration and exploitation, we also designed the neighbor-join mutation [16] to cooperate with the EAX. The main difference in methodology between the present work and our previous studies is the integrations of these mechanisms.

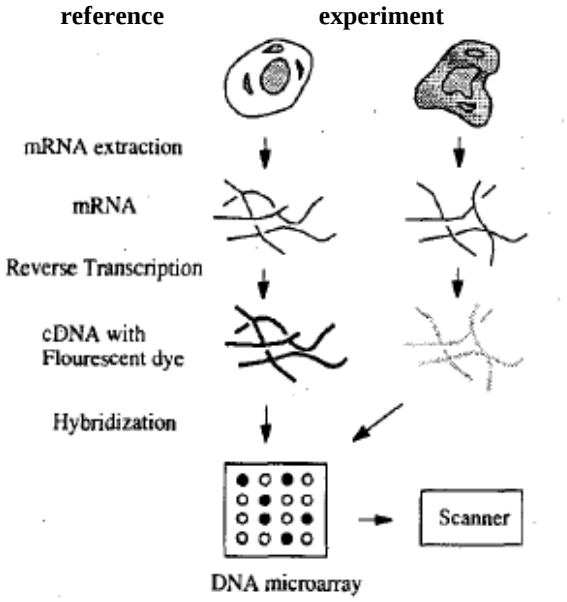


Figure 1. Schematic procedures for monitoring gene expression using DNA microarray

We illustrate features of FCGA by some TSPs benchmarks and biological data sets. The TSPs were used to verify the performance of FCGA by comparing with several methods [12][17-19]. Three biological data sets are tested to shown that FCGA is superior to the existing heuristic methods of gene order, including hierarchical clustering [2] and self-organizing map (SOM) [20]. Experimental results demonstrated that the FCGA is an encouraging approach for finding the optimal order of genes in expression profiles.

This paper is organized as follows. Section 2 describes the problem of ordering genes in expression profiles. Section 3 introduces the evolutionary nature of the FCGA. In Section 4, some experimental results are presented to illustrate the performance of the FCGA. We also compare the FCGA with various approaches on three biological problems and discuss the biological meanings. Concluding comments are drawn in Section 5.

2 PROBLEM DEFINITION

One important issue in the microarray data analysis is to display the data in a meaningful way that best illustrates the trends in gene expression. The problem can be formulated as follows: find an optimal order of genes such that genes with similar expression profiles are close together. Different criteria result in different objective functions: such as distances between gene expression profiles [1] or distances between both adjacent genes and block similarities [21]. In this paper, we used the sum of distances of adjacent genes as our fitness function defined as

$$\sum_{i=1}^{M} D(g_{\pi_i}, g_{\pi_{i+1}}), \tag{1}$$

where g_i denote a gene, $1 \le i \le n$, π denote a gene order, M is number of genes and $D(g_i, g_j)$ is the distance of two genes g_i and g_j. This problem is the same as to determine the shortest route passing through a set of M cities under the condition that each city is visited exactly once. This so-called traveling salesman problem is well known to be NP-complete [22].

Some methods have been proposed to define the distance $D(g_i, g_j)$, or called similar, between two genes, such as *Pearson* correlation, *absolute* correlation, *Spearman Rank* correlation, *Kendall's Tau*, and *Euclidean* distance. In this paper, we applied *centered Pearson correlation* which is widely used in DNA microarray. Let $X = x_1, x_2, ..., x_k$ and $Y = y_1, y_2, ..., y_k$ be the expression levels of two genes (prepared in log-transformed data) observed over a series of k conditions. Based on *Pearson correlation* the distance of genes X and Y can be given

$$D(X, Y) = 1 - s_{X,Y}. \tag{2}$$

$s_{X,Y}$ is the centered Pearson correlation defined as

$$s_{X,Y} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{x_i - \overline{X}}{\sigma_X}\right)\left(\frac{y_1 - \overline{Y}}{\sigma_Y}\right) \tag{3}$$

where $\overline{X}$ and σ_X is the mean and standard derivation of the expression levels. The value of σ_X is

$$\sigma_X = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x_i - \overline{X})^2}. \tag{4}$$

According to the above steps, the problem finding an optimal order of genes can be formulated as a TSP. Then we applied our method to solve this problem.

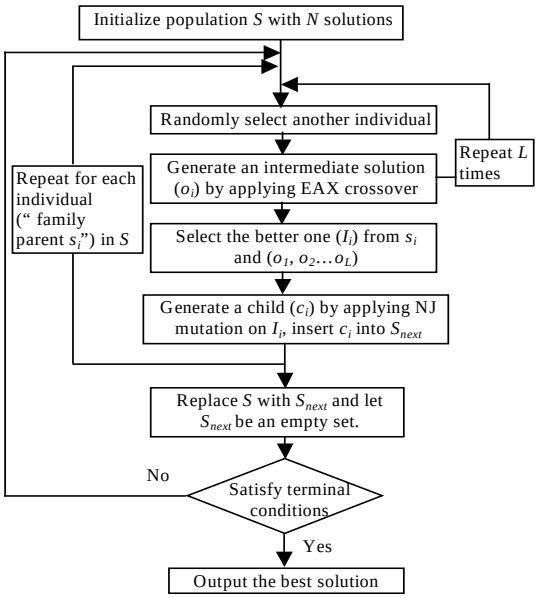


Figure 2. The outline of FCGA

3 METHOD

In this section, the details of the proposed genetic algorithm, called family competition genetic algorithm (FCGA), for optimizing the gene order in gene expression data are presented. The FCGA has three major mechanisms, including a family competition, the edge assembly crossover (EAX), and the neighbor-join mutation (NJ). The EAX and the NJ mutation are genetic operators considered to be able to preserve and add good edges to generate a child. The family competition is a local search mechanism incorporated into the EAX and the NJ mutation. These three mechanisms have been studied to balance exploration and exploitation in the search space.

Fig. 2 shows the main steps of the FCGA. N solutions are generated as the initial population. Each solution is represented as a random permutation from 1 to M where M is the number of genes. After evaluating the fitness, each individual in the population sequentially becomes the "family father (s_i)" to produce L offspring, (o_1,...,o_L), by conducting the EAX and the family competition. The one with lowest fitness value from o_1,o_2...o_L and s_i becomes the intermediate offspring (I_i). The NJ is then applied to generate a child (c_i) by refining from the intermediated solution I_i. Each individual in the population sequentially executes the above steps to generate its child. These N solutions (c_1,...,c_N) become the new population of the next generation. Therefore, LN solutions are generated in one generation and N solutions are selected as the parent population of the next generation.

Our algorithm terminated when one of criteria is satisfied: 1) the maximum preset search time is exhausted, 2) all individuals of a population are the same, or 3) all of the children generated in continuous five generations are worse than their parents. Please note that both the crossover and mutation rates are 1.0. In the following subsections, the family competition, the EAX, and the NJ mutation are described.

3.1 REPRESENTATION

In the chromosome representation of our FCGA, each solution s_i represents a gene order π, where $1 \le i \le N$ and N is the population size. Assume there are M genes $\{g_1,\ldots,g_M\}$, the solution s_i is represented as:

$$s_i = (g_{\pi_1}, g_{\pi_2}, ..., g_{\pi_M}). \quad (5)$$

The fitness function follows the equation (1).

3.2 FAMILY COMPETITION

The family competition, derived from (1+λ)-ES and Lin-Kernigan heuristic, is considered as a local search procedure in FCGA. In the family competition step, L offspring, (o_1,...,o_L), are generated by EAX crossover operator and after family selection, the one with best fitness from (o_1,...,o_L) and the family father (s_i) is survived. The procedure of the family competition is described as follows. Each individual (s_i) sequentially becomes the "family father." This "family father" and another solution (s_j) randomly chosen from the rest of the parent population are used as parents to do EAX crossover operation to generate an offspring (o_l). For each family father, such a procedure is repeated L times. Finally L solutions (o_1,...,o_L) are produced. After L solutions compete with "family father," only the one (I_i) with the best objective value survives. Since we create L solutions from the same "family father" and perform a selection, this is a family competition strategy. Because each individual sequentially becomes the "family father", LN offspring are generated in one generation.

3.3 EDGE ASSEMBLY CROSSOVER

The EAX [10] is considered a powerful crossover operator [23]. It has two important features: preserving parents' edges with a novel approach and adding new edges with a greedy method, analogous to a minimal spanning tree. Several issues, such as the selection mechanism and heuristic methods, influencing EAX performance have been discussed [15][16][23][24]. In this paper the EAX is considered as the global search strategy in our proposed algorithm.

The EAX is briefly described here. Two individuals, denoted as A and B, were selected as the parents. The EAX first merges A and B into a single graph denoted G. The EAX travels G to generate many *AB-cycles* by alternately picking edges from parents A and B. According to the heuristic and random selection rules, some of *AB-cycles* are selected to generate a quasi solution which contains some disjointed subtours. Then, the EAX uses a greedy method to merge these disjointed subtours into a valid solution. This solution is returned if the fitness of this solution is better than its parents. Otherwise this procedure is repeated until a solution that is better than both A and B or K children are produced where K is the local search length.

3.4 NEIGHBOR-JOIN MUTATION

The neighbor-join (NJ) operator constructs a new solution by stealing edges from other individuals in the population or by considering the geometric information. Although the NJ is applied only on the single solution, the offspring is generated considering both the neighborhood information and knowledge from other individuals. Thus, the NJ operator is a genetic operator combining with the characteristics of local search, mutation, and recombination.

The NJ is inspired by the inver-over mutation [25] and by analyzing the TSP search space [26]. The main difference between the inver-over mutation and the other mutations is that it inherits edges both from parent and from other individuals in the current population. According to the analysis of the optimal tour of *att532*,

we find that most of the links in the optimal tour of *att532* are the neighbor cities of each city.

The details of the NJ are described as follows. By given the input of an individual I_i and the search length K, the NJ applies K modifications from the start solution I_i'= I_i. In each modification, a gene c is randomly selected from I_i'. With equal probability, a gene c' is randomly selected either from the geometric nearest three neighbors of c or from the neighbor of c of an individual, which is randomly selected from the population. If the edge (c',c) does not appear in I_i', to reconnect c and c' together generates four possible types. The NJ generates four candidates by sequentially executing each type once. The one with lowest fitness from these four candidates and I_i' are selected as the parent of next loop. Above steps are executed K times.

In the four candidates, two are the simple invert operation to align c and c' together, the other two will result in two disjoint subtours. A greedy method is applied to merge two disjoint subtours into a valid solution. The greedy method works as follows: Let v_i represent a gene, (v_i, v_j), $i \neq j$, represents an edge, and $w(v_i, v_j)$ be the edge length of (v_i, v_j). At the same time, let (v_r, v_{r+1}) and (v_s, v_{s+1}) be the edges of the subtour G_r and the subtour G_s, respectively. We find a pair of edges (v_r, v_{r+1}) and (v_s, v_{s+1}) to connect these two subtours, G_r and G_s, into a legal tour by maximizing the value of the following equation:

$$w(v_r, v_{r+1}) + w(v_s, v_{s+1}) - w(v_r, v_{s+1}) - w(v_s, v_{r+1}) \quad (6)$$

$$\forall r, s;\ r \in G_r \text{ and } s \in G_s .$$

The new edges (v_r, v_{s+1}) and (v_s, v_{r+1}) are inserted to replace the original edges (v_r, v_{r+1}) and (v_s, v_{s+1}) to form the new solution.

4. EXPERIMENTAL RESULTS

In this section FCGA first was tested on some TSP benchmarks to verify the correctness and efficiency. Four efficient methods for TSPs were compared with FCGA to show the robustness of FCGA. FCGA is then applied on three biological data sets to find the optimal gene order. By comparing to the hierarchical clustering [2] and self-organizing map (SOM) [20], FCGA is superior to other approaches in: 1) minimizing the cost of order, 2) uncovering the correct cell cycle, and 3) most genes with the same group are aligned together. Finally we conclude this section by presenting biological results with visualized representation.

FCGA has been implemented in C++ and executed on a Pentium III 500MHz personal computer with single processor. As introduced in Section 3, the population size (N), the family competition length (L), and the local search length (K) are the main parameters in our algorithm. According to our previous study [15][16], the population size (N) is roughly set to the number of cities (for TSP problem) and the number of genes (for microarray data), while the family competition length (L) is set to 5 and the local search length (K) is set to 20 for the tradeoff between solution quality and convergence time.

4.1 RESULTS OF STSP PROBLEMS

Table I summarizes the results of our method and four other approaches, including nature crossover genetic algorithm (NGA) [18], ant colony system (ACS) [19], distance-preserving crossover genetic algorithm (DGA) [17], and EAX genetic algorithm (EGA) [12]. NGA integrated nature crossover and LK local search [27]; ACS is an ant colony system cooperated with 3-opt operator; DGA combined the distance-preserving crossover and 3-opt; and EGA used the EAX crossover. These four approaches perform well on these test problems according to our surveys. The results of first three methods were directly summarized from original papers. The average tour length and the average error of trails are used to measure the performance of comparative methods. The values in parentheses of the average tour length represent the percentages of error defined as $\frac{sol - optimum}{optimum}$, where *sol* is the experimental value and *optimum* is the optimum of a TSP problem.

Table I shows that our algorithm performs robustly for testing symmetric TSPs. For each problem the proposed algorithm can find the best tour in almost each trial and the error rate is only 0.01% away from the optimal. Since the solution qualities of FCGA applied on these TSPs are good, we believe that it would also proper to optimize the gene order in gene expression data.

TABLE I

Comparisons of FCGA with other methods, including NGA [18], ACS [19], DGA [17], and EGA[12], on five TSP problems based on the average tour length and average solution qualities (error) in 30 trails. The percentages of error defined as $\frac{sol - optimum}{optimum}$, where *sol* is the experimental value and *optimum* is the optimum of a TSP problem. "N/A" represents not available in original papers.

Problems/ (optimum)	Methods: average tour length (error in %)				
	FCGA	ACS	DGA	NGA	EGA
Lin318 (42029)	42029.00 (0.000)	N/A	42033.44 (0.011)	42029.00 (0.000)	42041.23 (0.011)
pcb442 (50778)	50778 (0.000)	N/A	50778 (0.000)	50778 (0.000)	50778 (0.000)
att532 (27686)	27688.49 (0.0081)	27718.20 (0.112)	27697.58 (0.042)	27695.61 (0.035)	27696.33 (0.037)
rat783 (8806)	8806.00 (0.000)	8837.90 (0.362)	8806.00 (0.000)	8806.00 (0.000)	8806.00 (0.000)
pcb3038 (137694)	137700.19 (0.0032)	N/A	137760.55 (0.048)	137765.02 (0.052)	137703.77 (0.007)

4.2 RESULTS OF BIOLOGICAL DATA

After the robustness of FCGA is shown, we applied it to three biological data sets to find the optimal gene order.

FCGA was compared with four widely used methods, including hierarchical agglomerative clustering algorithm (single-linkage, complete-linkage, and average-linkage [2]) and self-organizing map (SOM) [20].

In the aspect of data, the first and second data set, cell cycle-cdc15 and cell cycle, are about 800 genes which are cell cycle regulated in *saccharomyces cerevisia* with deferent number of experiments [28]. Spellman et al. [28] assigned these 800 genes to five groups termed *G1*, *S*, *S/G2*, *G2/M*, and *M/G1*. These groups approximate the commonly used cell cycle groups in the literature. The authors used a '*phasing*' method which compare the '*peak expression*' for each unknown gene with the expression of genes that were known to belong to each of these group. Although the group assignment is not the real grouping, it is still meaningful to some degree. So, we also use this information to evaluate a order of genes. The third data set, yeast complexes, is from MIPS yeast complexes database [2]. All these three data sets can be found in [7]. Table II gives the brief descriptions of each data set.

TABLE II

The description of three tested biological data set, including the source, number of experiments, and number of genes of each set.

Data name	source	Num. of experiment	Num. of genes
Cell cycle cdc15	Spellman et al. [28]	24	782
Cell cycle	Spellman et al. [28]	59	803
Yeast complexes	Eisen et al. [2]	79	979

All these data can be download from [7]

Two scoring systems are adapted here to verify the correctness of a gene order π. Assume $\pi = (g_{\pi_1}, g_{\pi_2} \ldots g_{\pi_M})$ is an order of genes, where M is the number of genes. The first score is the fitness function which is the sum of the distance between any two consecutive genes in π, denoted as $score_1(\pi)$. The score $Score_1(\pi)$ is defined as:

$$Score_1(\pi) = \sum_{i=1}^{M} D(g_{\pi_i}, g_{\pi_{i+1}}), \quad (7)$$

where $g_{\pi_{M+1}} = g_{\pi_1}$, $D(g_{\pi_i}, g_{\pi_{i+1}})$ is the distance between two genes (the distance measures are defined in section 2). In fact, this is just the fitness function (equation 1). The smaller the $score_1(\pi)$ is, the better order of genes we would get.

The second score, $score_2(\pi)$, is to measure the overall group distribution in π. As mentioned earlier in this section, the first and second data have descriptions about gene group information. By this information, the score $Score_2(\pi)$ is defined as:

$$Score_2(\pi) = \sum_{i=1}^{M} G(g_{\pi_i}, g_{\pi_{i+1}}), \quad (8)$$

where $g_{\pi_{M+1}} = g_{\pi_1}$, and $G(g_{\pi_i}, g_{\pi_{i+1}})$ is defined as:

$$G(g_{\pi_i}, g_{\pi_j}) = \begin{cases} 1, \text{if } g_{\pi_i} \text{ and } g_{\pi_j} \text{ are in the same group} \\ 0, \text{if } g_{\pi_i} \text{ and } g_{\pi_j} \text{ are not in the same group} \end{cases}$$

In a gene order π, if genes with the same groups are aligned next to each other, $score_2(\pi)$ would be higher. In summary, we use FCGA to get the optimal gene order by minimizing the fitness function $score_1(\pi)$. After the optimal gene order π is got, we hope the magnitude of $score_2(\pi)$ as larger as possible. Fig.3 shows an example of calculating the $score_1(\pi)$ and $score_2(\pi)$.

Table III and IV summarizes the comparisons on $score_1(\pi)$ and $score_2(\pi)$ of our method and these four approaches. Both tables show that the FCGA performs more robustly than comparative methods for testing sets in both $score_1(\pi)$ and $score_2(\pi)$. By counting the factor:

$$\frac{score_2(\pi)}{number\ of\ genes}, \quad (9)$$

we found that almost 70-80% genes with the same groups are aligned next to each other. In other words, two neighbor gene in the optimal gene order π are almost in the same group. In summary, FCGA provides a way to reorder the genes in a meaningful order and aligns genes with the same group together.

Gene order: π = (2,3,6,1,7,4,5)

$score_1(\pi) = D(2,3)+D(3,6)+ D(6,1)+D(1,7) + D(7,4)+D(4,5)$

$= 0.5+0.5+0.6+0.4+0.5+0.4+0.8 = 3.7$

$score_2(\pi) = G(2,3)+G(3,6)+ G(6,1)+G(1,7) +G(7,4)+G(4,5)$

$=1+0+0+0+0+1+0 = 2$

distance matrix

	1	2	3	4	5	6	7
1	0	0.5	0.3	0.2	0.5	0.6	0.4
2	0.5	0	0.5	0.3	0.8	0.3	0.2
3	0.3	0.5	0	0.4	0.4	0.5	0.2
4	0.2	0.3	0.4	0	0.4	0.5	0.5
5	0.5	0.8	0.4	0.4	0	0.6	0.6
6	0.6	0.3	0.5	0.5	0.6	0	0.3
7	0.4	0.2	0.2	0.5	0.6	0.3	0

group category

gene	Group
1	1
2	1
3	1
4	2
5	2
6	3
7	3

Figure 3. An example of calculating the $score_1(\pi)$ and $score_2(\pi)$.

Table III

Comparisons of our method (FCGA) with other methods including single-linkage, complete-linkage, average-linkage [2], and self-organizing map (SOM) [20], on three gene expression data sets based on $\boldsymbol{score_1}(\pi)$.

	Cell cycle cdc15	Cell cycle	Yeast complexes
FCGA	137.347	219.233	308.801
Single-linkage	655.483	599.329	621.311
complete-linkage	227.828	486.717	435.314
average-linkage	244.792	398.15	459.529
SOM	363.453	530.635	623.169

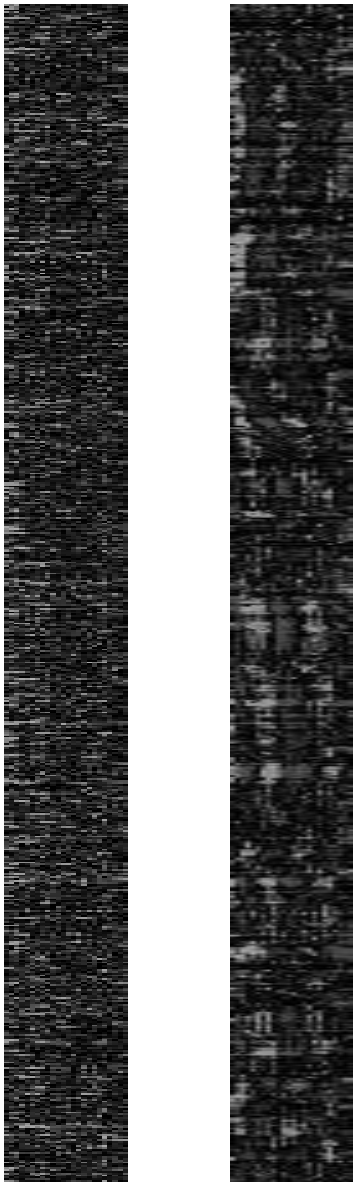
Table IV

Comparisons of our method (FCGA) with other methods including single-linkage, complete-linkage, average-linkage [2], and self-organizing map (SOM) [20], on two gene expression data sets based on $\boldsymbol{score_2}(\pi)$.

	Cell cycle cdc15	Cell cycle
FCGA	521	627
Single-linkage	251	336
complete-linkage	498	598
average-linkage	500	581
SOM	461	578

4.3 VISUALIZED RESULTS

To further understand the efficiency of our approach, we show the results by a visualized graph. Fig.4 is the gene expressions of the cell cycle cdc15 data [28] whose gene order is reordered by 1) original (random permutation) and 2) FCGA. For each gene in the figure, the expression profiles are represented as lines of color boxes and each box corresponding to one experiment. Comparing to the original data (random permutation), genes with similar expression profiles are grouped together by using FCGA. Some genes are not connected to their groups because: 1) the global minimization forces some genes to separate from their original group; 2) the missing values and the distance metric affect the overall ordering of genes. (more data results are available at http://bioinfo.csie.ntu.edu.tw/~survivor/ordering.)



(1)original **(2)FCGA**

Figure 4. The visualized gene expressions results. This figure shows the gene order of the cell cycle cdc15 data whose gene order is 1) original (random permutation) and 2) reordered by FCGA. For each gene, the expression profiles are represented as lines of color boxes and each box corresponding to one experiment. As we can see, (2) is more organized and most neighbor genes in the order have similar expression profiles.

5 CONCLUSION

This study presents that FCGA has successfully applied to solve the problem of displaying the microarray data in an optimal order. FCGA keeps the population diversity via the family competition and efficiently search

the solution space via incorporating EAX crossover and NJ mutation. Experiments of the TSPs verify that the proposed approach is very comparative with other evolutionary algorithms. Using FCGA on the biological data can recover the correct cell cycle and group similar genes in an optimal order. We believe that the flexibility and robustness of the FCGA make it an effective tool of analyzing microarray data.

In the future, we will: 1) test more biological data set to reveal new biological facts; 2) use different distance metrics to produce better results; and 3) investigate different objective functions for nonparametric clustering via FCGA.

References

[1]. Biedl, T., Brejova, B., Demaine, E. D., Hamel, A. M., and Vinai, T., "Optimal Arrangement of leaves in the tree representing hierarchical clustering of gene expression data," Technical report, Nov. 2001.

[2]. Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D., "Cluster analysis and display of genome-wide expression patterns," *Proc. Natl. Acad. Sci.*, pp. 14863–14868, 1998.

[3]. Alizadeh, A. A., Eisen, M. B., et al., "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, 403(6769), pp. 503-511, 2000.

[4]. Kawasaki, S., Borchert, C., et al., "Gene expression profiles during the initial phase of salt stress in rice," *Plant Cell*, 13(4), pp. 889-906, 2001.

[5]. Khodursky, A. B., Peter, B. J., et al., "DNA microarray analysis of gene expression in response to physiological and genetic changes that affect tryptophan metabolism in Escherichia coli," *Proc. Natl. Acad. Sci.*, pp. 12170-12175, 2000.

[6]. Schaffer, R., Landgraf, J., et al., "Microarray Analysis of Diurnal and Circadian-Regulated Genes in Arabidopsis," *Plant Cell*, 13(1), pp. 113-123, 2001.

[7]. Bar-Joseph, Ziv., Gifford, D. K., and Jaakkola, T. S., "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol. 17, suppl. 1, pp. s22-29, 2001, *http://www.psrg.lcs.mit.edu/clustering/ismb01/optimal.html*.

[8]. Herrero, J., Valencia, A., and Dopazo, J., "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, pp. 126-136, 2001.

[9]. Goldberg, D. E., *Genetic algorithms in search, optimization & machine learning.* Reading, MA: Addison-Wesley, 1989.

[10]. Chu, P. C. "A Genetic Algorithm for the Multidimensional Knapsack Problem," *Journal of Heuristics*, vol. 4, pp. 63-86,1998.

[11]. Dandekar, T. and Argos, P., "Folding the main chain of small proteins with the genetic algorithm," *J. Mol. Biol.*, vol. 236, pp. 844- 861, 1994.

[12]. Nagata, Y. and Kobayashi, S., "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in *Proceeding of the seventh international Conference on Genetic Algorithms* (*ICGA*), 1997, pp. 450-457.

[13]. Yang, J. M. and Kao, C.Y. "A Family competition evolutionary algorithm for automated docking of flexible ligands to Proteins," *IEEE Trans. on Information Technology in Biomedicine*, vol. 4, no. 3, pp. 225-237, 2000.

[14]. Yang, J. M., Horng, J.T, Lin, C. J., and Kao, C.Y. "Optical coating designs using an evolutionary algorithm," *Evolutionary Computation*, vol. 9, no.4, pp. 421-443, 2001.

[15]. Tsai, H. K., Yang, J. M., and Kao, C. Y. (2001) "A genetic algorithm for traveling salesman problems," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pp.687-693.

[16]. Tsai, H. K., Yang, J. M., and Kao, C. Y., "Solving Traveling Salesman Problems by Combining Global and Local Search Mechanisms," *Proceedings of the Congress on Evolutionary Computation* (CEC), 2002, *to appear*.

[17]. Dorigo, M. and Gambardella, L. M. (1997) "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. on Evolutionary Computation*, vol.1, no.1, pp53-66.

[18]. Freisleben, B. and Merz, P. (1996) "New genetic local search operators for the traveling salesman problem," *In Parallel Problem Solving from Nature IV*, Springer-Verlag, pp. 890-899.

[19]. Jung, S. and Moon B. R. (2000) "The nature crossover for the 2D Euclidean TSP," *Genetic and Evolutionary Computation Conference (GECCO 2000)*, pp. 1003-1010.

[20]. Tamato,P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., LANDER, E. S., and GOLUB., T. R., "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation," *Proc. Natl. Acad. Sci.*, vol 96, pp. 2907-2912, 1999.

[21]. Amir, B. D., Ron, S., and Zohar, Y. "Clustering Gene Expression Patterns," *journal of computational biology*, vol. 6, pp. 281-297, 1999.

[22]. Garey, M. R. and Johnson, D. S., “Computers and Intractability: A Guide to the Theory of NP-Completeness,” *Freeman*, 1979.

[23]. Watson, J., Ross, C., Eisele, V., Denton, J., Bins, J., Guerra, C., Whitely, D., and Howe, A. (1998) “The traveling salesrep problem, edge assembly crossover, and 2-opt,” *In Parallel Problem Solving from Nature V, A. E. Eiben et al, eds. Springer-Verlag*, pp.823-832.

[24]. Nagata, Y. and Kobayashi, S. (1999) “An analysis of edge assembly crossover for the traveling salesman problem,” *IEEE International Conference on Systems Man and Cybernetics*, pp. 628-633.

[25]. Tao, G. and Michalewicz, Z. (1998) ”Inver-over Operator for the TSP,” *In Parallel Problem Solving from Nature V*, Springer-Verlag, pp.803-812.

[26]. Padberg, M. and Rinaldi, G. (1987) “Optimization of a 532-city symmetric traveling salesman problem by branch and cut,” *Operation Research Letters*, vol. 6, pp.1-7.

[27]. Lin, S. and Kernighan B. (1973) “An effective heuristic algorithms for the traveling salesman problem,” *Operations Research*, Vol.21, pp.498-516.

[28]. Spellman, T. S., Sherlock, G., & et al., “Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisia* by microarray hybridization,” *Mol. Biol. of the Cell*, vol. 9, pp. 3273-3297, 1998.

Combining Competitive and Cooperative Coevolution for Training Cascade Neural Networks

Alexander F. Tulai
Computer Science Dept.
Carleton University
Ottawa, Ont, CANADA, K1S 5B6
alexander.tulai@rogers.com
Tel: (613) 730-2671

Franz Oppacher
Computer Science Dept.
Carleton University
Ottawa, Ont, CANADA, K1S 5B6
foppache@ccs.carleton.ca
Tel: (613) 520-2600/3520

Abstract

Cooperative Coevolution (CC) has been shown to be effective in problems where certain architectural details of the solution are evolved. This is the case of cascade neural networks where the number of hidden units is not pre-established but rather emerges through learning. We take a step towards having coadapted subcomponents emerge rather than being hand designed by showing that competing populations (evolved by GAs with different mutation and crossover probabilities) can be successfully used in selecting the species that are subsequently coevolved in a cooperative model. Our experimental results indicate that retraining is an essential step in the cooperative coevolution model. Previous studies used evolutionary algorithms (EAs) to train connection weights and neuron thresholds in artificial neural networks (ANNs). We show that by also evolving the characteristics of the neurons themselves, the quality of the solution (in terms of number of hidden units) could be significantly improved.

1 INTRODUCTION

EAs have been used in the past to train and/or initialize connection weights, evolve neural network architectures and learning rule adaptation, etc. (Yao, 1999).

This paper shows that in the case of cascade neural networks (CNNs) not only evolutionary strategy (ES) but also genetic algorithms (GAs) could be successfully used for evolving and training the nets. We also show that by evolving the neuron characteristics in addition to the connection weights a more compressed solution is obtained over the case of fix neuron activation function.

The paper is organized as follows. In section 2 we discuss the concept of coevolution. In section 3 we describe the problem under study and the definition of the species used by the EAs. In section 4 we describe the three algorithms that are used for comparison in this study. In section 5 we present and discuss the experimental results. In section 6 we discuss the impact of retraining in cooperative coevolution (CC) while in section 7 we discuss the effect of evolving the neuron characteristics followed by a discussion on algorithm robustness in section 8. Section 9 summarizes the conclusions of this paper.

2 COMPETITIVE AND COOPERATIVE COEVOLUTION

Coevolution is defined as a series of reciprocal evolutionary changes in interacting species acting as agents of selection for each other. Competitive and cooperative coevolution are two important forms of coevolutionary relationships. The nature of the relationship plays an important role in determining various components of the evolutionary model (like problem decomposition, credit assignment, etc.).

Combining competition and cooperation within a coevolution model has been used by cooperative coevolutionary GAs (Potter and De Jong, 1994). In the case of GAs, the competition is usually between individuals and not between populations.

The CC GA model inherits the limitations commonly associated with a GA, like pre-determining, through experimentation, of mutation and crossover rates or the population size. In addition to that, when new species are introduced in the CC model, certain selection criteria or a pool of candidates are needed to ensure the quality of the new species.

In our study, to alleviate these problems and increase the generality of the model, we introduce an extra step during which multiple populations, evolved by GAs with different mutation and crossover rates, participate in a competitive-cooperative coevolutionary process (competing among themselves but cooperating with the previous species) that results in one winning species. The individuals of the populations that underperform during

the competitive phase are re-distributed between the other populations. This process mimics real life situations when employees of a company that goes bankrupt join other successful companies. Consequently, during the competitive phase the competing populations may have slightly different sizes. The size of the population of the winning species will be equal to the sum of all initial populations but it is clear that the mutation and crossover rates are not known apriori as they depend on the GA used by the winning population. The winning species thus joins other previous winners in the cooperative coevolutionary phase of the algorithm. This process is repeated for every hidden unit introduced in the CNN.

The purpose of the competitive phase is twofold. On the one hand it selects a new species and on the other hand it selects the GA with the most appropriate mutation and crossover rates to evolve this new species. The purpose of the cooperative coevolutionary phase is to find those representatives from each species that together provide the best solution to the problem.

We compare the competitive-cooperative coevolutionary (CCC) GA model it with a CC model using evolutionary strategy (ES) and a pool of 8 initial candidate species to select from.

Both algorithms retrain all the species after a new species is introduced in the cooperative. This is different from other similar cooperative models (for example the cascade neural networks architecture can also be seen as a cooperative model) that do not perform retraining.

3 PROBLEM DESCRIPTION AND SPECIES DEFINITION

Proposed first by Alexis Wieland of Mitre Corp. the two-spiral has become a favorite hard problem to solve by training neural networks.

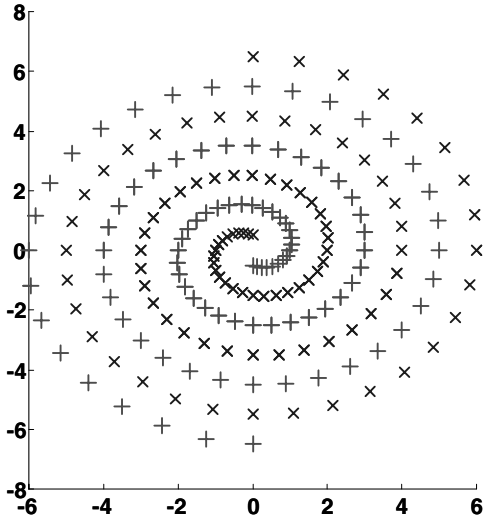


Figure 1 : The training patterns.

The problem consists of two intertwined spirals (see Figure 1) with 97 points on each spiral for a total of 194 points.

If a point that belongs to one spiral is input to the neural network the output of the network should be a positive signal and if the point belongs to the other spiral the output should be negative. When the problem is solved in this form, although it is rarely mentioned, we say that it is solved according to the $50-50$ criterion. Sometimes (Wah and Qian, 2000) classification problems are studied using the $40-20-40$ criterion.

Solutions to the two-spiral problem could either have an evolved architecture, like in the case of the CNNs first proposed by Fahlman (Fahlman and Labiere, 1990a), or they can have a fixed architecture. The best solution to the two-spiral problem, in terms of number of hidden units (HU), had 4 HUs and 19 connection weights and it was arrived at based on a fixed neural network architecture (Wah and Qian, 2000). The previously reported best solution based on an evolved architecture had 9 HUs and 75 weights (Fahlman and Labiere 1990b).

In previous studies (Fahlman and Labiere, 1990a; Potter and De Jong, 2000) the hidden units and the output unit were neurons with an output range of [-1,+1] and with a sigmoidal activation function given by the equation

$$a(x)=\begin{cases}-1, & x<-15\\ \dfrac{2}{1+e^{-x}}-1, & -15\le x\le 15\\ 1, & x>15\end{cases} \quad (1)$$

We are also using neurons with a [-1,+1] range but we treat the activation function itself as an evolvable function with the equation.

$$a_{\alpha,L}(x)=\begin{cases}-1, & x<-L\\ \tanh(\alpha x), & -L\le x\le L\\ 1, & x>L\end{cases} \quad (2)$$

Besides the number of hidden units, which is an architectural element, and the connection weights (that also include the bias) we are also evolving the characteristics of each neuron by including the parameters α and L (the gain and the input signal limit) in the individual genome of a species. Defining the species for the cooperative coevolution architecture solution to the two-spiral problem could be successfully done in accordance with the original cascade network training algorithm (Fahlman and Labiere, 1990b). That method consists in, first, evolving together all the connection weights and neuron thresholds (which in fact are also weights on connections to a constant +1.0 input) leading into a new hidden unit and than, second, train all the connection weights leading into the output unit (the two-spiral problem requires only one output). Following this idea (Potter and De Jong, 2000) assign a separate species to the weights leading into a unit whether an output or a hidden unit. This choice of the species has the disadvantage that whenever a new hidden unit is

introduced, a randomly initialized weight needs to be added to all the individuals of the species assigned to the output unit.

In our solution we decided to group all the connection weights needed for a new hidden unit (input, output and bias connections) as well as the neuron characteristics as one species as shown in Figure 2.

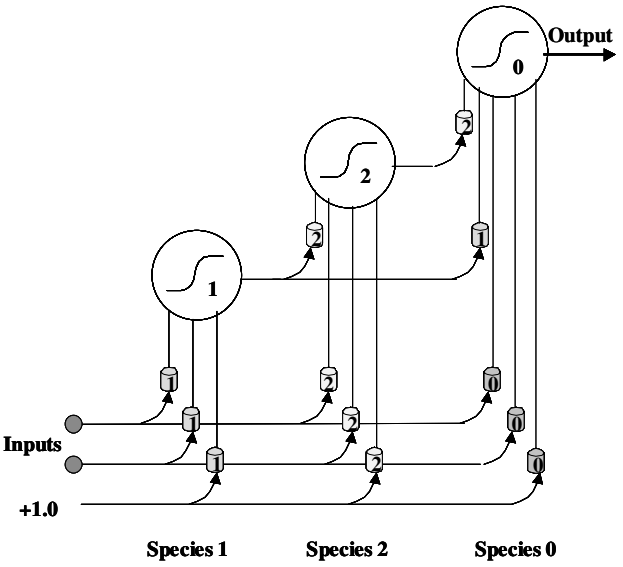


Figure 2 : Cascade net with 1 output and 2 hidden units. All elements of a species have the same number.

Each individual genome is represented by a set of $5+i$, genes where $i = 0,1,\cdots,\max HU$ is the index of the unit introduced, with species 0 representing the output unit and species 1 to maxHU representing the HUs. The first species created represents the output unit and has 5 genes, two for the neuron characteristics and three for the weights on its input connections (one of the connection weights will in fact represent the output unit bias as previously stated). Every time a new hidden unit is introduced, we create a new species with a genome that has one more gene than the previous species. A new hidden unit is always introduced between the last hidden unit and the output unit.

4 ALGORITHM DESCRIPTION

For this study we compare three different learning methods for evolving a cascade correlation neural network that solves the two-spiral problem.

- the GA-based method, that we propose, uses both competitive and cooperative coevolution GA and we will refer to as the CCC-GA method.
- a cooperative coevolution method using the (μ,λ) evolutionary strategy (ES) as described in (Schwefel, 1995) and used by (Potter and De Jong, 2000) for solving the same two-spiral problem. We will refer to this algorithm as the CC-ES method.
- a second order gradient-descent based method as described in (Fahlman and Labiere, 1990a). We will refer to this method as the cascade correlation method.

The cascade networks generated by Fahlman's method are comprised of symmetrical sigmoid units (both hidden and output units) as defined by equation (1). In order to properly compare the CCC-GA and the CC-ES algorithms both will generate cascade networks using neurons with an activation function as defined by equation (2) and will include the parameters α and L among the evolved genes.

All methods try to evolve cascade networks that classify all input patterns, and reach this objective by minimizing the squared error sum (SES) at the output of the cascade neural network over all the patterns in the two-spiral data set,

$$Error = \sum_{p} (d_p - y_p)^2$$

where p is the training pattern index, d_p is the desired output and y_p is the actual output.

4.1 CCC-GA METHOD

The CCC-GA method has two major phases, a competitive coevolution phase and a cooperative coevolution phase.

The competitive coevolution phase starts with N populations of m individuals, each population being evolved by a GA with overlapping population, and ends with 1 population with $N \cdot m$ individuals at the end of an iterative process that sees the least fit populations being absorbed by the fitter ones. Each GA will use a different mutation and crossover probability described at step 1 of the competitive phase of the algorithm.

Competitive coevolution phase of the algorithm

1. *Initialization*. Create N populations with m individuals in each population. Train each population with a similar GA but use different mutation and crossover probability. In our study N was set to 16, m to 64 and the mutation and crossover probabilities are distinct pairs $(pmut, pxover) \in \{0.125, 0.375, 0.625, 0.875\}^2$.
2. *1-step evolution*. The N populations are each trained on the whole set of input data once.
3. *Ranking*. All $N \cdot m$ individuals are ranked (*0* to $N \cdot m - 1$) based on their fitness $F_i, i = 0, \ldots, N \cdot m - 1$ calculated during training. The population ranking will be done based on the performance of the top M individuals in the ranking individual order with $M \geq m$. The only time when M will be different from m is when $F_m = F_{m-1}$ in which case we must consider all the individuals that have an identical fitness $F_{m-1} = F_m = \cdots = F_{M-1}$.

4. *Credit assignment*. Each of the M individual is assigned a credit Γ_i based on the equation $\Gamma_i = e^{(M-i)/M}, i = 0,\cdots,M-1$. If k individuals have identical fitnesses $F_j, F_{j+1},\cdots,F_{j+k-1}$ with $0 \le j, j+k-1 \le M-1$, they will each be assigned the same credit value

$$\Gamma_l = (\sum_{l=j}^{j+k-1} e^{(M-l)/M})/k$$

Each of the N populations receives a credit equivalent to the sum of the credits received by its own individuals. Please note that if a certain population has no representatives in the top M individual ranking, it receives a credit equal to 0.

5. *Check population elimination criterion*. When a population ranks last D times in succession it is eliminated at step 6, otherwise we continue with step 2. In our study, D is initially set to 10. To avoid possible processing traps, D is decreased by 1 every 10 iterations.

6. *Population elimination*. If the elimination criterion is met, the genomes of the population that is eliminated are one by one distributed between the other $N-1$ populations with a random starting point. After N is decremented by 1 the algorithm continues with step 2 if $N > 1$. The algorithm iterates until only one population with $N \cdot m$ individuals is left. This population, as a separate species, joins the other previous winning species in the cooperative coevolution phase of the algorithm.

Cooperative coevolution phase of the algorithm

7. *Winner integration*. The winning species is cooperatively coevolved with the best representatives from the previous winning species until the SES does not decrease by more than δ from one iteration to another in I consecutive iterations. At any point in time, the number of species cooperating equals the numbers of neurons in the CNN. At the end of the iterative process the genome with the highest fitness is used to grow the size of the cascade correlation neural network with one hidden unit. In our study $\delta = 0.1$ and $I = 10$.

8. *Retraining*. After the new hidden unit is added to the network, an attempt is made to retrain once all the species created up to this point (i.e. all the units of the cascade neural network). During retraining, each species contributes to the network with its best genome except the species that it is retrained. The species are retrained in a random order (for other possibilities see the section on retraining) except the newest species that is always retrained last. The retraining is done using the same stopping criterion as described at point 7. If a species cannot find a better genome through retraining, the previous one is kept.

During step 7 or 8 of the algorithm, if all the points in the training set are correctly classified, the algorithm stops. CCC-GA method uses a GA with overlapping populations (final size 1024 individuals) with a 50% probability of replacement and Tournament selection.

4.2 CC-ES METHOD

To facilitate the comparison of the results of our study with previous results from previous studies, we used the same (μ,λ) evolutionary strategy (ES) as described in (Schwefel, 1995) and used by (Potter and De Jong, 2000) for studying the same two-spiral problem, with the same choice of $\mu = 10$ and $\lambda = 100$.

A very interesting aspect of this evolutionary algorithm is that both the genes under study and the standard deviations used by the mutation operator are part of the genome and consequently are evolved together.

If $x_k^{(0)} \in (-10.0,10.0)^n, k = 1(1)\mu$ are the initial vectors for the connection weights, with n the genome length, we only have to initialize $x_1^{(0)}$ while the algorithm itself will initialize the other $\mu-1$ individuals by addition of $(0,(\sigma_i^{(0)})^2)$ normally distributed vectors. Just as in (Potter and De Jong, 2000) we initialize all the n components of the first standard deviation vector to $20/\sqrt{n}$ while the other $\mu-1$ standard deviation initial vectors are initialized by Schwefel's algorithm.

The n standard deviation components (mutation steps) are updated by multiplying them every generation t with

$$e^{Gauss(0,\sigma_i^{(t)})}$$

the initial values being given by $\sigma_i^{(0)} = 1/\sqrt{n}$. In our study we used the (μ,λ) implementation of the algorithm given by the 'korr2' program. Although the program allows recombination for both components of the parents (connection weights and standard deviations) we turned this option off and used only mutation. The recombination was turned off for a better comparison with (Potter and De Jong, 2000) and because we had also experienced a degradation of the results when it was turned on.

For every new species that is created, with the exception of the first species, the algorithm is executed eight times to generate a pool of eight species, the best genome in each species representing a candidate HUs. The species that produces the genome with the best results in reducing the network residual output error is inserted in the cooperative of species. The number eight has been chosen to match the size of the pool used by the cascade correlation algorithm. The existence of a pool of candidates and the need to pre-determine its size is a disadvantage for the CC-ES method when compared to the CCC-GA method that always runs the competitive coevolution phase only once for all the newly created species. Other cooperative coevolution models that do not use a pool of candidates but rather introduce some criteria of acceptance for the new species have a similar disavantage when compared to the CCC-GA method.

4.3 GRADIENT-DESCENT BASED METHOD

Classical methods for training neural networks have used gradient-descent techniques such as the back-propagation algorithm for a long time.

To speed up the learning process researchers have used a number of schemes like second order methods (some approximate form), conjugate gradient methods and so on. Fahlman's Quickprop algorithm for updating the connection weights also uses a second-order method.

The cascade correlation learning algorithm creates and installs new hidden units in two steps. First it tries to maximize the magnitude of the correlation between the new unit's output V and the residual error signal the algorithm tries to eliminate, which for a certain input data pattern p and a for a certain output unit o is $E_{p,o}$, summed up over all input data patterns and all output units

$$S = \sum_{o} \left| \sum_{p} (V_p - \overline{V})(E_{p,o} - \overline{E}_0) \right|$$

where $\overline{V}$ and $\overline{E}_o$ are the values of V_p and $E_{p,o}$ averaged over all data patterns.

The algorithm uses the Quickprop algorithm to update the new hidden unit's input weights while trying to maximize the correlation value S.

In the second phase of the algorithm, the new hidden unit is inserted in the network, its input connection weights are frozen and the input weights for the output unit are re-trained using the same Quickprop algorithm.

When this algorithm is used to solve the two-spiral problem, it starts with a pool of eight candidates whenever it needs to create a new hidden unit.

5 EXPERIMENTAL RESULTS

Fifty runs were performed using the CCC-GA and the CC-ES method because their running time is high. By comparison, the average computer time per run for the cascade correlation method is orders of magnitude smaller and, consequently, we were able to perform 10,000 runs on a comparable timescale.

Table 1: Required number of hidden units for the three methods under study

Method	Hidden units				Time[1]
	Min	Max	Mean	Fail	[s]
CCC-GA	7	16	10.54 ± 0.71	0	1620
CC-ES	8	15	9.9 ± 0.57	0	2700
Cascade correlation	10	19	13.4 ± 0.04	7	5

[1] Average running time calculated on a 500 MHz Pentium III PC

Table 1 shows minimum and maximum number of hidden units produced by each method as well as the average number of hidden units produced by each method, along with 99-percent confidence on the mean. If a solution to the two-spiral problem is not found after introducing 25 HUs the run is classified as a failure (non-convergence).

Our results for the cascade correlation based method have been obtained using the 'cascor' program. The results are significantly better than those reported in (Fahlman and Labiere, 1990a) or (Potter and De Jong, 2000) but are similar to those reported by (Hansen and Pedersen, 1994).

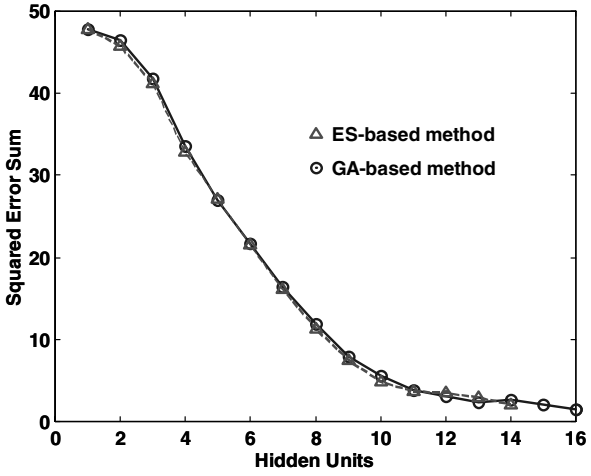


Figure 3: The average decay of the SES as a function of the number of hidden units.

As can be seen in Figure 3, the CCC-GA method and the CC-ES method have very similar behaviour when it comes to decreasing the SES, the two curves being almost identical when the results of 50 runs were averaged. Both CCC-GA and CC-ES methods produce better results than the cascade correlation method, both in terms of average number of hidden units as well as the minimum number of hidden units required for a cascade network solution to the two-spiral problem. However the cascade correlation learning method is a much faster method compared to both coevolutionary methods.

The CC-ES algorithm has produced the best results in terms of average number of hidden units but the CCC-GA produced a cascade network with only 7 hidden units. In fact the neural network with 7 hidden units is the smallest evolved cascade network solution (the number of hidden units is not known apriori) reported in the literature, we are aware of, to the two-spiral problem.

Another important distinction between our results for the CC-ES method and previously reported results (Potter and De Jong, 2000) is that we had no failures in all the runs we performed. In runs where the neuron activation function was given by equation (1) and the training patterns target for classification was set to {-1,+1} we also experienced occasional failures but it was noticed that when the algorithm stopped to converge the output

error was always an integer number. It was determined that when a pattern was misclassified such that the output of the net was minus the desired value (for example –1 instead of +1) the squared error was an integer number. In such cases because of the definition of the sigmoid activation function, any changes to the connection weights that did not result in pushing the output of the net back into the dynamic range of the output unit were not picked up by the algorithm and treated as "better" or "worse" solutions as they should have been. Consequently, the evolution was stuck in a local minimum and the algorithm reduced to a random search. To formally describe the problem, if the neural network is specified by a vector parameter $\mathbf{w} = (w_1, \ldots, w_n)^T \in W^n$, (where the $\mathbf{w}$ is made up of the best genomes from each of the coevolved species and W is the allowed gene range, in our study $[-10,+10]$), the task is to find the optimum vector $\mathbf{w}^*$ corresponding to a net that classifies all patterns in the input set P. When the $50-50$ criterion is used for classification the problem is considered solved when $d_p \cdot y_p(w) > 0, \forall p \in P$.

If $e_p(w) = d_p - y_p(w)$ is the network output error for input pattern $p \in P$ the algorithm achieves the classification goal by minimizing

$$E(w) = \sum_p e_p^2(w)$$

where $y_p(w)$ is the output of the neural network defined by the parameter w and the input pattern p. Let's assume the neural network is comprised of hidden and output units with the activation function given by the equation

$$a\ (x) = \begin{cases} -R, & x < -L \\ b(x), & -L \le x \le L \\ R, & x > L \end{cases}$$

with $\lim_{x \to -\infty} b(x) = -R$ and $\lim_{x \to \infty} b(x) = R$, and for any input pattern there are only two possible outputs used in training $d_p \in \{-R, +R\}$. Let's define an order relation on W^n, $w_1^f < w_2^f \Leftrightarrow E^f(w_1) < E^f(w_2), f \in \{a, b\}$ where the superscript f indicates what activation function is used by the neurons in the network. If $C(m)$ is the set of classified data patterns and $U(m) \neq \varnothing$ is the set of unclassified data patterns after m HUs have been introduced, we have $P = C(m) \cup U(m)$ where P is the set of all input data patterns.

If during training a state is reached where for any pattern $p \in U(m), e_p(w) = 2R$ for any w produced by the algorithm, the EA search space is reduced by all the vectors w_1, w_2 such that $w_1^b < w_2^b$ but $w_1^a = w_2^a$. When this happens *the search becomes purely random* unless the algorithm can find another point w_r such that $e_p(w_r) < 2R, p \in U(m)$ and $E^a(w_r) < E^a(w), \forall w \neq w_r$. Please note this phenomenon could happen because of the limitation on its input signal used by the output neuron activation function $a\ (x)$, but could also happen because of the computational approximation errors in calculating the network output.

Because the two-spiral problem is a classification problem (we shouldn't forget that) the simplest and most effective solution to this problem is to use different values for the desired training values d_p than those used by the activation function as output range limits. For example, in our study $R = 1$ so after we changed the training expected values to $d_p \in \{-1/2, +1/2\}$ the problem has never occurred again and the algorithm has always converged.

6 RETRAINING

A major difference between Fahlman's cascade correlation method and the two cooperative coevolution based methods is the use of retraining. The cascade correlation method, once it has introduced a new hidden unit does not change its input connection weights while both the CCC-GA and CC-ES algorithms use retraining as an important method for further reducing the SES at the network output. For example, when no retraining was used by the CCC-GA method, the performance decreased so severely that only 2 runs were successful in 10 runs performed.

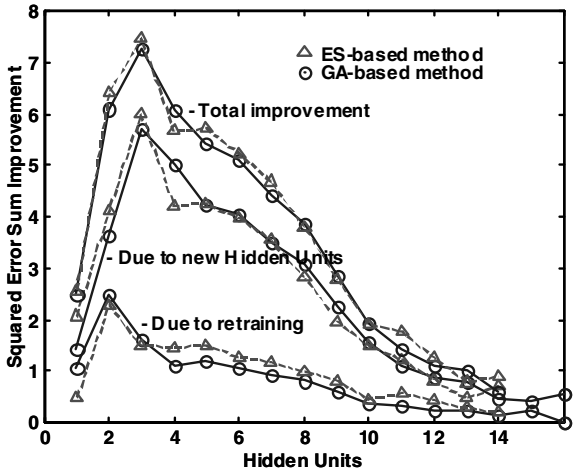


Figure 4: The improvements in squared error sum (SES) after new hidden units, retraining and the sum of the two.

While Figure 4 shows the improvements in SES are very similar for the CCC-GA and CC-ES methods, we could also see that starting with hidden unit 4, on the average, the CC-ES is consistently better in retraining the cascade neural network. This result may be due to the fact that in the case of the CCC-GA method, the GA used to evolve the species that get selected during the competitive coevolution phase, remains fixed during retraining, in the sense that both the mutation and the crossover rate are fixed. The CC-ES uses only mutation, but the mutation vector evolves during retraining just like the individual genes vector does. The search mechanism would benefit from a fully adaptive mutation and crossover probability (White and Oppacher 1994) and the CCC-GA could be modified relatively easily to cope with this limitation. For example, we could do it by randomly dividing the individuals of the species to be retrained in N populations with m individuals in each population,

assigning different mutation and crossover probability to the GA used to train the populations and letting them go again through a competitive phase. However, this would considerably slow down the algorithm in its current form.

Another aspect of retraining we looked at was the order of retraining the species. The results presented in the previous sections were based on retraining all species (including species 0 corresponding to the output unit) in a random order. We however questioned whether retraining done following the order in which the species have been introduced (0 to maxHU-1) or the reverse of that (maxHU-1 to 0) has any impact on the results. The results of the 50 different runs performed with the CCC-GA algorithm are presented in Table 2. In all three cases the last hidden unit introduced (maxHU) is retrained last.

Table 2: Required number of hidden units for CCC-GA when different retraining methods are used

Method	Hidden units			
	Min	Max	Mean	Fail
random	7	16	10.54 ± 0.71	0
$0 \rightarrow \max HU - 1$	8	16	10.42 ± 0.58	0
$\max HU - 1 \rightarrow 0$	8	14	10.46 ± 0.4	0

While the average number of hidden units seems to be similar for the three methods it does appear that the random retraining yields a slightly higher variance of the results. This factor may favour the discovery of the cascade network with 7 hidden units.

7 NEURON CHARACTERISTICS

In our study, the two EA methods have evolved not only the connection weights of the neural network but also the characteristics of the activation functions for all the neurons of the network. To determine the effect of including the characteristics of the neurons as evolvable genes, we have also performed 50 runs where the individual genomes included only the connection weights. In this case all the neurons in the evolved cascade network, the hidden and the output neurons, have a sigmoidal activation function given by equation (1) that does not change during training.

For this aspect of the study we have used only the CCC-GA and CC-ES methods and the results are presented in Table 3.

The results show clearly that including the characteristics of the neurons among the evolved genes not only has increased the generality of the algorithms but it has also significantly improved the results. In the case of the ES-based method the average size of the cascade network has decreased by two hidden units while in the case of the GA-based method the average and the minimum number of hidden units have decreased even more significantly.

Table 3**:** Required number of hidden units with fix or evolvable neuron characteristics.

Method	Hidden units				Neur. char.
	Min	Max	Mean	Fail	
CCC-GA	7	16	10.54 ± 0.71	0	Ev.
CCC-GA	10	17	13.18 ± 0.56	0	Fix
CC-ES	8	15	9.9 ± 0.57	0	Ev.
CC-ES	9	14	11.9 ± 0.45	0	Fix

These results suggest that future studies of neural network evolution should look for a comprehensive solution and also consider evolving the characteristics of the neural activation function.

8 METHOD ROBUSTNESS

The number of adjustable parameters, the sensitivity of the results to user choices, the efficiency of the model in solving different problems, are possible criteria of evaluating the robustness of a computational method.

When comparing the CCC-GA and the CC-ES methods the first thing to differentiate them is the method for choosing the next species to join the cooperative of coevolving species.

While the CC-ES chooses from a pool of 8 different species (why not 4 or 10?), the CCC-GA method doesn't have to make such a choice. In fact, if the CC-ES model doesn't use a pool of species and uses a randomly initialized population, the results (for example the average number of hidden units per evolved cascade network) degrade very significantly. From this point of view it is clear that the CCC-GA method is more robust than the CC-ES method.

When it comes to the evolutionary operators used, ES does not use crossover and the mutation probabilities are independently evolved by the algorithm (so the user has only to initialize an initial vector). While this seems to introduce a clear advantage for the CC-ES method, the CCC-GA compensates for it by evolving the species with the crossover and mutation probabilities best suited for a certain phase of the algorithm. In addition to that, through experimentation we found out that if the ES initial choice is to evolve the mutation probabilities as a group rather than individually (each mutation step evolved independently) the results of the CC-ES will again degrade from the results in Table 1.

Another important parameter influencing the generality and the robustness of the algorithm is the population size. In order to see how the population size influences the results we have performed 50 runs with the CCC-GA with a change in the population size of the species selected for the cooperative phase of the algorithm. Before the species winning the competitive phase is to join the other species

for the cooperative phase of the algorithm we halve the population such that instead of 1024 individuals, each species selected for cooperative coevolution will have only 512 individuals. The results of the two sets of runs are given in Table 4.

Table 4: Required number of hidden units for CCC-GA method with different final populations

Method	Hidden units				Pop. size
	Min	Max	Mean	Fail	
CCC-GA	7	16	10.54 ± 0.71	0	1024
CCC-GA	8	15	10.56 ± 0.62	0	512

The results obtained with species that have only half the population ($N \cdot m/2$) are almost identical to those obtained using species with an entire population ($N \cdot m$) proving that the CCC-GA method is also robust under population variations.

The choice of the GA for the CCC-GA method is not crucial either. A simple GA provides similar results, but it is twice as slow.

Finally, both algorithms need some accuracy of approximation parameters to be set.

While the CCC-GA method needs to be tested on new problems, so far this method has shown robustness and provides a good model for further coevolution studies.

9 CONCLUSIONS

In this study we have introduced the concept of competitive-cooperative coevolution GAs and have shown that, when this method is used for solving the two-spiral problem, the results are very similar to those obtained by using only cooperative coevolution based on an ES-method. The results of both methods, as far as we are aware, are better than any previous cascade networks solutions to the two-spiral problem. It should be noted that previous attempts to use GAs to study the two-spiral problem didn't yield good results. Moreover, in the case of CCC-GA, neither the mutation nor the crossover probabilities are pre-determined (through prior experiments) but are rather selected (from a finite set of possibilities) as a result of the competitive-cooperative coevolution phase of the algorithm.

The cascade neural networks evolved through our methods are complete in the sense that all the characteristics of the nets, number of hidden units, connection weights as well as neuron characteristics are evolved. While evolving the neuron characteristics is natural and easy to do with EA methods it may be more difficult to do in traditional approaches (like the gradient-descent methods). This constitutes a definite advantage of the EA methods for evolving cascade neural networks over other methods.

The CCC-GA algorithm uses the competitive phase to remove the need for a pool of candidates or empirical "goodness" criteria in introducing new species in the cooperative coevolution process and it shows robustness under population and method of retraining variation.

References

S.E.Fahlman and C.Labiere (1990a). The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

S.E.Fahlman and C.Labiere (1990b). The cascade-correlation learning architecture. In D. S. Touretzky (ed.), *Advances in neural Information Processing Systems 2*, 524-532. San Mateo, CA: Morgan Kaufmann.

L.K.Hansen and M.W.Pedersen (1994). Controlled growth of cascade correlation nets. *Proceedings of the International Conference on Artificial Neural Networks*, volume 1, 797—800. Sorrento, Italy.

M.A.Potter and K.A.DeJong (1994). A Cooperative Coevolutionary Approach to Function Optimization. In Y.Davidor and H.P.Schwefel (eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature,* 249-257. Berlin, Germany: Springer-Verlag.

M.A.Potter (1992). A genetic cascade-correlation learning algorithm. *Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks*,122-133. IEEE Computer Society Press.

M.A.Potter and K.A.DeJong (2000). Cooperative coevolution: an architecture for evolving co-adapted subcomponents. *Evolutionary Computations* 8(1):1-20.

H.P.Schwefel (1995). *Evolution and Optimum Seeking*. New York, NY: John Wiley and Sons.

B.W.Wah and M.Qian (2000). Constrained Formulations for Neural Networks Training and Their Applications to Solve the Two-Spiral Problem. *Proceedings of the Fifth International Conference on Computer Science and Informatics*.

T.White and F.Oppacher (1994). Adaptive Crossover Using Automata. In Y.Davidor and H.P.Schwefel (eds.), *Proceedings of the Third Conference on Parallel problem Solving from Nature,* 229-238. Berlin,Germany: Springer-Verlag.

X.Yao (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE* 87(9):1423-1447.

From TwoMax to the Ising Model: Easy and Hard Symmetrical Problems

Clarissa Van Hoyweghen
Intelligent Systems Lab
University of Antwerp
Groenenborgerlaan 171
2020 Antwerp, Belgium
hoyweghe@ruca.ua.ac.be

David E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois
117 Transportation Building
104 S. Mathews Av. Urbana, IL 61801
deg@illigal.ge.uiuc.edu

Bart Naudts
Intelligent Systems Lab
University of Antwerp
Groenenborgerlaan 171
2020 Antwerp, Belgium
bnaudts@ruca.ua.ac.be

Abstract

The paper shows that there is a key dividing line between two types of symmetrical problems: problems for which a genetic algorithm (GA) benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum. By analyzing in detail the search process of a selectorecombinative GA optimizing a TwoMax and comparing this search process with that of a one-dimensional Ising model, the paper investigates the difference between these two types of symmetrical problems. For the first type of problems, naively adding a niching technique to the genetic algorithm makes the problem harder to solve. For the last type of problems, niching is necessary to find an optimum.

1 INTRODUCTION

In the context of optimization by genetic algorithms (GAs), scaling, deception, epistasis, and noise are well known examples of problem difficulty characteristics. The presence of one such characteristic in the representation of a search problem indicates a certain type of difficulty the GA is to encounter during its search for global optima. In this paper we investigate another aspect of problem difficulty: the existence of equally good partial solutions in symmetrical or hierarchical problems. The loss of some equally good partial solutions due to genetic drift can prevent a GA to find an optimum, but for some problems it can also be beneficial.

The purpose of this paper is twofold. Firstly, the paper analyzes the search process of a GA solving a multimodal equivalent of the OneMax problem, the so called TwoMax, and compares its search with the search process of a GA solving a OneMax. Secondly, the paper shows that there is a key dividing line between two different types of symmetrical problems: problems for which the GA benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum and for which niching becomes a necessity. Naively adding a niching technique to an algorithm optimizing a problem of the first type makes the problem harder to solve.

The paper is structured as follows. Section 2 introduces equally good partial solutions or non-inferior BBs and shows how genetic drift chooses between them. Section 3 analyzes the search process of an easy symmetrical problem, a TwoMax, and shows that there exists a class of problems that benefits from the fact that genetic drift chooses between equally good partial solutions. Section 4 and 5 compare two types of symmetrical problems: problems for which searching in terms of non-inferior BBs is necessary and problems that benefit from the fact that genetic drift chooses between non-inferior BBs. Section 6 summarizes and concludes the paper.

2 NON-INFERIOR BBS AND NICHING

The working of a genetic algorithm can be explained by the search for superior building blocks. Building blocks (BBs) with above average fitness are combined to construct higher order building blocks. However, recent studies [16, 12, 7, 14] show that the search for superior BBs is not always sufficient to find a solution. When an optimization problem contains BBs which are equally good and superior to all alternatives, so called *non-inferior BBs* [4], making a choice between such BBs can avoid a GA to reach an optimum. A search strategy preserving all non-inferior BBs can then become a necessity to solve the problem quickly, reliably, and accurately. In [14], for example, it is shown that the simple GA cannot solve the standard Ising model in a reasonable amount of time because the population loses some non-inferior BBs due to genetic

drift. When a niching technique is used, the non-inferior BBs are preserved and an optimum is reached quickly. The same phenomenon can be seen when solving the H-IFF problem [16]. H-IFF combines non-inferior BBs with a hierarchical problem structure. Without a niching technique the simple GA is unable to solve the problem up to the highest level. Another class of functions which question the algorithm's search for superior BBs is the Tobacco Road functions [3, 4]. The Tobacco road functions combine hierarchy, multimodality, and deception. At the lower level the problem can be decomposed into several non-overlapping folded trap functions, each having two complementary optima: 'all 0s' and 'all 1s'. At the higher level the problem consist of one or more trap functions with solution 'all 1s' and deceptive trap 'all 0s'. Without keeping all alternative low-level partial solutions, an algorithm is not able to solve the problem.

Next section describes in more detail the effect of genetic drift on non-inferior BBs and shows how niching can prevent that genetic drift chooses between them.

2.1 NON-INFERIOR BBS AND GENETIC DRIFT

A BB is called superior if it has a higher average fitness value than its competitors. For example, if

$$\begin{aligned} f(*0*) &= 1, \\ f(*1*) &= 2, \end{aligned} \tag{1}$$

we say that BB $*1*$ is superior to BB $*0*$. Non-inferior BBs are BBs which are equally good or superior to all alternatives. For example, if

$$\begin{aligned} f(*01*) &= 1, \\ f(*10*) &= 1, \\ f(*00*) &= 2, \\ f(*11*) &= 2, \end{aligned} \tag{2}$$

BBs $*00*$ and $*11*$ are non-inferior BBs. They have equally good fitness averages and their fitness average is superior to the fitness averages of BBs $*01*$ and $*10*$. When optimizing a problem containing equally good partial solutions (or non-inferior BBs), we distinguish two types of genetic search: searching for superior BBs and searching for non-inferior BBs. In the first case, the algorithm blindly chooses between equally good partial solutions. In the second case, equally good partial solutions stay in the population with equal proportions.

Consider following examples: A population contains BBs, 00, 11, 01, and 10, each occupying one fourth of the population. [1] The fitness value of 00 and 11 equals 2, whereas the fitness value of 01 and 10 equals 1. Each generation, a new population is created as follows. Randomly pick two BBs with replacement and add the BB with the highest fitness to the new population. When two BBs have equal fitness, one of them is randomly picked and added to the new population. This selection strategy is known as binary tournament selection with replacement. Figure 1 (a) shows the result of this selection process. Clearly, the process can be divided into two phases (separated in the figure by a vertical line). During the first phase, there are two types of selection: selection for good reason between BBs with different fitness, and selection due to genetic drift between BBs with equal fitness. During this phase, BBs 00 and 11 takeover the population. Notice that at the end of the first phase BBs 00 and 11 already occupy a different proportion of the population. During the second phase, there is only selection due to genetic drift. Both BBs have equally good fitness; hence, there is no reason to prefer one above the other. The stochastical errors made by randomly picking one of them during each equal tournament cause that one of the two non-inferior BBs disappears out of the population. This example reminds us of the fact that the standard selection process works in term of superior BBs: BBs with high fitness average are preferred above BBs with low fitness average, but the selection process cannot guarantee that non-inferior BBs stay in the population with equal proportion. A well-know technique to preserve multiple good

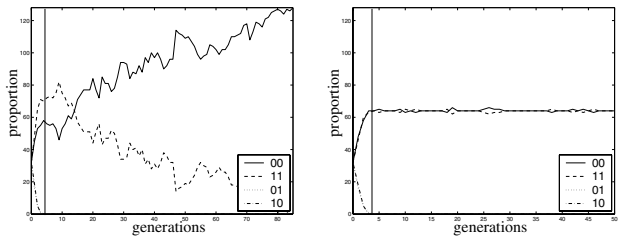


Figure 1: The plot shows the number of representatives of the BBs 00, 11, 01 and 10 during the selection process. Initially, each BB has 32 representatives in the population. In (a) no niching technique is used. In (b) each BB has to share its fitness with all BBs in the population which are not further away than Hamming distance 1.

solutions is niching. Niching tries to allocate subpopulations among non-inferior BBs such that the selection process works more in terms of non-inferior BBs instead of superior BBs. The niching technique used in this paper is fitness sharing [6], as detailed in section 5. Figure 1 (b) shows the result of the selection process using binary tournament selection with continuously updated sharing [10]. Each BB shares its fitness with all BBs in the population that are not further away than Hamming distance 1. The selection process has again two phases. During the first phase, BBs 00 and 11 takeover the population. During the second phase, the non-inferior BBs 00 and 11 are kept in

[1] In the example, a BB can be seen as an individual and the term fitness average can be replaced by fitness value.

the population in equal proportion. This second example shows that when optimizing a problem using tournament selection with continuously updated sharing equally good partially solutions are selected for reproduction in the same proportion and the GA will evolve in terms of non-inferior BBs, keeping equally good partial solutions in the population.

The Ising model, H-IFF, and the Tobacco Road functions ask a search strategy that works more in terms of non-inferior BBs instead of superior BBs. Adding a niching technique to evolutionary algorithms is a possible way to get this behavior. In the context of designing a robust problem solver which solves problems of bounded difficulty quickly, reliably, and accurately, one can therefore ask the question if niching should be a standard component of a competent GA. However, in next section we will see that there exists a class of problems that benefit from the fact that genetic drift chooses between equally good partial solutions and become harder when a niching technique is naively used.

3 ANALYSIS OF A TWOMAX

The OneMax or bit-counting problem is well known and well studied in the context of GAs. In this paper, we analyze the search process of a simple GA solving a multimodal equivalent of the OneMax, the TwoMax (also called the Twin-Peaks problem [2]), and compare its search with the search process of a GA optimizing a OneMax. The OneMax problem is defined by

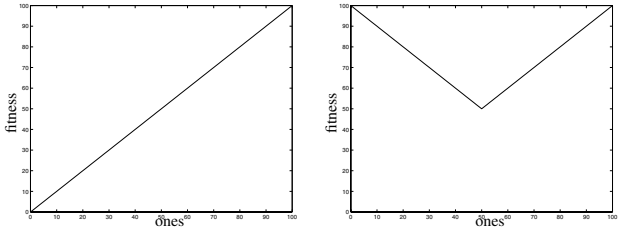
$$f_{OneMax}(s) = u, \tag{3}$$

with u the number of ones in the string s. The optimum is reached in the all 1s string. The TwoMax problem is defined by

$$f_{TwoMax}(s) = \left| \frac{\ell}{2} - u \right| + \frac{\ell}{2}, \tag{4}$$

with u the number of ones in the string s and ℓ the string length. A TwoMax returns the number of ones if there are more ones than zeros. If there are more zeros than ones, it returns the number of zeros, $\ell - u$. Figures 2 (a) and (b) show both fitness functions with string length 100.

A TwoMax has two optima, the string containing only ones and the string containing only zeros. Moreover, a TwoMax contains *spin-flip* or *bit-flip symmetry* (see [9] in the context of GAs), a symmetry which is characterized by fitness invariant permutations on the alphabet and found in the Ising model and H-IFF. The average fitness of a schema is equal to the average fitness of the schema's complement, e.g. $f(*1*01*) = f(*0*10*)$. The problem therefore contains many equally good partial solutions or non-inferior BBs.



(a) (b)

Figure 2: (a) OneMax and (b) TwoMax with string length 100

Our primary interest in this paper is to study the diversity of the population and the preservation or extinction of non-inferior BBs. The GAs used in the rest of the paper are so-called *selectorecombinative* GAs – they use only selection and recombination. To emphasize that the GA does not use mutation, we use the abbreviation srGA when confusion is possible. The GA does not use a mutation operator to add diversity to the population since mutation has only a fairly local scope and it cannot guarantee that equally good partial solutions are kept in the population. Unless stated otherwise, all experiments use binary tournament selection together with uniform crossover. The crossover rate is set to 1, and the population size equals 200.

3.1 CONVERGENCE MODEL

When solving a problem, it is useful to know how long it takes before a population converges to a population containing only optimal strings. In this section we construct a convergence model for a TwoMax using the convergence models for OneMax [8, 13]. By doing so, we gain important insight in the search process of an srGA optimizing a TwoMax.

We start by giving a quick reminder of how the convergence model for OneMax is derived. The *response to selection* equation,

$$\Delta f = f_{t+1} - f_t = b_t I \sigma_t, \tag{5}$$

introduced into the evolutionary computation community by Mühlenbein and Schlierkamp-Voosen [8], calculates the difference in mean fitness of two successive populations using the *selection intensity* I, the population's fitness variance σ_t^2, and the *heritability* b_t. For a OneMax the heritability b_t is known to be constant and equal to 1. This means that the average fitness of the offspring is equal to the average fitness of the selected parents. The OneMax problem has an interesting characteristic: one single proportion value p_{1_t}, giving the percentage of ones in the population at generation t, reveals a lot of information about the population at generation t. If we assume that the proportion of ones is uniformly divided among the loci

in the population, the mean fitness at generation t equals ℓp_{1_t} and the fitness variance σ_t^2 can be approximated by $\ell p_{1_t}(1 - p_{1_t})$. The population is optimal when $p_{1_t} = 1$. Equation 5 now yields

$$p_{1_{t+1}} - p_{1_t} = \frac{I\sqrt{p_{1_t}(1 - p_{1_t})}}{\sqrt{\ell}}. \quad (6)$$

Approximating the above difference equation with a differential equation and integrating this equation gives

$$p_{1_t} = \frac{1}{2}\left(1 + \sin(\frac{It}{\sqrt{\ell}} - \theta_0)\right), \quad (7)$$

with $\theta_0 = \arcsin(1 - 2p_{1_{t=0}})$. The convergence time $(p_{1_t} = 1)$ can then be calculated by

$$t_{conv} = \left(\frac{\pi}{2} + \theta_0\right)\frac{\sqrt{\ell}}{I}. \quad (8)$$

The above derivation holds for all selection strategies which select individuals based upon their rank in the population. For a random initialized population, $p_{1_{t=0}} = 0.5$ is a reasonable approximation and gives $\theta_0 = 0$. Using binary tournament selection, the selection intensity approximates 0.5642 [13] and the expected convergence time for a 100 bit OneMax becomes 27.84 generations.

Figure 3 plots the model for a 100 bit OneMax and compares this with experimental results. The experiments converge a bit slower due to the build-up of covariances between the alleles . The model becomes more accurate when the alleles are decorrelated by recombining twice at each generation. More details about the creation of genetic covariance together with a refined model for OneMax can be found in [13]. For this paper there is no need to use a more detailed model than the one of Eq. 7.

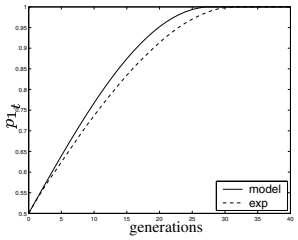


Figure 3: Convergence model and experimental results of the proportion of ones when optimizing a OneMax using tournament selection and uniform crossover. The results are averaged over 50 independent runs.

The convergence model of a OneMax predicts the proportion of ones in the population at every generation. During search this proportion p_{1_t} rises from 0.5, initial population, to 1, optimal population. The spin-flip symmetry in a TwoMax makes it impossible to use p_{1_t} to construct a convergence model for a TwoMax; it has multiple values corresponding to an optimal population. In particular, $p_{1_t} = 1$ corresponds to the optimal population containing only 'all 1s' and $p_{1_t} = 0$ corresponds to the optimal population containing only 'all 0s'. Therefore, we create a new proportion value p_t,

$$p_t = \left|\frac{1}{2} - p_{1_t}\right| + \frac{1}{2}, \quad (9)$$

which gives the proportion of the value which appears the most in the population. When $p_t = 1$, the population contains only ones or only zeros and is converged to one of the optimal populations. For $p_t = 0.5$ several situation are possible including two extreme ones. Firstly, $p_t = 0.5$ corresponds to a population containing only strings with an equal amount of ones and zeros. This population has a minimal mean fitness, $\frac{\ell}{2}$. Secondly, $p_t = 0.5$ corresponds to a third optimal population containing both optima in equal proportion. However, we will see that an srGA without niching never converges to an optimal population containing both optima; therefore p_t will suffice to analyze the search process of a TwoMax.

Before using proportion value p_t to analyze the search process of an srGA optimizing a TwoMax, we apply it to the OneMax problem. Note that for a OneMax individuals with more ones have a higher fitness and are therefore favored by the selection strategy to produce more offspring. Hence, p_{1_t} will always increase during search and its value will never be smaller than its initial value 0.5. This allows us to replace p_{1_t} by p_t in equation 7, yielding convergence model

$$p_t = \frac{1}{2}\left(1 + \sin(\frac{It}{\sqrt{\ell}} - \theta_0)\right), \quad (10)$$

with $\theta_0 = \arcsin(1 - 2p_{t=0})$.

Figure 4 compares the search process of an srGA optimizing a OneMax with the search process of an srGA optimizing a TwoMax using proportion value p_t. The figure shows that except for a small delay during the first few generations in case of the TwoMax, both searches look very similar. In the next section we will analyze the differences and similarities of both searches in more detail.

3.2 TWO PHASES OF A TWOMAX

The individuals in a population running on a TwoMax can be divided into two niches: *niche 0*, containing only individuals with more zeros than ones, and *niche 1*, containing only individuals with more ones than zeros. Experiments show that when optimizing a TwoMax using a selectorecombinative GA only one of the two optima is found. One niche disappears out of the population which allows us to to divide the search process of a TwoMax into two phases:

a first phase during which both niches are still in the population, and a second phase with one niche only.

The first phase starts with a randomly initialized population. Both niches are present in the population, and $p_t = 0.5$. Parents of different niches are likely to produce poor offspring. Their children have an almost equal amount of ones and zeros, and their fitness value varies around $\frac{\ell}{2}$. This slows down the creation of fit offspring. The only good offspring are created by recombination of parents from the same niche. After a few generations one niche will contain more and fitter individuals than the other niche and takes over the population. Figure 5 shows an example of how one niche takes over the population. After 8 generations niche 1 has left the population. Figure 6 (a) shows for each niche the number of individuals and the number of individuals with a fitness value above average. The total number of individuals in the population equals 200.

To estimate the duration of the first phase, we experimentally determine the time when one niche dies out. The results are averaged over 500 runs. Figure 6 (b) shows that the first phase scales up like the *takeover time* [5]. This can be understood as the speed with which a selection strategy takes an initial proportion of best individuals to a substantial share in the population. For binary tournament selection, the time it takes the best individual to take over the population can be written as

$$t^* = \frac{\ln(n) + \ln(\ln(n))}{\ln(2)}, \tag{11}$$

with n the population size. For population size 200, $t^* = 10.049$ generations. Experimental verification shows that for this population size on average one niche disappears after 9.586 generations. In general the first phase takes a bit less than takeover time since the fittest individuals of a niche do not have to takeover the whole population, they only have to make sure that the other niche disappears out of the population.

It is important to note that we cannot use the convergence model with p_t to model the search process during the first phase because the necessary side conditions are not fulfilled. To derive the convergence model from the response to selection equation (equation 5), it is vital that ℓp_t approximates f_t and $\ell p_t(1-p_t)$ approximates σ_t^2 . This is not the case in the first phase. Although, it is possible to end the first phase a bit earlier, for example, when one niche has (almost) no individuals with fitness average above the population's mean fitness.

The second phase is easier to analyze. The search process is now similar to the search process of a OneMax and we can use the convergence model described in equation 10 to model the search and to estimate the duration of

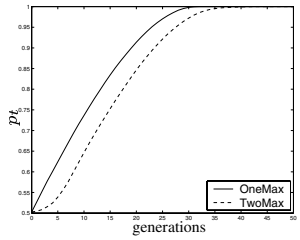


Figure 4: Experimental results of the proportion value p_t when optimizing a OneMax and a TwoMax. All results are averaged over 50 independent successful runs.

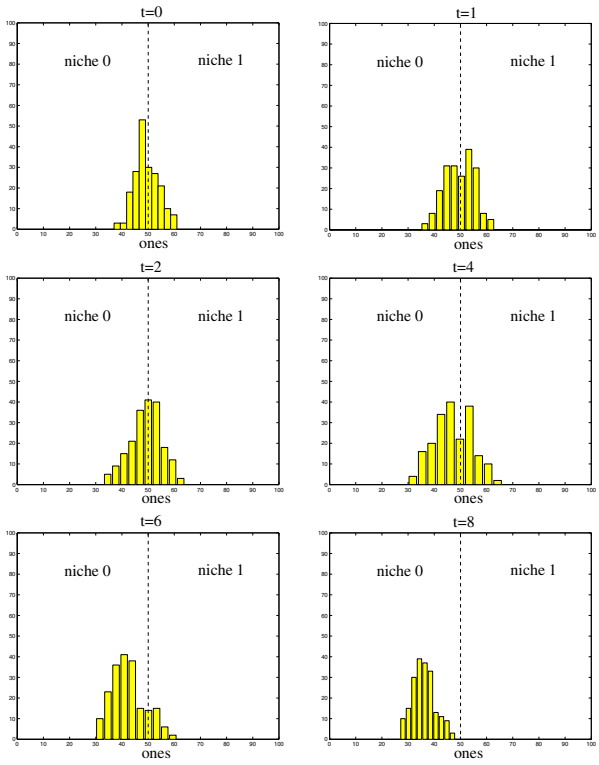
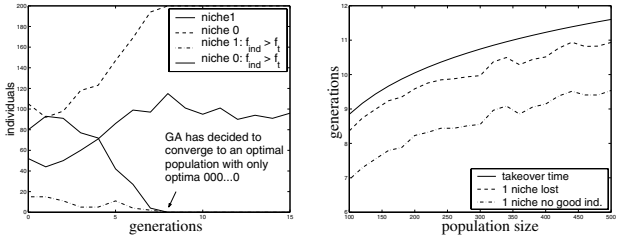


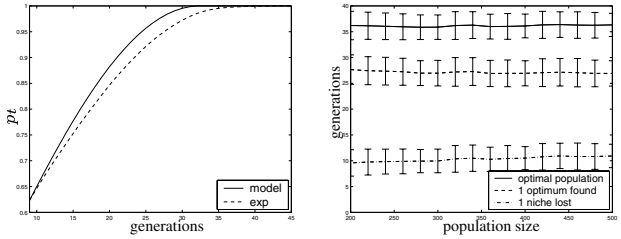
Figure 5: An srGA running on a TwoMax. The histograms have the number of ones in an individual on the horizontal axis and the number of corresponding individuals in the population on the vertical axis. Generations 0, 1, 2, 4, 6 and 8 are displayed.



(a) (b)

Figure 6: (a) shows, for each niche, the number of individuals and the number of individuals with a fitness value above average. (b) shows when one niche has no individuals anymore with fitness value above average and when one niche leaves the population. The results in (b) are averaged over 500 runs.

the second phase. Experiments show that for population size 200, p_t equals 0.6232 at the beginning of the second phase (after about 9 generations). Plugging $p_{t=0} = 0.6232$ ($\theta_0 = -0.2490$) into equation 10 yields the convergence model for the second phase. The calculated convergence time for the second phase is 23.4284 generations. Figure 7 (a) shows the convergence model for the second phase of a 100-bit TwoMax and compares this with experimental results. The experiments again converge a bit slower due to the build-up of covariances between the alleles. Figure 7 (b) shows the end of the first and the second phase for a 100-bit TwoMax using different population sizes. Note that the algorithm finds always an optimum. The results are averaged over 500 runs. The figure shows also the time when the first optima is found.



(a) (b)

Figure 7: (a) Convergence model and experimental results of the p_t when optimizing a TwoMax using tournament selection and uniform crossover. The results are averaged over 50 independent runs. (b) The figure shows the end of the first phase, the time when the first optimum is found and the end of the second phase. The results are averaged over 500 runs.

4 DIFFICULT SYMMETRICAL PROBLEMS

Previous sections showed that when optimizing a TwoMax, the srGA chooses between equally good solutions and benefits from this choice. The population converges quickly to an optimal population containing only one optimum. This is not the case for the search process of an Ising model, a search problem which can be defined by

$$f_{Ising}(s) = \sum_{i=1}^{\ell} \delta(s_i, s_{i+1}), \tag{12}$$

with string length ℓ, $s_{l+1} \equiv s_1$ and

$$\delta(s_i, s_j) = \begin{cases} 1 & \text{if } s_i = s_j, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

The problem contains spin-flip symmetry and has as optima the string containing only ones and the string containing only zeros. The loss of non-inferior BBs in the population prevents a GA (with or without a mutation operator) to optimize an Ising model. In this section we explain the difference in difficulty between easy symmetrical problems, like a TwoMax, and a hard symmetrical problem, like the Ising model. For a more in depth analysis of the Ising model and its difficulties we refer to [14].

Symmetrical problems contain multiple non-inferior BBs. Due to the finite population size, stochastical errors cause that some non-inferior BBs disappear out of the population. This can prevent a GA to find the optimum, as it does in case of the Ising model or H-IFF. The GA gets stuck in a local optimum because the schemata in its current population, which are non-inferior BB of different optima, cannot be combined to form an optimum, and the mutation operator is not strong enough to add diversity to the population. We say that the GA has a *synchronization problem* [15]. Figure 8 shows an srGA with population size 10000 having a synchronization problem while optimizing an Ising model. The diversity measure at the z-axis represents the probability that an individual in the population has a value 1 at a certain string position. The population is not diverse enough to create an optimum.

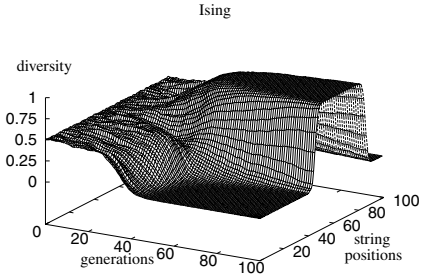


Figure 8: Diversity plot of an srGA optimizing an Ising model with population size 10000. The algorithm uses only two-point crossover.

Comparing the BBs of an Ising model with those of a TwoMax explains why GAs running on the Ising model are more likely to encounter synchronization problems than GAs running on a TwoMax. If we look at BBs whose fixed positions are not all adjacent, we note that BBs like $*00*11*$, which are not BBs of an optimum, and BBs like $*11*11*$, which certainly are BBs of an optimum, have the same average fitness value for the Ising model because there is no interaction between the non-adjacent string positions. This means that they both have the same probability to be selected for reproduction, although one of them is not a BB of an optimum and leads the algorithm to a synchronization problem when diversity is lost. This is not the case for a TwoMax. For TwoMax, BBs with equal values at all fixed string positions have an higher average fitness value than other BBs. The algorithm decides faster to which of the two optima it converges and when diversity is lost, a well sized population still contains the necessary BBs to construct an optimum.

Symmetrical problems which frequently encounter synchronization problems are simple to characterize. In contrary to easy symmetrical problems, an increase in fitness does not automatically imply a decrease in Hamming distance towards the global optimum. Typically, they have highly fit candidate solutions which are at hamming distance far from an optimum and are usually combinations of different optima. Many hierarchical problems satisfy these characteristics and encounter synchronization problems when optimized. The Ising model shows that synchronization problems can also appear when optimizing non-hierarchical problems. A useful strategy to avoid synchronization problems and to keep all non-inferior BBs in the population is niching.

5 NICHING

Fitness sharing [6] is a well-know niching technique that accomplishes subpopulations or niches by degrading an individual's fitness according to the similarity with other individuals in the population. Section 2.1 showed that when using binary tournament selection with continuously updated sharing, the selection process works more in terms of non-inferior BBs and equally good solutions are kept in the population. In this section we try to find the two optima of respectively a 100-bit TwoMax, and a 100-bit Ising model using an srGA with this sharing method. To avoid linkage problems on the Ising model, the algorithm uses two-points crossover. As sharing function the *triangular sharing function* defined in [1] is used. The sharing threshold is calculated by $\sigma_{sh} = \frac{1}{2}(\ell + \frac{1}{q}\sqrt{\ell})$, with q the problem's number of optima. Individuals at Hamming distance smaller than 52.5 share their fitness.

When optimizing an Ising model, equally good partial solutions should be kept in the population. Experiments show that an srGA with sharing and two-point crossover finds both optima quickly. For example, using population size 500, both optima are present in the population after about 79.78 generations. This result is obtained by taking the average over 50 independent runs. Adding a niching technique clearly helps an srGA running on an Ising model to avoid synchronization problems.

An obvious question is now: how does an srGA with sharing perform on easy symmetrical problems, like the TwoMax problem? A TwoMax benefits from genetic drift, and as soon as one niche disappears the algorithm converges quickly to an optimum. When sharing is used to keep both niches in the population, crossover between individuals from different niches produces poor offspring. This has a pernicious influence on the search process. For example, an srGA with sharing, population size 500, and two-point crossover is not able to find an optimum in 2000 generations. When using an undersized population that is not able to keep both niches in the population, sharing only prolongs the first phase of the search process. A soon as one niche disappears, the algorithm converges quickly to an optimal population containing only one optimum. In both cases sharing has no beneficial influence on the search process.

In contrary to an Ising model, a TwoMax becomes much harder when traditional niching techniques are used. The recombination operator becomes too disruptive. This is confirmed by the analysis of an srGA without niching running on a TwoMax. As soon as one niche disappears out of the population, the population's mean fitness increases drastically and the solution is found quickly. Adding a restricted mating technique that uses the Hamming distance as a measure to determine with which individual one can mate, to an srGA with sharing does not improve the performance of the recombination operator. The Hamming distance is not a good measure to separate the individuals with more ones from those with more zeros, and more complex search techniques are necessary. Good results for a TwoMax as well as for an Ising model are found in [11] where a clustering technique is used in combination with an univariate marginal distribution algorithm. The disruptive effect of recombination is eliminated by only allowing recombination between individuals of the same cluster and niching is induced by allowing each cluster to produce an amount of offspring proportional to their size or to their fitness average. In this way, the individuals from niche 0 and niche 1 have their own cluster, and since the individuals only mate within their own cluster, each cluster is actually solving a OneMax (or a ZeroMax).

6 SUMMARY AND CONCLUSION

This paper shows that there exists a key dividing line between two types of symmetrical problems: problems for which the GA benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum. By analyzing the search process of a selectorecombinative GA optimizing a TwoMax and comparing this search process with that of an Ising model, we tried to understand the difference between these two types of symmetrical problems. For the first type of problem, searching in terms of superior building blocks is sufficient to find an optimum. For the second type of problems, searching in terms of non-inferior building blocks is necessary to find an optimum and can be obtained by using a niching technique.

In the context of designing a robust problem solver which solves problems of bounded difficulty, designing an algorithm which solves both types of symmetrical problems quickly, accurately, and reliably is not obvious. Naively adding fitness sharing as a standard component of a GA makes easy symmetrical problems like a TwoMax hard to solve. When using traditional niching techniques, the recombination operator is too disruptive, and more complex methods are necessary. Next to looking for more complex methods that solve both types of problems, we suggest looking for an a priori or runtime detector which discriminates between the two types of problems and selects the appropriate algorithm automatically.

Acknowledgments

The authors would like to thank Alessio Ceroni, Martin Pelikan and Kumara Sastry for many useful discussions. The first author is supported by the Flemish Institute for the Encouragement of Scientific and Technological Research in Industry – (IWT) (Flanders) (Belgium).The third author is a Post Doctoral Fellow of the Fund for Scientific Research – (FWO) (Flanders) (Belgium). The work was partly sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U.S. Army, or the U.S. Government.

References

[1] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufmann, 1989.

[2] L. J. Eshelman and J. D. Schaffer. Crossover's niche. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 9–14. Morgan Kaufmann, 1993.

[3] D. E. Goldberg. Four keys to understanding building-block difficulty, June 15 1998. Presented in Project FRACTALES Seminar at I.N.R.I.A. Rocquencourt, Le Chesnay, Cedex.

[4] D. E. Goldberg. The design of innovation: Lessons from and for competent genetic algorithms. In press, 2002.

[5] D. E. Goldberg and K. Deb. A comparative study of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[6] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[7] J. H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):372–391, 2000.

[8] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[9] B. Naudts and J. Naudts. The effect of spin-flip symmetry on the performance of the simple GA. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 67–76, Berlin Heidelberg New York, 1998. Springer-Verlag.

[10] C. K. Oei, D. E. Goldberg, and S. J. Chang. Tournament selection, niching, and the preservation of diversity. IlliGAL Report 91011, University of Illinois at Urbana-Champaign, 1991.

[11] M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering and the breaking of symmetry. IlliGAL Report 2000013, University of Illinois at Urbana-Champaign, 2000.

[12] M. Pelikan and D. E. Goldberg. Hierarchical problem solving and the bayesian optimization algorithm. In D. Withley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 2000*, pages 267–274. Morgan Kaufmann, 2000.

[13] D. Thierens. *Analysis and Design of Genetic Algorithms*. PhD thesis, Catholic Univerity Leuven, Belgium, 1995.

[14] C. Van Hoyweghen, D. E. Goldberg, and B. Naudts. Building block superiority, multimodality and synchronization problems. In Lee Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, pages 694–701. Morgan Kaufmann, 2001.

[15] C. Van Hoyweghen, B. Naudts, and D. Goldberg. Spin-flip symmetry and synchronization. Submitted to the journal of Evolutionary Computation, 2001.

[16] R. Watson, G. S. Hornby, and J. B. Pollack. Modeling building block interdependency. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, pages 97–106. Springer-Verlag, 1998.

Simulating Gender Separation with Genetic Algorithms

Dana Vrajitoru
IUSB, Computer & Information Sciences
1700 Mishawaka Ave, P.O.Box 7111
South Bend, IN 46634-7111
email: danav@cs.iusb.edu

Abstract

Gender separation is a largely encountered in the natural systems that allows the preservation of the genetic diversity in a species. In this paper, we want to analyse the mechanism by which this reproduction mode may have evolved and the way it influences the evolution of a population toward an optimal individual.

1 INTRODUCTION

It is a known fact that gender separation is a powerful biological tool that helps preserving the genetic diversity in a population by preventing organisms from self-duplicating. It has also been shown that allogen reproducing organisms have a greater capacity to quickly adapt to difficult environments and to resist hostile exterior factor like disease (Hamilton, Axelrod, and Tanese, 1990).

The gender separation and sexual reproduction have been of interest in the study and application of genetic algorithms (GAs), since they are an important feature of the living organisms. Some research has followed the direction of sexual reproduction and the impact on the biodiversity (Todd and Miller, 1997; Todd, 1997).

In another direction, sexual reproduction can also be viewed as a factor for developing the communication through the emergence of complex mating signals (Werner and Todd, 1997; Noble, 1999).

Gender can also be seen as a factor for developing the collaboration between the individuals in a population (Hemelrijk, 1999), and can be a component for multi-optimization algorithms (Allenson, 1992).

In this paper, we try to model the way gender separation has occurred in nature in the first place and how it became such a wide-spread feature in our ecosystem. Our hypothesis is that the reproduction constraints related to the sexual reproduction alone cannot explain the survival and spreading of this reproduction mode in nature. Thus, the individuals that initially developed mating schemes that aim to preserve the genetic diversity in a population can eventually lead to the emergence of two distinct subpopulations that reproduce in general with each other.

Section 2 presents a model in which the distribution of gender types in a given population evolves along with the rest of the genotype. Section 3 examines the fitness of the individuals that evolve following various reproduction modes. Section 4 studies the influence of the population size on the performance of each of the models.

2 EVOLVING THE REPRODUCTION MODE

In this section present a simulation model based on several gender types that can evolve through genetic operations along with the rest of the chromosome.

2.1 MODEL DESCRIPTION

The first experiment that we introduce in our paper concerns the evolution of various gender types through natural selection. We start from the assumption that the initial population is composed in an equal way of individuals presenting various reproduction constraints, that we classify as gender types. In the mating process, some of them require specifically a partner of different gender in order to reproduce, while others can mate with a partner of any gender. Part of them can even self-duplicate.

The gender type is inherited by the offspring from the parents in a random fashion with no particular prefer-

ence. The goal of the experiment is to analyse the composition of the population as it evolves toward individuals of higher fitness through the mechanism of natural selection. Thus, the gender distribution in the final population is determined by two factors: the probability of occurrence of each gender type in the next generation based on the reproduction constraints, and the gender inherited by the fittest individuals in a given generation, which are more likely to be chosen for reproduction.

In our model, we have four types of individuals: male (M), female (F), self-fertilizing (S-F), and hermaphrodite (H). The gender of the individual is assigned by random in the initial population with no preference for any of the four types. This parameter is inherited by the offspring and thus evolves by natural selection.

Each of the four gender types has a different constraint in the mating process. The self-fertilizing individuals can mate with any individual regardless of their gender, including themselves. The hermaphrodite individuals can mate with any individual other than themselves. The males and females can mate an individual of any gender type different from their own. Thus, we suppose that in the initial conditions of gender separation, a male or a female could mate with either a female, or with a hermaphrodite individual presenting the features of both gender types.

The male - female mating constraints can lead the algorithm to a deadlock in the case where the population becomes exclusively composed of one of these two gender types, such that the reproduction is not possible anymore. We prevent these situations by introducing the possibility of spontaneous sex change from male to female or the reverse in the situation where an individual has made a number of unsuccessful mating attempts equal to the least of 25 and the quarter of the population size.

The offspring inherits the gender type of the parents randomly, with an equal probability assigned to each parent. Let us denote by n_{0M}, n_{0F}, n_{0S}, and n_{0H} the number of male, female, self-fertilizing, and hermaphrodite individuals respectively in a given generation. We can compute the expected number of occurrences of each gender type n_{1M}, n_{1F}, n_{1S-F}, and n_{1H} in the next generation considering exclusively the mating scheme we have described and ignoring the mechanism of natural selection. Let $n_{0M} + n_{0F} + n_{0S-F} + n_{0H} = n$, the size of the population.

The general reproduction scheme that we have used consists in selecting two parents for reproduction by crossover, and producing exactly two children by this operation. Since the gender type is randomly inherited by the offspring, we expect the children to have the same gender distribution as the parents. For example, from a male and a hermaphrodite we expect to produce again a male and a hermaphrodite, only with a different genetic code.

The male and female individuals can be found in the next generation either if they are selected as the first parent, or if the first parent has a different gender than their own and they are selected as a second parent. The probability that a random individual from the initial population is a male is equal to n_{0M}/n, and the probability of the contrary event is $(n - n_{0M})/n$. Considering that we select $n/2$ individuals as the first parent and the same number as the second parent, we can compute the expected number of occurrences of the male gender type in the next generation as

$$\begin{aligned} n_{1M} &= \frac{n}{2}\frac{n_{0M}}{n} + \frac{n}{2}\frac{(n - n_{0M})}{n}\frac{n_{0M}}{n} \\ &= n_{0M}\frac{2n - n_{0M}}{2n} < n_{0M} \end{aligned}$$

The same computation can be done for female individuals.

$$\begin{aligned} n_{1F} &= \frac{n}{2}\frac{n_{0F}}{n} + \frac{n}{2}\frac{(n - n_{0F})}{n}\frac{n_{0F}}{n} \\ &= n_{0F}\frac{2n - n_{0F}}{2n} < n_{0F} \end{aligned}$$

We can find a hermaphrodite in the new generation if a hermaphrodite was chosen as a first parent, or if the second parent is a hermaphrodite and not identical to the first one. We have

$$\begin{aligned} n_{1H} &= \frac{n}{2}\frac{n_{0H}}{n} + \frac{n}{2}\frac{n - n_{0H}}{n}\frac{n_{0H}}{n} + \frac{n}{2}\frac{n_{0H}}{n}\frac{n_{0H} - 1}{n} \\ &= n_{0H}\frac{2n - 1}{2n} < n_{0H} \end{aligned}$$

In this formula, the first term of the sum corresponds to the event that the first parent is hermaphrodite. The second term corresponds to the event that the first parent is not a hermaphrodite, but the second parent is, in which case they cannot be the same individual. The third term corresponds to the event that the both parents are hermaphrodites and different individuals.

The number of self-fertilizing individuals should remain stable, since they can constitute any of the parents, regardless of what the other parent is.

$$n_{1S-F} = 2\frac{n}{2}\frac{n_{0S-F}}{n} = n_{0S-F}$$

As the sum of these numbers is inferior to n, we should scale them correspondingly with a factor greater than 1. This would most probably modify the expected number of hermaphrodite individuals in the new generation such that $n_{1H} \geq n_{0H}$. The result is that the number of self-fertilizing and hermaphrodite individuals are expected to increase, while the two others may converge to 0.

2.2 EXPERIMENTAL CONDITIONS

This section presents the fitness functions that the GA must optimize and the parameter settings that we have used for the experiment.

Fitness Functions

We have chosen three classes of problems: the set of 10 standard test functions, an NP-complete problem and several deceptive functions. Each class presents a special challenge for the GA.

Standard functions This class contains 10 standard minimization functions used in many cases to experiment with GAs (Whitley et al., 1996).

Deceptive problems This class of problems is based on the phenomenon of deception (Whitley, 1990; Deb and Goldberg, 1994) and contains problems that are known to be difficult for GAs. For this reason, they are a frequent choice as test functions in the study of GAs (Goldberg, Deb and Horn, 1992; Kingdon and Dekker, 1995; Mohan, 1998). Their difficulty comes from the fact that the optimal individual is isolated from other individuals of high performance, and there are one or more suboptimal individuals that are easier to reach by hill-climbing.

We have chosen 8 deception problems that consist in concatenating a number of 3-bit functions as shown in Table 1. For these problems, the optimal individual is represented by a string of 3 bits whose closest neighbors display the lowest performance. We have conducted our experiences with individuals composed of 100 strings of 3 bits, so the optimal individual should have a performance of 3000.

Hamiltonian circuit (HC) Given a graph, does there exist a circuit that passes once and only once through each vertex? This problem is known to be NP-complete (Brassard and Bratley, 1994).

We have performed our experiences with 10 HC problems with graphs of 9 to 150 vertices and up to 3000 edges. The direct representation of a HC problem for the GAs can be difficult. De Jong and Spears (1989) suggest to transform the HC instances into SAT instances, whose genetic representation is easier.

Table 1: Deception functions

	000	001	010	011	100	101	110	111
decep1	28	26	22	0	14	0	0	30
decep2	28	26	22	14	14	26	22	30
decep3	22	0	28	26	0	30	14	0
decep4	0	14	30	0	26	28	0	22
decep5	22	14	28	26	22	30	14	26
decep6	26	14	30	22	26	28	14	22
decep7	22	14	28	26	14	30	24	14
decep8	14	22	30	14	24	28	14	26

SAT (Boolean satisfiability) Given a Boolean expression depending on some variables, does there exist an assignment to those variables such that the expression evaluates as true?

A detailed description of the reduction of a HC instance into a SAT instance can be found in (Brassard and Bratley, 1994) or (Vrajitoru, 1999). For any given graph, a Boolean variable corresponds to each edge, and is given the true value if the edge belongs to the HC. Thus, if we represent each variable as a gene, the size of an individual is equal to the number of edges in the graph.

To evaluate a Boolean expression to more than true or false, we have used fuzzy logic measures, also proposed by De Jong and Spears (1989). The 'and' operation is evaluated to the average of the terms, while the 'or' operation returns the maximum of the terms. Thus, the fitness function takes values between 0 and 1, and the optimal individual has a fitness of 1.

2.3 PARAMETER SETTINGS

We have performed for each problem 100 runs of the GA, half of which without mutation, and half with a mutation rate of 0.01. The crossover rate is equal to 1 in all the cases. Each generation contains 50 individuals and the number of generations is limited to 500. We have used the fitness proportionate or roulette wheel selection (Goldberg, 1989), and a variant of the elitist reproduction called monotone: the worst individual in the new generation is replaced by the ancient best individual, if and only if the new generation contains nothing better than it (Vrajitoru, 1999).

Table 2: Gender convergence in 500 generations

	M/F	H	S-F
Standard	9	438	553
Deception	6	370	424
HC	11	973	1116
Total	26	1781	2093
%	0.67	45.67	53.66

We have used a crossover operator called *combined balanced* that combines four variations of crossover in each generation : the 1-point (Holland, 1975), 2-point (De Jong, 1975), uniform (Syswerda, 1989), and dissociated (Vrajitoru, 1999). For each operation, one of the four crossover forms is used by a random choice giving each of them equal chances.

2.4 SIMULATION RESULTS

The parameter that interests us in this first experiment is the composition of the population in the last generation. The simulation has shown that the distribution of the gender types in the population converges to one of the four types very fast.

To illustrate this fact, Figure 1 shows the evolution of the gender types in 30 generations, computed over 80 runs on 4 HC problems of various difficulty. The measure that we have plotted is the average number of occurrences of each type in the population according to the generation number.

From this figure we can see that the male and female gender types have completely disappeared after 30 generations. In 77 out of 80 runs, the population is already completely composed of one of the two remaining gender types. In the conditions of our experiment, the gender converges much more rapidly than the rest of the genetic material.

A second set of results shows that after 500 generations, the population is composed completely of one gender type in all the cases. Table 2 shows the distribution of each gender type in the last generation for each class of problems. We can remark that although the general distribution of gender types follows the tendency that we have also observed in Figure 1, the number of resulting populations that are composed of male/female individuals is not equal to the expected number (0).

This fact is due to the influence of the natural selection mechanism. Thus, the best individuals are more likely to be chosen for mating, and their gender type will be transmitted to a greater number of the new individuals than expected from the mating constraints.

This factor can explain the mechanism by which a genetic feature that restrains the potential number of individuals with which a given individual can mate, has survived and currently dominates an important part of our natural system. We can entail that the first organisms developing only one gender type also presented some adaptation to their environment that made them stronger and allowed the feature to survive and spread.

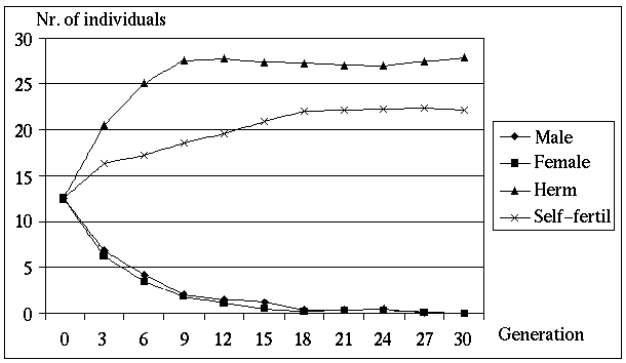


Figure 1: Gender evolution in 30 generations

3 COMPARING THE REPRODUCTION MODES

In this section we compare several reproduction schemes based on the gender types previously described. We are interested in the influence of the self-fertilization and of the gender separation on the performance of the GAs.

3.1 REPRODUCTION SCHEMES

We propose four reproduction schemes that we compare using the average performance over 100 runs on each test problem.

The first reproduction scheme is the usual one based on the fitness-proportionate selection. In this case, all the individuals are self-fertilizing, and thus the process of mating doesn't require any special operation. We denote this scheme by *simple*.

The second scheme is the one presented in the previous section, where the gender types belong to any of the four categories. They are assigned randomly in the initial population and evolve through generations. We denote this scheme by *mixed*.

The last two schemes, denoted by *gender* and

Table 3: Results on the standard test functions

Problem	Simple	Mixed	Gender	Herm
F1	0.192	0.177	0.155	0.182
F2	0.449	0.31	0.59	0.365
F3	11.85	11.84	11.97	11.83
F4	2.894	3.053	2.751	3.013
F5	5.14	6.208	4.765	3.943
F6	3.205	3.407	3.228	3.475
F7	616.627	620.549	616.096	596.507
F8	2.115	2.306	2.429	2.237
F9	0.125	0.132	0.13	0.138
F10	1.456	1.725	1.357	1.622

Table 4: Results on deception problems

Problem	Simple	Mixed	Gender	Herm
d1	2684.36	2697.44	2699.16	2689.32
d2	2851.22	2929.4	2853.84	2857.02
d3	2494.2	2502.12	2504.92	2503.88
d4	2504.94	2498.54	2501.22	2500.7
d5	2767.86	2771.5	2767.1	2772.42
d6	2764.82	2765.42	2769.24	2768.92
d7	2650.18	2652.48	2653.22	2651.22
d8	2665.78	2667.86	2666.9	2668.4

herm, are based on populations completely formed of male/female and hermaphrodite individuals respectively. For both of them, an individual cannot self-duplicate. For the first one, the mating choice is limited to about half of the population.

3.2 EXPERIMENTAL RESULTS

Table 3 presents the average performance over 100 runs in 500 generations achieved by each reproduction scheme on the set of standard function. Each problem in this class is a minimization problem, so that the smaller results are the best ones.

Table 4 presents the results under the same conditions on the deception problems. This class contains optimization problems, where the maximal performance is equal to 3000.

Table 5 shows the results of the reproduction schemes on the HC problems. The maximal performance in this case is equal to 1.

From these tables we can notice that the best average performance is obtained almost in all the cases by either the two-gender or the hermaphrodite populations.

Table 5: Results on HC problems

Problem	Simple	Mixed	Gender	Herm
hc9	0.968	0.97	0.972	0.97
hc10	0.966	0.966	0.966	0.964
hc11	0.965	0.963	0.964	0.965
hc12	0.963	0.962	0.962	0.962
hc13	0.96	0.96	0.961	0.962
hc14	0.959	0.959	0.962	0.961
hc15	0.959	0.959	0.96	0.96
hc20	0.958	0.958	0.958	0.957
hc25	0.953	0.952	0.953	0.953
hc30	0.944	0.943	0.943	0.944
hc50	0.937	0.937	0.937	0.937
hc60	0.94	0.938	0.938	0.939
hc70	0.939	0.939	0.939	0.939
hc80	0.94	0.94	0.94	0.94
hc90	0.94	0.94	0.94	0.942
hc100	0.94	0.939	0.939	0.94
hc110	0.942	0.941	0.94	0.942
hc120	0.938	0.938	0.939	0.94
hc130	0.939	0.939	0.939	0.937
hc140	0.935	0.935	0.936	0.937
hc150	0.941	0.94	0.935	0.938

The mixed scheme, where the gender type evolves by genetic operations, is almost never the best one. We can deduce that avoiding self-fertilization can improve the performance of GAs.

Table 6 summarizes the results presented in the previous tables, by counting the number of times each scheme presents the best average performance for each class of problems (the columns marked by μ). We have also considered the number of problems where each scheme presented the best result for the optimal run out of 100 (the columns marked by *opt*). This table emphasizes the conclusions we have made based on the previous tables, and shows that the best reproduction scheme for the GAs is the hermaphrodite. This means that for the size of the population we have considered, it is better to allow a wider mating choice for each individual, while avoiding the self-duplication.

From this second set of experiments, we can deduce that given the success of gender separation in our natural system, this factor must contribute to the capacity of survival and of adaptation of the individuals to their environment in other ways than just providing a means of preserving the genetic diversity.

One of the possible advantages of gender separation is the development of complex communication sys-

Table 6: Summary of results

	standard		deception		HC		Total
	μ	*opt*	μ	*opt*	μ	*opt*	
Simple	3	3	1	3	7	5	22
Mixed	1	2	1	2	2	3	11
Gender	4	1	4	3	4	3	19
Herm	2	4	2	0	8	10	26

tems that are enhances by the necessities of finding a mate. A second factor could be the fact that the biological differences between the two genders has given way to the raise of a social structure for some of the more evolved organisms and to the distribution of the survival tasks. This phenomenon has evolved into the complex social systems of the human beings, and is also a positive factor for developing the intelligence and for preserving the experience accumulated through generations. These hypothesis constitute the subject of a future research.

4 THE ROLE OF THE POPULATION SIZE

The previous experiences have shown that in the given conditions, the gender separation can be slowing down the search for the optimal individual. This is probably due to the fact that during the crossover operation, a given individual can only mate with half of the individuals in the population. Thus, the best individuals that were found so far may not be available in the process because they are of the same gender as the first parent selected for reproduction.

In the natural habitat, the populations are in general much larger than the experimental size for the GAs. When the population size increases and if there is a balanced distribution of the two genders in the population, it is possible for an individual to find an individual of high fitness among the opposite sex.

Our hypothesis is that the mating restriction inherent to the gender separation presents a significant disadvantage only in the conditions where the population size is small. In the following experiences we will analyze the impact of the population size on the model with gender separation and on the model where the individuals are hermaphrodite but cannot mate with themselves.

We have performed experiences with a HC problem based on a graph with 150 vertices. This problem presents an interesting challenge because the size of the individual is around 3000 genes according to the number of edges in the graph.

Table 7: Results on HC problems

Population size	Simple	Gender	Herm
50	0.961	0.962	0.963
60	0.968	0.969	0.968
70	0.970	0.971	0.970
80	0.972	0.972	0.972
90	0.973	0.973	0.973
100	0.973	0.974	0.974

Table 7 shows the performance of the simple self-fertilizing model, of the model with gender separation, and of the model with hermaphrodite individual that cannot reproduce with themselves. From this table, we can see that in general, for a difficult problem, avoiding the self-fertilization can help the GA in finding fitter individuals.

Figure 2 shows the evolution of the performance of the two models that avoid self-fertilization (gender separation and hermaphrodite) in 500 generations as the population size increases. The measure that we have considered is the average fitness based on 10 experiences in the same conditions. From this figure we can remark that the difference between the performance of the models decreases as the population size increases. A second remark is that in a given number of generations, both models can find individuals of better fitness when the population is larger.

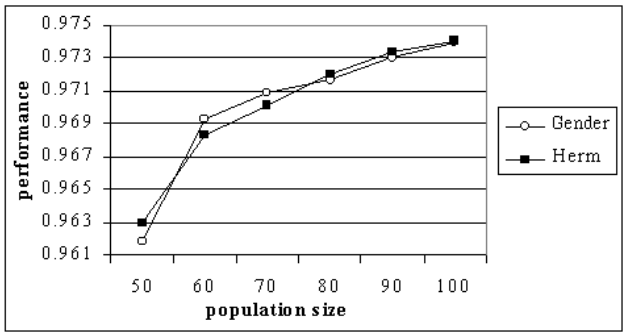


Figure 2: Comparing performance of the models according to the population size

To emphasize the effect of the population size on the performance of the models that we are interested in, Figure 3 shows the evolution of the absolute difference between the average performance of the gender separated model and that of the hermaphrodite model. We can clearly see that as the population size increases,

the difference between the average performance of the two models is dramatically reduced.

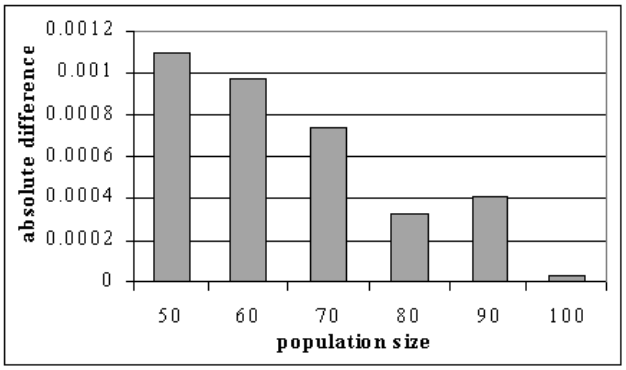


Figure 3: Absolute difference between the models according to the population size.

The experiments introduced in this section have shown that although the hermaphrodite model can present an advantage over the model with gender separation, as suggested by the previous experiences, this advantage seem to disappear as the population size increases.

This phenomenon can explain the large occurrence of gender separation in nature, where the size of the populations is much higher than in our simulations. We can also observe a disadvantage of gender separation for some of the endangered species, where the population size is too small. For some of these species, like the Panda bears, finding a mate can be a real challenge because of the small number of available individuals in a given geographical area.

5 CONCLUSIONS

In this paper, we have used the GAs to analyse some of the aspects related to a phenomenon that is largely encountered in the natural system, the gender separation.

Section 2 has introduced a model in which the gender type evolves through the genetic operations as the individuals try to adapt to a specific problem. From these experiments we can deduce that the first individuals that developed this feature must have presented some adaptation advantages over the others to allow this feature to survive.

Section 3 compares the quality of the individuals that evolve through various reproduction modes, some of which are based on gender separation. The results presented in this section suggest that for small size populations, gender separation can slow down the rate by which the individuals can adapt to a problem.

To continue the idea from the second experiment, Section 4 examines the influence of the population size on the performance of each of the models. These last results suggest that the difference between the models concerning the fitness of the evolved individuals becomes insignificant as the population size increases. Moreover, for a larger size of the population, avoiding the self-duplication can be an advantage.

Acknowledgments

The experiments presented in this paper have been partially developed in the LCVM^2 laboratory, EPFL, Switzerland.

References

[Allenson, 1992] Allenson, R. (1992). Genetic algorithms with gender for multi-function optimisation. Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland.

[Brassard and Bratley, 1994] Brassard, G. and Bratley, P. (1994). *Fundamentals of Algorithmics*. Prentice-Hall.

[De Jong, 1975] De Jong, K. (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

[De Jong and Spears, 1989] De Jong, K. and Spears, M. (1989). Using genetic algorithms to solve NP-complete problems. In *Proceedings of the International Conference on Genetic Algorithms*, pages 124–132, Fairfax (VA). George Mason University.

[Deb and Goldberg, 1994] Deb, K. and Goldberg, D. E. (1994). Sufficient conditions for arbitrary binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (MA).

[Goldberg et al., 1992] Goldberg, D. E., Deb, K., and Horn, J. (1992). Massive multimodality, deception and genetic algorithms. In Manner, R. and Manderick, B., editors, *Proceedings of Parallel Problem Solving from Nature II*, pages 37–46.

[Hamilton et al., 1990] Hamilton, W., Axelrod, R., and Tanese, R. (1990). Sexual reproduction as an adaptation to resist parasites. *Proceedings of the National Academy of Sciences*, 87:3566–3573.

[Hemelrijk, 1999] Hemelrijk, C. (1999). Effects of cohesiveness on intersexual dominance relationships and spatial structure among group-living virtual entities. In *Proceeding of the European Conference on Artificial Life V*, pages 524–534. Springer Verlag.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[Kingdon and Dekker, 1995] Kingdon, J. and Dekker, L. (1995). The shape of space. In *Proceedings of the Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, pages 543–548, London (UK). IEE.

[Mohan, 1998] Mohan, C. K. (1998). Selective crossover: Towards fitter offspring. In *Proceedings of the Symposium on Applied Computing (SAC'98)*, Atlanta (GA).

[Noble, 1999] Noble, J. (1999). Sexual signalling in an artificial population: When does the handicap principle work? In Floreano, D., Mondada, F., and Nicoud, J.-D., editors, *Proceeding of the European Conference on Artificial Life V*, pages 644–653. Springer Verlag.

[Syswerda, 1989] Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the International Conference on Genetic Algorithms*, San Mateo (CA). Morgan Kaufmann Publishers.

[Todd, 1997] Todd, P. (1997). Searching for the next best mate. In Conte, R., Hegselmann, R., and Terna, P., editors, *Simulating social phenomena*, pages 419–436, Berlin. Springer-Verlag.

[Todd and Miller, 1997] Todd, P. and Miller, G. (1997). Biodiversity through sexual reproduction. In Langton, C. and Shimohara, K., editors, *Proceedings of Artificial Life V*, pages 289–299, Cambridge (MA). MIT Press/Bradford Books.

[Vrajitoru, 1999] Vrajitoru, D. (1999). Genetic programming operators applied to genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 686–693, Orlando (FL). Morgan Kaufmann Publishers.

[Werner and Todd, 1997] Werner, G. and Todd, P. (1997). Too many love songs: Sexual selection and the evolution of communication. In Husbands, P. and Harvey, I., editors, *The Fourth European Conference on Artificial Life*, Cambridge (MA). MIT Press/Bradford Books.

[Whitley, 1990] Whitley, D. (1990). Fundamental principles of deception in genetic algorithms. *Foundations of Genetic Algorithms*, pages 221–241.

[Whitley et al., 1996] Whitley, D., Mathias, K., Rana, S., and Dzubera, J. (1996). Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276.

A Fixed Point Analysis of a Gene Pool GA with Mutation

Alden H. Wright *
Computer Science
University of Montana
USA
wright@cs.umt.edu
44 121 414 2793

Jonathan E. Rowe
Computer Science
University of Birmingham
UK
j.e.rowe@cs.bham.ac.uk

Riccardo Poli
Computer Science
University of Essex
UK
rpoli@essex.ac.uk

Christopher R. Stephens
Instituto de Ciencias Nucleares
UNAM, Mexico
stephens@nuclecu.unam.mx

Abstract

This paper analyzes a recombination/mutation/selection genetic algorithm that uses gene pool recombination. For linear fitness functions, the infinite population model can be described by ℓ equations where ℓ is the string length. For linear fitness functions, we show that there is a single fixed point and that this fixed point is stable. For the ONEMAX fitness function, the model reduces to a linear recurrence in a single variable which can be explicitly solved. The time-to-convergence for ONEMAX is given.

1 Introduction

A major goal of genetic algorithm theory is a tractable model of a GA that gives quantitative predictions over multiple generations. The Vose dynamical system model [9] is exact in the infinite-population limit, but it tends to be intractable for results on specific problems due to the fact that the model keeps track of the frequency of every string. What is needed is a "coarse-grained" model that simplifies the model. One natural way to attempt to coarse-grain a GA model is to look at the representations of schemata, especially low-order schemata. (See [7] and [10].) One would like to track schema averages over multiple generations.

Crossover does not change the schema frequencies of order 1 schemata. Stated another way, it preserves the frequencies of each allele. The effect of crossover on a population is to move the population closer to linkage equilibrium. In other words, it decorrelates the alleles at different positions. In a linkage equilibrium population, the representation of any string is determined by the allele frequencies. Geiringer's theorem [1] shows that the limit of repeated applications of crossover is a population in linkage equilibrium (also known as Robbins' proportions).

In gene pool recombination, the population is taken directly to linkage equilibrium in one step. In other words, the alleles are completely decorrelated, and the population is completely described by the allele frequencies. This can be implemented for a finite population by choosing the genes for a new individual from a pool constructed from the whole population. In other words, the population allele distribution at a particular locus defines a probability distribution, and the allele at that position in a new individual is selected from that probability distribution. Gene pool recombination was introduced by Voigt and Mühlenbein in [8].

After a gene pool recombination step, the complete state of the population is determined by the allele frequencies, and the number of these variables is linear rather than exponential in the string length. This "coarse-graining" makes the gene pool much more tractable for analysis than the two-parent recombination GA, and this is the focus of this paper.

Mühlenbein, Mahnig and others have done considerable work on the Univariate Marginal Distribution Algorithm (UMDA) [6], [4]. This algorithm uses gene pool recombination, and selection, and no mutation. Response to selection is computed or approximated for a variety of selection methods. In [3], Mahnig and Mühlenbein introduce mutation into UMDA by a technique called Bayesian prior. The emphasis is on the interaction of the mutation rate and the population size.

By using the Walsh basis, we are able to analyze a GA that uses gene pool recombination, selection, and mutation. We find and rigorously prove the stability of fixed points which is a different approach than has been taken in the work on UDMA which is primarily oriented towards "response to selection". Our work shows how the Vose dynamical system model framework can be used to model algorithms like UMDA.

We will call the GA that uses gene pool recombination,

*This paper was written while Alden Wright was visiting the School of Computer Science, University of Birmingham, UK, supported by EPSRC grant GR/R47394.

selection, and mutation a gene pool GA.

The paper considers linear fitness functions. Other classes of fitness functions, such as the class of single-peak (needle-in-the-haystack) fitness functions will be considered elsewhere.

There has been a long history of research on the analysis of fitness functions by means of their Walsh coefficients. The Vose model [9] uses the Walsh transform to simplify the model for crossover and mutation. This paper makes a connection between these two lines of research. It should be the first step in a research program that uses the Walsh transform to analyze selection/mutation/recombination genetic algorithms.

In addition, the gene pool GA can suggest general properties of recombination that can then be tested for a GA that uses two-parent recombination.

We show that for any linear fitness function over fixed length binary strings, there is a unique fixed point which is asymptotically stable. For the ones-counting (ONEMAX) fitness function, explicit formulas are given for all fixed points of the $\mathcal{G}$ function that defines the infinite population model. To our knowledge, this is the first time that formulas for fixed points have been given for a GA that involves selection, recombination, and mutation and for arbitrary string length.

A gene pool recombination GA can either be used as an approximation to a two-parent recombination GA or as an alternative GA. We show empirically that the gene pool GA is a good approximation to a two-parent recombination GA for linear fitness functions.

For all results where separate proofs are given, the proofs are in the appendix.

2 Notation

Notation in this paper mostly follows [9].

The search space for this paper is the set of all binary strings of length ℓ which will be denoted Ω. The binary representation of a string induces a correspondence from the elements of Ω to the set of integers from 0 to $2^\ell - 1$. Thus, the integer 0 corresponds to the all-zeros string, and the integer $2^\ell - 1$ corresponds to the all-ones string. A sum over $i \in \Omega$ is equivalent to a sum from $i = 0$ to $i = 2^\ell - 1$.

The bitwise mod 2 sum of two strings j and k is denoted by $j \oplus k$; Ω is a group under this operation. Note that $j \oplus k$ is also the XOR of j and k. The identity element is the string of zeros, which will be denoted by 0. The bitwise product of strings j and k is denoted by $j \otimes k$. The ones-complement of k is denoted by $\overline{k}$.

The number of ones in a binary string k is denoted by $\#k$.

For each $u \in \Omega$, $\Omega_u = \{i \in \Omega : i \otimes u = i\}$; Ω_u is the set of binary strings which have a 1 only in positions where u has a 1. It is also a schema denoted by a string over $\{0,*\}$ where there are asterisks in those positions where the corresponding bit of u is 1, and where the fixed positions are all zeros. For example, if $\ell = 4$ and $u = 0101$, then Ω_u is the special schema $0*0*$, and is also the set $\{0000, 0001, 0100, 0101\}$. Note that Ω_u is a subgroup of Ω.

Let $\mathcal{L} = \{j \in \Omega : \#j = 1\}$. Under the identification of Ω with the integers, $\mathcal{L} = \{2^k : k = 0, 1, \ldots, \ell - 1\}$. Let $\mathcal{L}_u = \{j \in \mathcal{L} : j \otimes u = j\} = \mathcal{L} \cap \Omega_u$. For example, if $\ell = 6$ and $u = 21$, $\mathcal{L}_u = \{2^0, 2^2, 2^4\} = \{1, 4, 16\} = \{000001, 000100, 001000\}$.

The set $\mathcal{L}$ will be used extensively as an index set. This might seem unnatural since $\mathcal{L}$ is a subset of Ω which does not correspond to consecutive integers. If this bothers the reader, a product over $i \in \mathcal{L}$, such as $\prod_{i \in \mathcal{L}} S_i$, could be rewritten as $\prod_{k=0}^{\ell-1} S_{2^k}$.

A population (a multiset of Ω) is represented as a population vector x indexed by Ω; $x_v = x_{v_0 v_1 \ldots v_{\ell-1}}$ is the fraction of the population which is string $v = v_0 v_1 \ldots v_{\ell-1}$, where $v_0, v_1, \ldots, v_{\ell-}$ are the bits of v. For example, if $\ell = 2$ and the population as a multiset is $\{00, 01, 01, 11\}$, then the corresponding population vector is $[\frac{1}{4}, \frac{1}{2}, 0, \frac{1}{4}]^T$. A population vector is a population-size independent representation of a population, and thus is natural for infinite-population models.

If a population x depends on time (or GA generation) t, then $x(t)$ is the population at time t.

All population vectors are contained in the simplex $\Lambda = \{x : \sum_{j \in \Omega} x_j = 1 \text{ and } x_j \geq 0 \text{ for all } j\}$. The simplex is a natural setting for the dynamical systems model since it allows population vectors to range continuously over a subset of $\mathbb{R}^{2^\ell}$ and thus allows derivatives and calculus to be used.

The *Walsh matrix* W is an 2^ℓ by 2^ℓ matrix defined by $W_{i,j} = 2^{-\ell/2}(-1)^{\#(i \otimes j)}$ The Walsh matrix is symmetric and $W = W^{-1}$. For example, the Walsh matrix for $\ell = 2$ is:

$$W = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

If x is a population vector, then the Walsh transform of x is Wx and is denoted by $\widehat{x}$. Let $e_0, e_1, \ldots, e_{N-1}$ be the standard basis vectors for $\mathbb{R}^N$. Then the vectors $\widehat{e}_0, \widehat{e}_1, \ldots, \widehat{e}_{N-1}$ form the Walsh basis for $\mathbb{R}^N$. If x is a vector then $\widehat{x} = Wx$ is expressed in the Walsh basis. In other words, x_j is the jth coordinate of x in the standard basis and $\widehat{x}_j$ is the jth coordinate of x in the Walsh basis. If A is a 2^ℓ by 2^ℓ matrix, then WAW is the Walsh trans-

form of A and is denoted by $\widehat{A}$.

3 Linkage Equilibrium

This section gives some basic results relating schema averages, the Walsh representation of a population, and properties of a population that is at linkage equilibrium.

These results will show how string and schema frequencies can be determined from the Walsh coefficients that we use to describe our models.

If x is a population vector and $u \in \mathcal{L}$, let

$$x_v^{(u)} = \sum_{i \in \Omega} x_i \delta_{i \otimes u, v}$$

where $\delta_{j,k} = 1$ iff $j = k$. In other words, $x_v^{(u)}$ denotes the proportion of individuals i in the population such that the position of i corresponding to u is the same as the corresponding position of v. For example, if $x = \frac{1}{64}[19,\ 5,\ 29,\ 11]^T$, then $x_{00}^{(01)} = 48/64$ and $x_{01}^{(01)} = 16/64$.

Definition 1 *A population x is in* **linkage equilibrium** *if*

$$x_k = \prod_{u \in \mathcal{L}} x_{u \otimes k}^{(u)}$$

Thus, a population is in linkage equilibrium if the frequency of each string is the product of the corresponding 1 schema averages. For example, the population $\frac{1}{64}[19,\ 5,\ 29,\ 11]^T$ (with Walsh basis representation $\frac{1}{32}[16,\ 8,\ -4,\ -1]^T$) is in linkage equilibrium.

The following relates the schema sums of the order 1 schemata and the order 1 Walsh coefficients. It follows from the formula for the Walsh coefficients.

Lemma 2 *If $u \in \mathcal{L}$,*

$$x_v^{(u)} = \frac{1}{2}\left(1 + (-1)^{\#v} 2^{\ell/2} \widehat{x}_u\right)$$

The frequency of any string can be found from the Walsh basis representation of the population.

Corollary 3 *Let x be in linkage equilibrium. For any $v \in \Omega$,*

$$x_v = 2^{-\ell} \prod_{i \in \mathcal{L}} \left(1 + (-1)^{\#(i \otimes v)} 2^{\ell/2} \widehat{x}_i\right)$$

The following theorem gives the relationship of linkage equilibrium to the Walsh basis representation of a population. The result follows easily from theorem 10.9 of [9].

Theorem 4 *If population x is in linkage equilibrium, then*

$$\widehat{x}_k = 2^{(\#k-1)\ell/2} \prod_{i \in \mathcal{L}_k} \widehat{x}_i$$

Remark: While higher-order schemata will not be used in this paper, it is interesting to note that the higher order schemata are products of order 1 schemata for a population at linkage equilibrium. Thus, in traditional notation, the frequency of the schema $0*1*$ is the product of the frequencies of $0***$ and $**1*$.

Theorem 5 *If $x \in \Lambda$, then $|\widehat{x}_k| \leq 2^{-\ell/2}$ for all $k \in \Omega$.*

4 The Gene Pool Model in the Walsh Basis

In this paper, one generation of the gene pool GA will consist of the following three steps:

1. Gene pool recombination.
2. Proportional selection.
3. Mutation.

The dynamical system model will be described by a function $\mathcal{G} : \Lambda \longrightarrow \Lambda$. In other words, if $x(t)$ is the population at generation t, then the population at generation $t+1$ is given by $x(t+1) = \mathcal{G}(x(t))$.

$\mathcal{G}$ is a composition of $\mathcal{M}$, $\mathcal{F}$, and $\mathcal{U}$ (i. e., $\mathcal{G} = \mathcal{U} \circ \mathcal{F} \circ \mathcal{M}$). The $\mathcal{M}$, $\mathcal{F}$, and $\mathcal{U}$ functions are described below.

The next three subsections tell how to compute $\mathcal{F}$, $\mathcal{U}$, and $\mathcal{M}$ in the Walsh basis.

4.1 Proportional selection

Following [9], the effect of proportional selection can be described by a function $\mathcal{F} : \Lambda \longrightarrow \Lambda$. The probability that an individual $k \in \Omega$ is chosen to be in the new population is $\mathcal{F}(x)_k$. Or stated another way, if proportional selection is applied to a population x, then the expected frequency vector for the resulting population is $\mathcal{F}(x)$.

If $f \in \mathbb{R}^{2^\ell}$ is the fitness function (i. e., f_k is the fitness of k) then

$$\mathcal{F}(x) = \frac{Fx}{f^T x}$$

where F is the diagonal matrix such that $F_{k,k} = f_k$.

The average fitness $f^T x$ when computed in the Walsh basis is the same: $\widehat{f^T}\widehat{x} = (f^T W)(Wx) = f^T x$. Thus,

$$\widehat{\mathcal{F}(x)} = \frac{WFx}{\widehat{f}^T\widehat{x}} = \frac{WFWWx}{\widehat{f}^T\widehat{x}} = \frac{\widehat{F}\widehat{x}}{\widehat{f}^T\widehat{x}}$$

Lemma 6 *For any $i, j \in \Omega$,*

$$\widehat{F}_{i,j} = 2^{-\ell/2} \widehat{f}_{i \oplus j}$$

Let σ_k denote the $2^\ell \times 2^\ell$ matrix defined by $(\sigma_k)_{i,j} = \delta_{i\oplus k,j}$. Then it is easy to show that $(\sigma_k x)_i = x_{i\oplus k}$.

The following corollary tells how to compute proportional selection in the Walsh basis.

Corollary 7

$$\widehat{F(x)}_k = \frac{\sum_{i\in\Omega} \widehat{f}_{i\oplus k}\widehat{x}_i}{2^{\ell/2}\widehat{f}^T\widehat{x}} = \frac{\sum_{i\in\Omega} \widehat{f}_i\widehat{x}_{i\oplus k}}{2^{\ell/2}\widehat{f}^T\widehat{x}} = \frac{\widehat{f}^T\sigma_k\widehat{x}}{2^{\ell/2}\widehat{f}^T\widehat{x}}$$

4.2 Mutation

The effect of mutation can be described by a function $\mathcal{U} : \Lambda \longrightarrow \Lambda$. The probability that an individual $k \in \Omega$ is chosen to be in the new population is $\mathcal{U}(x)_k$.

If the probability that a bit is mutated (flipped) is μ, then the probability that an individual $i \in \Omega$ is mutated to an individual j is $\mu^{\#(i\oplus j)}(1-\mu)^{\ell-\#(i\oplus j)}$. (Note that $\#(i\oplus j)$ is the Hamming distance from i to j.) Thus, we can define a $2^\ell \times 2^\ell$ mutation matrix U with $U_{i,j} = \mu^{\#(i\oplus j)}(1-\mu)^{\ell-\#(i\oplus j)}$, and $\mathcal{U}(x) = Ux$.

Following Vose (section 4.3 of [9]), we define a vector μ indexed over Ω by

$$\mu_i = (\mu)^{\#i}(1-\mu)^{\ell-\#i}$$

(Whether μ denotes the scalar mutation rate or the mutation vector should be clear from the context.) The following lemma is proved in [9].

Lemma 8

$$\widehat{\mu}_k = 2^{-\ell/2}(1-2\mu)^{\#k}$$

The Walsh transform of U is easily shown to be diagonal.

Lemma 9 *$\widehat{U}$ is a diagonal matrix which diagonal entries given by:*

$$\widehat{U}_{i,i} = (1-2\mu)^{\#i}$$

Theorem 10

$$\widehat{\mathcal{U}(x)}_k = (\widehat{U}\widehat{x})_k = (1-2\mu)^{\#k}\widehat{x}_k$$

Thus, mutation is very simple in the Walsh basis: it corresponds to multiplication by a diagonal matrix.

4.3 Gene pool recombination

To implement gene pool recombination in a finite population GA, the order 1 schema sums $x^{(u)}$ for $\#u = 1$ are computed from the current population x. Then each individual of the new population is constructed by choosing each bit according the probabilities determined by these schema sums. Thus, the probability that the bit at position i is 0 is $x_0^{(2^i)}$ and the probability that the bit at position i is 1 is $x_{2^i}^{(2^i)}$.

In the infinite population model, the population y resulting from applying gene pool recombination to population x has the property that $\widehat{y}_u = \widehat{x}_u$ for $\#u = 1$. In addition, y is at linkage equilibrium. Thus, the Walsh coefficients of the population after gene pool recombination can be computed from theorem 4.

As with proportional selection and mutation, we can describe gene pool recombination through a function $\mathcal{M} : \Lambda \longrightarrow \Lambda$. By theorem 4, we have:

$$\widehat{\mathcal{M}(x)}_k = 2^{(\#k-1)\ell/2}\prod_{i\in\mathcal{L}_k}\widehat{x}_i = 2^{-\ell/2}\prod_{i\in\mathcal{L}_k}2^{\ell/2}\widehat{x}_i$$

This formula tells how to compute gene pool recombination in the Walsh basis. Note that if $k \in \mathcal{L}$ (i. e., $\#k = 1$), then $\widehat{M(x)}_k = \widehat{x}_k$. In other words, $\mathcal{M}$ is the identity on the order-1 Walsh coefficients.

5 Linear fitness and the ONEMAX problem

Definition 11 *A fitness function represented by a fitness vector f is* **linear** *if*

$$f_i = c + \sum_{j\in\mathcal{L}} b_j\delta_{i\otimes j,j}$$

where c and b_j, $j \in \mathcal{L}$ are constants. (Note that if $j = 2^k \in \mathcal{L}$, then $i \otimes j = j$ if and only if bit k of i is 1.)

Without loss of generality, we can assume that $b_j \geq 0$ for $j \in \mathcal{L}$. If some $b_j < 0$ where $j = 2^p$, this says that a string with 0 at position p is more fit than the same string with a 1 at that position. Thus, a change of representation where 0 and 1 are interchanged in all strings at that position will make $b_j > 0$.

The ONEMAX fitness function is the linear fitness function where $c = 0$ and $b_j = 1$ for all $j \in \mathcal{L}$.

The following lemma shows that when a linear fitness is expressed in the Walsh basis, only the order 0 and order 1 coefficients are nonzero.

Lemma 12 *For any $k \in \Omega$,*

$$\widehat{f}_k = \begin{cases} 2^{\ell/2}c + 2^{\ell/2-1}\sum_{j\in\mathcal{L}} b_j & \text{if } k = 0 \\ -2^{\ell/2-1}b_k & \text{if } k \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

If $c \geq 0$ and $b_j \geq 0$ for $j \in \mathcal{L}$, then $\widehat{f}_j \leq 0$ and $\widehat{f}_0 \geq -\sum_{j\in\mathcal{L}}\widehat{f}_j$.

Since we are working in the Walsh basis, it will convenient to simply characterize a linear fitness in terms of its Walsh coefficients. So in this section, we assume that $\widehat{f}_k = 0$ for $k \notin \mathcal{L}\cup\{0\}$, $\widehat{f}_j \leq 0$ for $j \in \mathcal{L}$ and $\widehat{f}_0 + \sum_{j\in\mathcal{L}}\widehat{f}_j \geq 0$.

These correspond to the assumptions that fitness is linear, $b_j \geq 0$, and $c \geq 0$.

We can now compute selection in the Walsh basis. For $k \in \mathcal{L}$,

$$\begin{aligned}\widehat{\mathcal{F}(y)}_k &= \frac{\widehat{f}^T \sigma_k \widehat{y}}{2^{\ell/2} \widehat{f}^T \widehat{y}} \\ &= \frac{\sum_{i\in\Omega} \widehat{f}_i \widehat{y}_{i\oplus k}}{2^{\ell/2} \sum_{i\in\Omega} \widehat{f}_i \widehat{x}_i} \\ &= \frac{2^{-\ell/2} \widehat{f}_k + \widehat{f}_0 \widehat{y}_k + 2^{\ell/2} \sum_{j\in\mathcal{L}\setminus\{k\}} \widehat{f}_j \widehat{y}_{j\oplus k}}{\widehat{f}_0 + 2^{\ell/2} \sum_{j\in\mathcal{L}} \widehat{f}_j \widehat{y}_j}\end{aligned}$$

Now assume that $y = \mathcal{M}(x)$ where $\mathcal{M}$ denotes gene pool recombination. Then $\widehat{y}_k = \widehat{x}_k$ and $\widehat{y}_{j\oplus k} = 2^{-\ell/2} \widehat{x}_j \widehat{x}_k$ for for $j, k \in \mathcal{L}$ by theorem 4.

For $k \in \mathcal{L}$,

$$\begin{aligned}\widehat{\mathcal{G}(x)}_k &= \widehat{e}_k^T \mathcal{U}(\mathcal{F}(\mathcal{M}(x)))_k \\ &= (1-2\mu) \frac{2^{-\ell/2} \widehat{f}_k + \widehat{f}_0 \widehat{x}_k + 2^{\ell/2} \widehat{x}_k \sum_{j\in\mathcal{L}\setminus\{k\}} \widehat{f}_j \widehat{x}_j}{\widehat{f}_0 + 2^{\ell/2} \sum_{j\in\mathcal{L}} \widehat{f}_j \widehat{x}_j}\end{aligned}$$

This formula is a recurrence that defines the gene pool model for linear fitness in terms of the variables $\widehat{x}_k$ for $k \in \mathcal{L}$.

Notation is simplified by a variable substitution. Let $\widehat{z}_k = 2^{\ell/2} \widehat{x}_k$. Then if z is in the simplex, lemma 5 shows that $-1 \leq \widehat{z} \leq 1$. The recurrence written in terms of the $\widehat{z}_k$ is:

$$\begin{aligned}\widehat{\mathcal{G}(z)}_k &= (1-2\mu) \frac{\widehat{f}_k + \widehat{f}_0 \widehat{z}_k + \widehat{z}_k \sum_{j\in\mathcal{L}\setminus\{k\}} \widehat{f}_j \widehat{z}_j}{\widehat{f}_0 + \sum_{j\in\mathcal{L}} \widehat{f}_j \widehat{z}_j} \\ &= (1-2\mu)\widehat{z}_k + (1-2\mu) \frac{\widehat{f}_k (1 - \widehat{z}_k^2)}{\widehat{f}_0 + \sum_{j\in\mathcal{L}} \widehat{f}_j \widehat{z}_j} \qquad (1)\end{aligned}$$

Later we will want to be able to evaluate the average fitness in terms of the $\widehat{z}_j$. The average fitness is

$$\begin{aligned}f^T x = \widehat{f}^T \widehat{x} &= 2^{-\ell/2} \widehat{f}_0 + \sum \widehat{f}_j \widehat{x}_j \\ &= 2^{-\ell/2} \widehat{f}_0 + \sum \widehat{f}_j 2^{-\ell/2} \widehat{z}_j \\ &= 2^{-\ell/2} (\widehat{f}_0 + \sum \widehat{f}_j \widehat{z}_j)\end{aligned}$$

The fixed point equations are:

$$\widehat{f}_k \widehat{z}_k^2 + \frac{2\mu}{1-2\mu} \left(\widehat{f}_0 + \sum_{j\in\mathcal{L}} \widehat{f}_j \widehat{z}_j \right) \widehat{z}_k - \widehat{f}_k = 0 \qquad (2)$$

Lemma 13 *If fitness is linear and $0 < \mu < 1/2$, then the gene pool GA has a unique fixed point in the simplex.*

Theorem 14 *For linear fitness and $0 < \mu < 1/2$, the gene pool GA model has a unique fixed point, and this fixed point is asymptotically stable.*

5.1 The ONEMAX problem

For a rescaled ONEMAX fitness function, we can take $\widehat{f}_0 = \ell$ and $\widehat{f}_j = -1$ for all $j \in \mathcal{L}$. Rescaling has no effect on the $\mathcal{G}$ function or the fixed points, but it does affect the total fitness. From lemma 12, this original ONEMAX fitness is $2^{\ell/2-1}$ times the rescaled ONEMAX fitness. From an earlier remark, the average rescaled fitness is $2^{-\ell/2}(\widehat{f}_0 + \sum \widehat{f}_j \widehat{z}_j) = 2^{-\ell/2}(\ell - \sum \widehat{z}_j)$. Thus, the average ONEMAX fitness is $\frac{1}{2}(\widehat{f}_0 + \sum \widehat{f}_j \widehat{z}_j) = \frac{1}{2}(\ell - \sum \widehat{z}_j)$.

If we assume a symmetric population, i. e., if we assume that $\widehat{z}_j = w$ for all j, then the recurrence simplifies to:

$$\begin{aligned}\mathcal{G}(w(t)) = w(t+1) &= (1-2\mu) \frac{(\ell-1)w(t) - 1}{\ell} \\ &= \frac{(1-2\mu)(\ell-1)}{\ell} w(t) - \frac{1-2\mu}{\ell}\end{aligned}$$

For $\mu = 0$, this recurrence is equivalent to one given in [4].

This linear recurrence can easily be solved. Let $A = \frac{(1-2\mu)(\ell-1)}{\ell}$, $C = \frac{1-2\mu}{\ell}$.

$$\mathcal{G}^t(w) = w(t) = A^t w(0) - C \left(\frac{A^t - 1}{A - 1} \right)$$

The fixed point is:

$$w_{fixed} = \frac{-C}{A-1} = -\frac{1-2\mu}{2\mu(\ell-1)+1}$$

We can evaluate the average fitness for the original ONEMAX fitness at this fixed point. From above, the average ONEMAX fitness is $\frac{1}{2}(\ell - \sum \widehat{z}_j) = \frac{1}{2}(\ell - \ell w) = \frac{\ell}{2}(1-w)$. At the fixed point $w = w_{fixed}$, this evaluates to $\frac{\ell(1+\mu\ell-2\mu)}{1+2\mu\ell-2\mu}$. If the mutation rate is $\mu = \frac{1}{\ell}$, this is $\frac{\ell(2-2/\ell)}{3-2/\ell} \approx \frac{2\ell}{3}$.

Theorem 15 *Let $\epsilon > 0$ and let $\alpha = \ell\mu$ so that $\mu = \alpha/\ell$. If the gene pool GA model is started on the ONEMAX fitness function from a random population ($w(0) = 0$), and if t generations are done where t is chosen so that*

$$t > \frac{-\ell \ln \epsilon}{1 + 2\alpha}$$

then

$$\frac{|\mathcal{G}^t(0) - w_{fixed}|}{|w_{fixed}|} < \epsilon$$

In other words, the relative error after t generations is at most ϵ.

The theorem says that for a fixed mutation rate, the number of generations is $\Theta(\ell)$. Section 13.2 of [9] says that $\mathcal{G}$ for general fitness is logarithmically convergent. Theorem 15 agrees with this result, only it gives explicit values for the constants for the ONEMAX fitness function.

The theorem also says that the time to convergence goes down as the mutation rate increases. However, as the mutation rate increases, the fixed point moves closer to the middle of the simplex, so the algorithm has less far to go. When the mutation rate is $1/2$, the fixed point is at the center of the simplex, and the algorithm starts at the fixed point.

5.2 Empirical comparisons

The gene pool GA gives a good approximation to a two-parent GA for linear fitness functions. Figure 5.2 show the average fitness and figure 5.2 shows the number of optimal individuals for 99 generations with gene pool, uniform, and one-point recombination, where a crossover rate of 1 is used for uniform and one-point recombination. The string length was 18, and the mutation rate was 0.01. The graph shows an average of 10 runs, each with a population size of 500000. The errors are negligible.

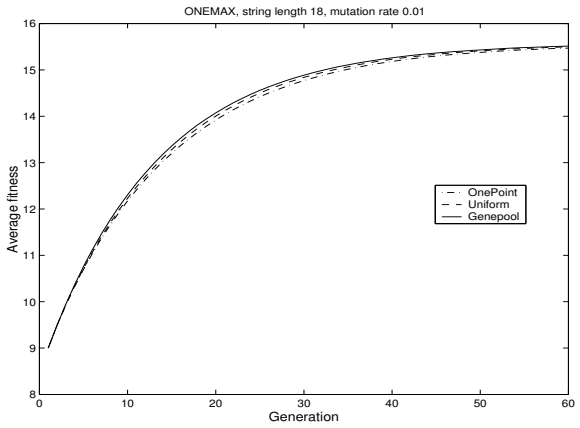


Figure 1: The average fitness for different types of recombination

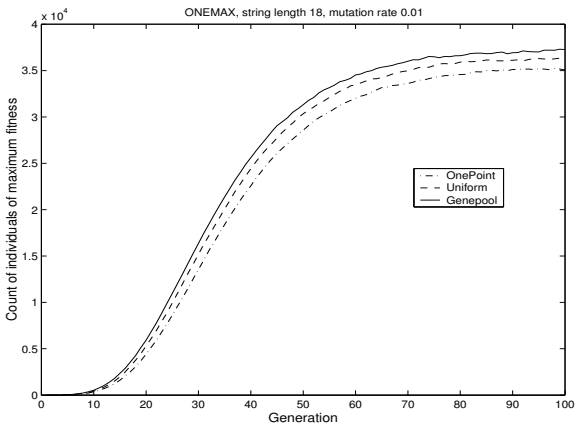


Figure 2: The number of optimal individuals for different types of recombination

6 Discussion and Conclusion

In the paper we have given an exact infinite population model of a selection/mutation/recombination genetic algorithm that is tractable for large string lengths. We have shown that for linear fitness functions, there is a unique fixed point, and this fixed point is asymptotically stable in the space of all populations. For the ONEMAX problem, the model reduces to a single variable, and an explicit solution to the recurrence equation is given for symmetric populations. The fixed point equation is a single quadratic equation in one variable, so an explicit formula for the fixed point is given. The time to convergence is shown to be $O(\ell)$ generations, where ℓ is the string length.

This paper should be the first step in the unification of the Walsh basis analysis of crossover/mutation given in [9] with the Walsh basis analysis of fitness functions.

7 Appendix

First is a technical lemma.

Lemma 16

$$\sum_{j\in\Omega_u}(-1)^{\#(j\otimes w)} = \begin{cases} 2^{\#u} & \textit{if } w\in\Omega_{\overline{u}} \\ 0 & \textit{otherwise} \end{cases}$$

Proof. Suppose that $w \in \Omega_{\overline{u}}$. This implies that $j \otimes w = 0$ for all $j \in \Omega_u$. Thus, the summation is equal to the number of elements in Ω_u, which is $2^{\#u}$.

Now suppose $w \notin \Omega_{\overline{u}}$. The we can write $w = q \oplus v$ where $\#q = 1$, $q \in \Omega_u$, and $q \otimes v = 0$. Note that $\Omega_q = \Omega_{u\otimes q} = \{0, q\}$. Then

$$\begin{aligned}
&\sum_{j\in\Omega_u}(-1)^{\#(j\otimes w)} \\
&= \sum_{i\in\Omega_{u\otimes q}}\sum_{k\in\Omega_{u\otimes\overline{q}}}(-1)^{\#((i\oplus k)\otimes(q\oplus v))} &(3)\\
&= \sum_{i\in\Omega_{u\otimes q}}(-1)^{\#(i\otimes q)}\sum_{k\in\Omega_{u\otimes\overline{q}}}(-1)^{\#(k\otimes v)} &(4)\\
&= \sum_{k\in\Omega_{u\otimes\overline{q}}}(-1)^{\#(k\otimes v)} - \sum_{k\in\Omega_{u\otimes\overline{q}}}(-1)^{\#(k\otimes v)} = 0 &(5)
\end{aligned}$$

Equation (4) follows from (3) since $(i \oplus k) \otimes (q \oplus v) = (i\otimes q)\oplus(k\otimes v)$, and since $\#((i\otimes q)\oplus(k\otimes v))$ (mod 2) $= \#(i \otimes q) + \#(k \otimes v)$ (mod 2) . Equation (5) follows from (4) since $\Omega_q = \Omega_{u\otimes q} = \{0, q\}$. □

Proof of lemma 5: The simplex Λ is the convex hull of the basis vectors $e_0, e_1, \ldots, e_{N-1}$ of the standard basis. The vectors $\widehat{e}_0, \widehat{e}_1, \ldots, \widehat{e}_{N-1}$ are the same geometric points expressed in the Walsh basis, so the simplex is still the convex hull of these points. But these correspond to the columns of the Walsh matrix, and every entry of the Walsh matrix

is $\pm 2^{-\ell/2}$. Thus $\widehat{x}_k$ is a convex combination of $2^{-\ell/2}$ and $-2^{-\ell/2}$. □

Proof of lemma 6:

$$\begin{aligned}\widehat{F}_{i,j} &= 2^{-\ell}\sum_{u\in\Omega}(-1)^{\#(u\otimes i)}\sum_{v\in\Omega}(-1)^{\#(v\otimes j)}F_{u,v}\\ &= 2^{-\ell}\sum_{u\in\Omega}(-1)^{\#(u\otimes i)}\sum_{v\in\Omega}(-1)^{\#(v\otimes j)}\delta_{u,v}F_{u,u}\\ &= 2^{-\ell}\sum_{u\in\Omega}(-1)^{\#(u\otimes i)}(-1)^{\#(u\otimes j)}f_u\\ &= 2^{-\ell}\sum_{u\in\Omega}(-1)^{\#(u\otimes (i\oplus j)}f_u\\ &= 2^{-\ell/2}\widehat{f}_{i\oplus j}\end{aligned}$$

□

Proof of lemma 12:

$$\begin{aligned}\widehat{f}_k &= 2^{-\ell/2}\sum_{i\in\Omega}(-1)^{\#(k\otimes i)}f_i\\ &= 2^{-\ell/2}c\sum_{i\in\Omega}(-1)^{\#(k\otimes i)}\\ &\quad + 2^{-\ell/2}\sum_{j\in\mathcal{L}}b_j\sum_{i\in\Omega}(-1)^{\#(k\otimes i)}\delta_{i\otimes j,j}\\ &= 2^{\ell/2}c\delta_{k,0} + 2^{-\ell/2}\sum_{j\in\mathcal{L}}b_j\\ &\quad \sum_{v\in\Omega_j}(-1)^{\#(k\otimes v)}\sum_{u\in\Omega_{\overline{j}}}(-1)^{\#(k\otimes u)}\delta_{(v\oplus u)\otimes j,j}\end{aligned}$$

Note that $(v\oplus u)\otimes j = v\otimes j$ and $\delta_{v\otimes j,j}=1$ iff $v=j$. Thus,

$$\begin{aligned}\widehat{f}_k &= 2^{\ell/2}c\delta_{k,0} + 2^{-\ell/2}\sum_{j\in\mathcal{L}}b_j(-1)^{\#(k\otimes j)}\sum_{u\in\Omega_{\overline{j}}}(-1)^{\#(k\otimes u)}\\ &= 2^{\ell/2-1}\delta_{k,0}(2c+\sum_{j\in\mathcal{L}}b_j) - 2^{\ell/2-1}\sum_{j\in\mathcal{L}}\delta_{k,j}b_j\end{aligned}$$

The last statement of the lemma follows easily from the formula. □

Proof of Lemma 13: Let $\widehat{z}$ be a fixed point in the simplex.

In the special case where $\widehat{f}_k = 0$, $\widehat{z}_k = 0$, so throughout the rest of the proof we assume that $\widehat{f}_k < 0$.

Let $B = \frac{\mu}{1-2\mu}(\widehat{f}_0 + \sum_{j\in\mathcal{L}}\widehat{f}_j\widehat{z}_j)$. Solving equation (2) for $\widehat{z}_k$ gives:

$$\widehat{z}_k = \frac{1}{\widehat{f}_k}\left(-B \pm \sqrt{B^2+\widehat{f}_k^2}\right) \tag{6}$$

We claim that for a solution to be in the simplex, the plus sign must be used in equation (6). So assume that $\widehat{z}$ represents a solution in the simplex.

First we claim that $B > 0$. Recall that we assumed that $\widehat{f}_0 + \sum_{j\in\mathcal{L}}\widehat{f}_j \geq 0$ and $\widehat{f}_j < 0$ for all j. Since $\widehat{z}$ is in the simplex, and since mutation is positive, $\widehat{z}_j < 1$ for all $j\in\mathcal{L}$ by lemma 5. It follows easily that $B > 0$.

Now we assume that the minus sign is used in equation (6) and derive a contradiction. Thus,

$$\begin{aligned}\widehat{z}_k < 1 &\Longrightarrow \frac{1}{\widehat{f}_k}\left(-B-\sqrt{B^2+\widehat{f}_k^2}\right) < 1\\ &\Longrightarrow B+\sqrt{B^2+\widehat{f}_k^2} < -\widehat{f}_k\\ &\Longrightarrow \sqrt{B^2+\widehat{f}_k^2} < -\widehat{f}_k - B\\ &\Longrightarrow B^2+\widehat{f}_k^2 < \widehat{f}_k^2 + 2\widehat{f}_k B + B^2\\ &\Longrightarrow 0 < 2\widehat{f}_k B\end{aligned}$$

Since $\widehat{f}_k < 0$ and $B > 0$, this is a contradiction.

Thus, the plus sign must always be used in equation (6), and there is a unique fixed point. □

To prove theorem 14 we will use the Gershgorin Circle Theorem (page 685 of [2]).

Theorem 17 *Let J by an n by n real-valued matrix, and let λ be an eigenvalue of J. For some integer k,*

$$|J_{k,k}-\lambda| \leq \sum_{j\neq k}|J_{k,j}|$$

Proof of theorem 14: The existence and uniqueness of the fixed point was proved in lemma 13.

We calculate the differential of $\mathcal{G}$ from equation (1).

$$\frac{\partial\mathcal{G}(\widehat{z})_k}{\partial\widehat{z}_i} = (1-2\mu)\frac{\widehat{f}_k\widehat{f}_i\left(\widehat{z}_k^2-1\right)}{\left(\widehat{f}_0+\sum_j\widehat{f}_j\widehat{z}_j\right)^2}$$

$$\frac{\partial\mathcal{G}(\widehat{z})_k}{\partial\widehat{z}_k} = (1-2\mu)\frac{\left(\widehat{f}_0+\sum_{j\in\mathcal{L}\setminus\{k\}}\widehat{f}_j\widehat{z}_j\right)^2-\widehat{f}_k^2}{\left(\widehat{f}_0+\sum_j\widehat{f}_j\widehat{z}_j\right)^2} \tag{7}$$

Equation (2) can be rewritten in the form:

$$2\mu\widehat{z}_k\left(\widehat{f}_0+\sum_{j\in\mathcal{L}}\widehat{f}_j\widehat{z}_j\right)+(1-2\mu)\widehat{f}_k\left(\widehat{z}_k^2-1\right)=0 \tag{8}$$

We claim that for a solution $\widehat{z}$ of these equations, $\widehat{z}_k \leq 0$ for all k. This follows from the equations since we have assumed that $\widehat{f}_k \leq 0$ and $\widehat{f}_0+\sum_{j\in\mathcal{L}}\widehat{f}_j\widehat{z}_j \geq 0$, and we know that $|\widehat{z}_k| \leq 1$ since $\widehat{z}$ corresponds to a point in the simplex.

Assume that $\widehat{z}$ is a fixed point. Then using equation (8):

$$\frac{\partial \mathcal{G}(\widehat{z})_k}{\partial \widehat{z}_i} = \frac{-2\mu \widehat{f}_i \widehat{z}_k \left(\widehat{f}_0 + \sum_j \widehat{f}_j \widehat{z}_j\right)}{\left(\widehat{f}_0 + \sum_j \widehat{f}_j \widehat{z}_j\right)^2} = \frac{-2\mu \widehat{f}_i \widehat{z}_k}{\widehat{f}_0 + \sum_j \widehat{f}_j \widehat{z}_j}$$

We assumed earlier that $-\sum_j \widehat{f}_j \leq \widehat{f}_0$. Since $\widehat{z}$ corresponds to point in the simplex, $\widehat{z}_k \geq -1$, or $-\widehat{z}_i \leq 1$. Thus,

$$\begin{aligned}\sum_{i \in \mathcal{L} \setminus \{k\}} \left| \frac{\partial \mathcal{G}(\widehat{z})_k}{\partial \widehat{z}_i} \right| &= \frac{2\mu(-\widehat{z}_k)}{\widehat{f}_0 + \sum_j \widehat{f}_j \widehat{z}_j} \sum_{i \in \mathcal{L} \setminus \{k\}} (-\widehat{f}_i) \\ &\leq \frac{2\mu}{\widehat{f}_0 + \sum_j \widehat{f}_j \widehat{z}_j} \widehat{f}_0 < 2\mu\end{aligned}$$

Using equation (7),

$$\frac{\partial \mathcal{G}(\widehat{z})_k}{\partial \widehat{z}_k} \leq (1 - 2\mu)$$

The Gershgorin Circle Theorem shows that any eigenvalue of the differential is less that 1. □

Proof of Theorem 15: The relative error after t generations is:

$$\frac{|\mathcal{G}^t(0) - w_{fixed}|}{|w_{fixed}|} = \frac{\left| -\frac{C(A^t - 1)}{A - 1} - \frac{C}{A-1} \right|}{\frac{C}{A-1}} = A^t$$

Further,

$$A^t < \epsilon \Longleftrightarrow t \ln A < \ln \epsilon \Longleftrightarrow t > \frac{-\ln \epsilon}{-\ln A}$$

Let $x = 1/\ell$ and let

$$\begin{aligned}f(x) = -\ln A &= -\ln\left(1 - \frac{2\alpha + 1}{\ell} + \frac{2\alpha}{\ell}\right) \\ &= -\ln(1 - (2\alpha + 1)x + 2\alpha x^2)\end{aligned}$$

The first-order Taylor series with remainder for f about $x = 0$ is

$$f(x) = (2\alpha+1)x + \frac{-8\alpha^2\xi - 4\alpha\xi + 8\alpha^2\xi^2 + 4\alpha^2 + 1}{\left(1 - (2\alpha + 1)\,\xi + 2\alpha\xi^2\right)^2} \quad \frac{x^2}{2}$$

where $0 < \xi < x$.

We claim that $B = -8\alpha^2\xi - 4\alpha\xi + 8\alpha^2\xi^2 + 4\alpha^2 + 1 \geq 0$. Make the substitution $\xi = \psi + \frac{2\alpha+1}{4\alpha}$. Then the expression becomes $8\alpha^2\psi^2 + \frac{1}{2}(4\alpha^2 - 4\alpha + 1)$ which is clearly positive.

Thus, $f(x) = (2\alpha + 1)x + B \geq (2\alpha + 1)x$, and

$$\frac{-\ell \ln \epsilon}{1 + 2\alpha} \geq \frac{-\ln \epsilon}{(1 - 2\alpha)\left(\frac{1}{\ell}\right) + B\left(\frac{1}{\ell}\right)^2} = \frac{-\ln \epsilon}{f\left(\frac{1}{\ell}\right)} = \frac{-\ln \epsilon}{-\ln A}$$

□

References

[1] H. Geiringer. On the probability of linkage in mendelian heredity. *Annals of Mathematical Statistics*, 15:25–57, 1944.

[2] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley and Sons, New York, third edition, 1972.

[3] Thilo Mahnig and Heinz Mühlenbein. Optimal mutation rate using bayesian priors for estimation of distribution algorithms. In K. Steinhüfel, editor, *Stochastic Algorithms: Foundations and Applications*, LNCS. Springer-Verlag, 2001.

[4] Heinz Mühlenbein. The equation for the response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1998.

[5] Heinz Mühlenbein and Thilo Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.

[6] Heinz Mühlenbein and Thilo Mahnig. Evolutionary algorithms: from recombination to search distributions. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computation*, pages 137–176. Springer Verlag, 2000.

[7] C. R. Stephens and H. Waelbroeck. Effective degrees of freedom in genetic algorithms and the block hypothesis. In Thomas Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 34–40, San Mateo, 1997. Morgan Kaufman.

[8] H. Voigt and H. Mühlenbein. Gene pool recombination and the utilization of covariances for the breeder genetic algorithm. In Z. Michalewicz, editor, *Proc. of the 2nd IEEE International Conference on Evolutionary Computation*, pages 172– 177, New York, 1995. IEEE Press.

[9] M. D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1999.

[10] M. D. Vose and A. H. Wright. Form invariance and implicit parallelism. *Evolutionary Computation*, 2001.

Adaptive Non-Uniform Crossover Based on Statistics for Genetic Algorithms

Shengxiang Yang

Department of Mathematics and Computer Science
University of Leicester
University Road, Leicester LE1 7RH, UK
Email: s.yang@mcs.le.ac.uk

Abstract

The genetic algorithm (GA) is a meta-heuristic search algorithm based on mechanisms abstracted from population genetics. Through the population, the GA implicitly maintains the statistics about the search space. This implicit statistics can be used explicitly to enhance GA's performance. In this paper, a new statistics-based adaptive non-uniform crossover (SANUX) is proposed. SANUX uses the statistics information of the allele distribution in each locus to adaptively adjust the crossover operation. Our preliminary experiment results show that SANUX is more efficient than traditional one-point, two-point, and uniform crossover across a representative set of search problems.

1 INSTRUCTIONS

Based on Holland's simple genetic algorithm (Holland, 1975), there have been many variations developed both in GA's macro-structure and micro-structure (Goldberg, 1989). The GA, as one kind of generation-based evolutionary algorithm, maintains a population of candidate solutions to a given problem, which are evaluated according to a problem-specific fitness function that defines the environment for the evolution. New population is created by selecting relatively fit members of the present population and evolving them through recombination and mutation operations. The performance of a GA is dependent on many factors, such as encoding scheme, selection method, population size, crossover and mutation operators. This makes it difficult, if not impossible, to choose operators for optimal performance. In this paper we focus on the recombination operator.

In most GAs, recombination operators act on a pair of individuals (parents) to produce new offsprings (children) by exchanging segments of the parents' corresponding genetic material. This kind of two-parent recombination is usually called "crossover". The number of crossover points determines how many segments are exchanged. Traditionally, GAs have used the one-point crossover and two-point crossover that choose crossing points uniformly at random. This decision was supported theoretically and empirically by early work of GA's researchers (De Jong, 1975; Holland, 1975). However, researchers have also carried out experiments with multi-point crossover: the n-point crossover (Eshelman, 1989) which exchanges $n-1$ segments of the two parents and the uniform crossover (Reed, Toombs and Barricelli, 1967; Syswerda, 1989) which generates offsprings by swapping each bit of the parents with a fixed probability $p = 0.5$. Spears and De Jong (1991) proposed that the swapping probability under uniform crossover could be fixed to values other than 0.5. Recently, researchers have applied adaptation techniques to recombination to enhance GA's capabilities and have made the adaptation of recombination operators and/or their parameters one of the most promising research areas in GAs.

As a search algorithm based on mechanisms abstracted from population genetics, the GA implicitly maintains the statistics about the search space via the population. It uses the selection, crossover and mutation to explicitly extract the implicit statistics from the population to reach the next set of points in the search space. This implicit statistics can be used explicitly to enhance GA's performance. In this paper, a new statistics-based adaptive non-uniform crossover operator, called SANUX, is proposed. SANUX explicitly uses the statistics information of the distribution of alleles in each gene locus over the population to adaptively guide the process of crossover. With SANUX, the probability of swapping alleles of the parents for

each locus is derived from the distribution of alleles in that locus over the population and thus is adaptively adjusted with the progress of the GA. In the rest of this paper, we first briefly review previous relevant work, then describe SANUX in detail, finally present our experiment study that compares SANUX over traditional one-point, two-point and uniform crossover based on a representative set of test problems.

2 REVIEW OF RELATED WORK

2.1 ADAPTATION IN THE GA

The performance of a GA significantly depends on the operators and relevant parameters used. However, choosing the right GA operators and appropriate parameters is a difficult task. Traditionally, they are determined by experience or primary experiments from a particular domain in advance and then are fixed during the running of the GA. This kind of constant parameter setting approach is time-consuming, can lead to sub-optimal performance when parameters are inappropriately set. And what is worse lies in the nature that the optimal parameter values may vary with the evolution process of GAs. Hence, researchers have applied many adaptation techniques into GAs to enhance their performance (Eiben, Hinterding and Michalewicz, 1999). Based on the mechanism of change, adaptation in GAs can be classified into three categories: *deterministic adaptation* where the value of a strategy parameter is altered according to some deterministic rule, *adaptive adaptation* where there is some form of feedback information from the search process that is used to direct the change of a strategy parameter, and *self-adaptive adaptation* where the parameter to be adapted is encoded into the chromosomes and undergoes genetic operations (hence, also called *co-evolution*).

2.2 ADAPTATION IN CROSSOVER

Since crossover is one of the primary genetic operations in GAs, adaptation in crossover operators has long been studied and there have been many results (Spears, 1997). Generally speaking, adaptation in crossover happens in three levels from top to bottom.

2.2.1 Adapting the Type of Crossover

In this top level, crossover operators are themself adapted during a run of the GA. Davis (1989) proposed that the GA needn't apply both crossover and then mutation to the selected parent, instead it can select operators from a set of operators, each with a fixed probability. Eshelman and Schaffer (1994) proposed an adaptive mechanism that uses restarts (when converged, the population is partially or fully randomized except the best individual) and between restarts switches between two crossover operators based on their performance. Spears (1995) appended to each individual one *tag bit* that co-evolves with the individual and is used to switch between two-point and uniform crossover. If both the tag bits of two parents are 1, choose two-point crossover; if both are 0, choose uniform crossover; otherwise, choose either randomly.

2.2.2 Adapting the Rate of Crossover

In this medium level, the rate or probability of crossover is altered during a run of the GA. Julstrom (1995) proposed an adaptive mechanism that regulates the ratio between crossover and mutation based on their performance. Corne, Ross and Fang (1994) devised the COst Based operator Rate Adaptation (COBRA) method for adapting operator probabilities in timetabling problems. With COBRA the GA periodically swaps given k fixed probabilities between k operators by giving the highest probability to the operator that has been producing the most gains in fitness. Tuson and Ross (1998) extended the COBRA method by encoding into each individual the crossover and mutation probabilities as real numbers (normalized to one) that are used by and co-evolve with the individual.

2.2.3 Adapting the Crossing Position or Swapping Probability in Each Locus

In this bottom level, the position of crossing or swapping probability in each locus is adapted during a run of the GA. Rosenberg (1967) attached to each locus i an integer $x_i \in \{1, \ldots, 7\}$ and calculated the crossing probability p_i of locus i from the probability distribution defined by $p_i = x_i / \sum x_i$. Schaffer and Morishima (1987) proposed an self-adaptive scheme that appends at the end of each individual a crossover bitmap that specifies allowable crossing positions and co-evolves with the individual. Booker (1992) introduced the notion of recombination distributions which describe the probability of all possible recombination events. Different probability distributions describe different operators. White and Oppacher (1994) developed an adaptive uniform crossover where each bit string in the population is augmented at each bit position with an automation whose state maps to a crossover probability for that bit location. Levenick (1995) inserted a metabit before each bit of the individual. If the metabit was "1" in both parents swapping occurred with base probability P_b, otherwise with reduced probability P_r. Vekaria and Clark (1999) proposed the "se-

lective crossover" which biases alleles that are known to have increased an individual's fitness. It attaches a real vector to an individual to accumulate fitness information in previous generations and uses this information to preserve known fit alleles.

A unique topic in this level is the study of *gene linkage*, the property of grouping interactive genes to evolve them together. Fraser (1957a, 1957b) offered one of the earliest computer simulations of genetic systems where he suggested a crossover that associates a variable swapping probability for each locus of a string. Interaction between genes could be addressed by forming linkage groups based on their swapping probabilities. Those genes with small-valued probabilities (i.e., close to zero) form a linked group because it is unlikely for crossover to disrupt that group. Goldberg, Korb and Deb (1989) devised the "messy GA" as an attempt to explicitly link genes using variable length strings. An alternative scheme was presented by Harik and Goldberg (1996) that attempted to co-evolve the positions and values of genes using a representation which consider loci as points on a circle with the real-valued distance between points denoting their linkage. Smith and Fogarty (1996) developed a Linkage Evolving Genetic Operator (LEGO) that attaches to each gene two flags, denoting whether it is linked to its neighbours on the left and right respectively. A pair of genes had to be bi-directionally linked by setting the right flag on one and the left flag on the other before they were considered a block. LEGO allows these blocks to come together from multi-parents to form new individuals.

2.3 PROBABILITY-BASED OPTIMIZATION ALGORITHMS

Recently, based on the convergence property of the GA, a number of GA-like algorithms have been developed that replace the GA's population and crossover operator with a probabilistic representation and generation method. The Population-Based Incremental Learning (PBIL) by Baluja (1994) evolves a probability vector, the values of which are initialized to 0.5. A number of solutions are generated based on the probabilities in the vector. The probability vector is then pushed towards the generated solution with the highest evaluation. A new set of solutions are generated according to the updated vector, and the cycle continues till terminated. As search progresses, the values in the vector gradually shift to represent high evaluation solution vectors. Harik, Lobo and Goldberg (1998) proposed the compact GA (cGA) that also evolves a probability vector initialized to 0.5 for each gene locus. This probability vector is also used to generate a set of solutions but is updated with a different learning rule from PBIL.

3 STATISTICS-BASED ADAPTIVE NON-UNIFORM CROSSOVER

3.1 CANONICAL UNIFORM CROSSOVER

Uniform crossover is the generalization of n-point crossover (Syswerda, 1989). It creates offsprings by deciding, for each bit of the parent, whether to swap the allele of that bit with the corresponding allele of the other parent. The decision is made using a coin flip, i.e., the probability of swapping is $p = 0.5$. Figure 1 shows an example of applying the uniform crossover operator to two 6-bit string parents.

Swapping Prob.:	p	p	p	p	p	p
Coin Flipping:	⇓	⇓	⇓	⇓	⇓	⇓
Created Mask:	0	1	0	1	0	1
Applying Mask:		⇓		⇓		⇓
Parent P_1:	0	1	0	1	1	1
Parent P_2:	1	1	1	1	1	0
Swapping:		⇓		⇓		⇓
Child C_1:	0	1	0	1	1	0
Child C_2:	1	1	1	1	1	1

Figure 1: An example operation of the uniform crossover where $p = 0.5$.

When performing uniform crossover on two parents P_1 and P_2, we first generate a mask bit by bit by flipping a coin unbiasedly, i.e., generating 0 or 1 with an equal probability $p = 0.5$. The generated mask is then used to guide the crossover by exchanging those bits of P_1 and P_2 that correspond to the positions where there are a "1" in the mask and leaving the bits of other loci of P_1 and P_2 unchanged. As illustrated in Figure 1, the downward arrows marked by "⇓" in the lines of "Applying Mask" and "Swapping" correspond to the positions where there is a "1" in the mask and where the alleles of P_1 and P_2 are exchanged.

In parameterized uniform crossover (Spears and De Jong, 1991), the decision for each locus is made by biased coin flipping, i.e., the swapping probability p could be other than 0.5. However, the degree of bias, i.e., the value of p, is the same for all loci and hence the 1-bits are uniformly distributed over the mask. In fact, the value of p may be different for each locus, that is, the 1-bits may be non-uniformly distributed

over the mask. This is realized adaptively in SANUX.

3.2 STATISTICS-BASED ADAPTIVE NON-UNIFORM CROSSOVER

For the convenience of description and analysis, we first introduce the concepts of intrinsic attribute and extrinsic tendency of allele valuing for a gene locus. In the optimal solution (encoded in binary string) of a given problem, for a gene locus if its allele is 1 it is called *1-intrinsic*, if its allele is 0 it is called *0-intrinsic*, otherwise if its allele can be either 0 or 1 it is called *neutral*. Whether a locus is 1-intrinsic, 0-intrinsic, or neutral depends on the problem solved and encoding scheme, e.g., whether introns are inserted (Levenick, 1995). During the running of a GA, for a gene locus, if the frequency of 1's in its alleles over the population tends to increase (to the limit of 1.0) with time(generation), it is called *1-inclined*; if the frequency of 1's tends to decrease (to the limit of 0.0), it is called *0-inclined*; otherwise, if there is no tendency of increasing or decreasing, it is called *non-inclined*. Whether a locus is 1-inclined, 0-inclined, or non-inclined depends on the problem solved, encoding scheme, genetic operators and initial conditions.

Usually and hopefully as the GA progresses, those gene loci that are 1-intrinsic (or 0-intrinsic) will appear to be 1-inclined (or 0-inclined), i.e., the frequency of 1's in the alleles of these loci will eventually converge to 1 (or 0). SANUX makes use of this convergence information as feedback information to direct the crossover by adjusting the swapping probability for each locus.

We will use the frequency of 1's in the alleles in a locus over the population (equivalently we can also use the frequency of 0's as the argument) to calculate corresponding swapping probability of that locus. The frequency of 1's in a locus's alleles can be looked as the degree of convergence to "1" for that locus. Let L be the length of binary strings, $f_1(i,t)$ $(i = 1,\ldots,L)$ denote the frequency of 1's in the alleles in locus i over the population at time (generation) t and $p_s(i,t)$ $(i = 1,\ldots,L)$ denote the swapping probability of locus i at time t. Then, as shown in Figure 2, the calculation equation from $f_1(i,t)$ to $p_s(i,t)$ is given as follows:

$$p_s(i,t) = \begin{cases} f_1(i,t), & \text{if} \quad f_1(i,t) \le 0.5 \\ 1 - f_1(i,t), & \text{if} \quad f_1(i,t) > 0.5 \end{cases} \qquad (1)$$

Now during the evolution of the GA, after a new population has been generated, we first calculate the distribution of 1's $f_1(i,t)$ (hence the swapping probability $p_s(i,t)$) for each locus over the population. Then we can perform SANUX operations. When applying

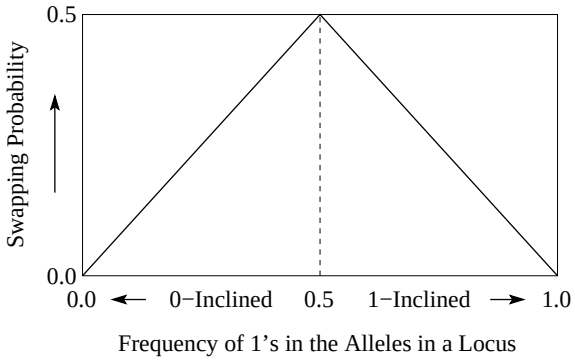


Figure 2: Calculate a locus's swapping probability.

SANUX, the mask is bit by bit generated by flipping a coin biasedly to generate a "1" with probability $p_s(i,t)$. Finally, the generated mask is used to guide the crossover the same way as it guides the traditional uniform crossover. Figure 3 shows an example of applying SANUX to the same parents as in Figure 1.

1's Freq. in loci:	0.4	0.2	0.6	0.9	0.9	0.2
Calculating:	⇓	⇓	⇓	⇓	⇓	⇓
Swapping Prob.:	0.4	0.2	0.4	0.1	0.1	0.2
Biased Flipping:	⇓	⇓	⇓	⇓	⇓	⇓
Created Mask:	1	0	1	0	0	0
Applying Mask:	⇓		⇓			
Parent P_1:	0	1	0	1	1	1
Parent P_2:	1	1	1	1	1	0
Swapping:	⇓		⇓			
Child C_1:	1	1	1	1	1	1
Child C_2:	0	1	0	1	1	0

Figure 3: An example operation of SANUX.

3.3 DISCUSSIONS ON SANUX

According to the classification of adaptation for GAs reviewed in section 2, SANUX belongs to the class of adaptive adaptation that occurs at the bottom-level of crossover because it uses the feedback information of allele distribution during the running of the GA and adjusts the swapping probability for each gene locus.

SANUX is much simpler than those adaptation mechanisms that add extra tag bit (e.g., Schaffer and Morishima, 1987; Levenick, 1995) or value (e.g., Rosen-

burg, 1967; Fraser, 1957b) per genetic bit and co-evolves these tag bits or values with each individual. With SANUX, what we add to traditional uniform crossover are spacially only one real vector that records the frequency of ones for each locus, and computationally only one statistics per generation that calculates the frequency of ones (hence the swapping probability) for each locus. This simple extra statistics added is well rewarded in the sense of computational complexity. For each crossover operation at generation t, the number of swappings on strings of length L is $L/2$ on the average with uniform crossover and $\sum_{i=1}^{i=L} p_s(i,t)$ with SANUX. When the population is randomly initialized, the frequency of 1's in the alleles (hence the swapping probability) in each locus is statistically about 0.5, which gives on the average $L/2$ crossings for SANUX. However, with the running of the GA, 1-intrinsic and 0-intrinsic genes tend to converge to 1 and 0 respectively and their associated swapping probabilities decrease according to equation (1). This results in reduced number of crossings with SANUX, i.e., $\sum_{i=1}^{i=L} p_s(i,t) < L/2$. And there's more. As the population converges, with uniform crossover, in fact, fewer and fewer 1's in the mask will involve a change, that is, many crossings are wasted on those converged bits. Most of these wasted crossings by uniform crossover are saved by SANUX. As illustrated in Figure 1 and 3, after certain generations the distribution of 1's over loci may be as shown in Figure 3, uniform crossover made no use of this information and still generated 3 crossings while SANUX generated 2.

Another more important point of SANUX is its property of implicit gene linkage. Similar to Fraser's crossover (Fraser, 1957b), SANUX adaptively links genes based on their swapping probabilities. For example, in Figure 3 loci 4 and 5 are more convergent and implicitly linked because the probability for them to co-evolve via crossover is $0.1 * 0.1 + 0.9 * 0.9 = 0.82$. SANUX differs from Fraser's crossover in that SANUX adapts one probability vector for all individuals based on one simple statistics per generation while Fraser's crossover modifies one probability vector for each individual per crossover based on a random learning rule.

Here we must note the relationship between SANUX and probability-based algorithms (Baluja, 1994; Harik, 1998). They are all based on probability distribution. However, those algorithms use the explicitly maintained probability vector to sample solutions while SANUX obtains the distribution probability from the statistics information implicit in the population. In the sense of applying probability vector, they are dual algorithms. The motivation of SANUX is to explicitly use the statistics information implicit in the population to guide the crossover.

4 THE TEST PROBLEMS

4.1 THE MAX ONES PROBLEM

The Max Ones problem simply counts the ones contained in a binary string as the fitness of that string. The aim is to maximize ones in a string. A string length of 100 bits was used for our study.

4.2 THE ROYAL ROAD FUNCTIONS

The Royal Road functions (Forrest and Mitchell, 1992) are devised to investigate GA's performance with respect to schema processing and recombination in an idealized form. Royal Road functions R_1 and R_2 contain tailor-made building blocks (schemas) based on 64-bit binary strings. Each schema s_i is given a coefficient c_i which is equal to its order $o(s_i)$ (a schema's order is the number of fixed positions within that schema). R_1 consists of 8 disjunctive order-8 schemas of which each has 8 adjacent ones. R_2 consists of four levels of schemas: level 0 (bottom level) is the same as R_1, level 1 has 4 order-16 schemas of which each combines two adjacent schemas in level 0, level 2 contains 2 order-32 schemas each combining two adjacent schemas in level 1, and finally level 3 (the optimal schema) combines the 2 schemas in level 2.

The fitness of a bit string x for $R_1(x)$ and $R_2(x)$ is computed by summing the coefficients c_i corresponding to each of the given schema s_i of which x is an instance. The optimal solutions for R_1 and R_2 are given as: $R_1(111..1) = 64$ and $R_2(111..1) = 192$.

4.3 THE L-SAT PROBLEM GENERATOR

The random L-SAT problem generator (De Jong, Potter and Spears, 1997) is a boolean satisfiability problem generator devised to investigate the effects of epistasis on the performance of GAs. It generates random boolean expressions in conjunctive normal form of clauses subject to three parameters V (number of boolean variables), C (number of disjunctive or conjunctive clauses) and L (the length of the clauses). Each clause is created by selecting L of V variables uniformly randomly and negating each variable with probability 0.5. For each generated boolean expression, the aim is to find an assignment of truth values to the V variables that makes the entire expression true. Since the boolean expression is randomly generated, there is no guarantee that such an assignment exists. The complexity of the problem varies with the parameters V, C and L. For example, increasing the

number of clauses increases the epistasis. The fitness function for the L-SAT problem is as follows:

$$f(chrom) = \frac{1}{C}\sum_{i=1}^{C} f(clause_i)$$

Where $chrom$ consists of C clauses and the fitness contribution of clause i, $f(clause_i)$, is 1 if the clause is satisfied or 0 otherwise.

In our experiments we used the same parameters as De Jong, Potter and Spears. We fixed the number of variables V to 100 and the length of the clauses L to 3. The number of clauses C is varied from 200 (low epistasis) to 1200 (medium epistasis) to 2400 (high epistasis).

5 EXPERIMENTAL RESULTS

For each experiment of combining crossover operators (1-point, 2-point, uniform crossover and SANUX) and test problems, 100 independent runs were executed. In order to have a strict comparison between crossover operators the same 100 different random seeds were used to generate populations for the 100 runs of each experiment. In all the experiments, the GA uses the fitness proportionate selection with the stochastic universal sampling (Baker, 1987) and the elitist model (De Jong, 1975), and bit flip mutation. And typically the probabilities of crossover and mutation were fixed to 0.6 and 0.001 respectively and the population size was set to 100 for each run. For each run, we recorded the best-so-far fitness every 100 evaluations. Here, only those chromosomes changed by crossover and mutation operations are evaluated and counted into the number of evaluations. Each experiment result is averaged over the 100 independent runs.

5.1 RESULTS ON MAX ONES PROBLEM

The experiment results for the Max Ones problem are shown in Figure 4. From Figure 4 it can be seen that SANUX outperforms all traditional crossover operators on the Max Ones problem.

5.2 RESULTS ON R_1 AND R_2

The experiment results on royal road functions are shown in Figure 5 and Figure 6 respectively. From these figures it can be seen that during the early stage of GA's searching, 1-point and 2-point are better than SANUX. However, after certain evaluations, when the GA has built up some useful schemas SANUX outperforms them due to its implicit gene linkage. This is also proved by SANUX's consistent advantage over

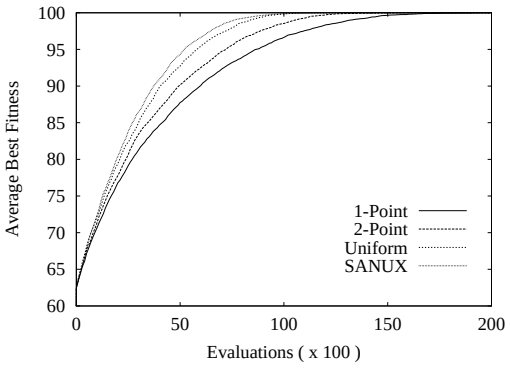


Figure 4: Average best curves for GAs with different crossover on Max Ones problem.

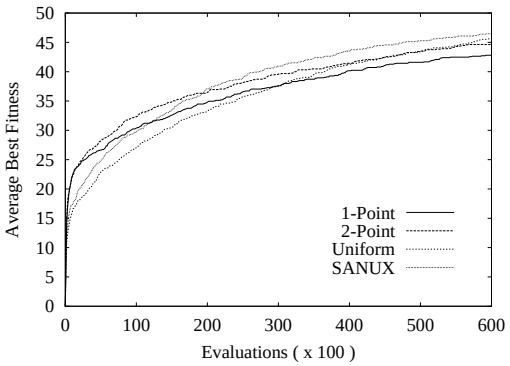


Figure 5: Average best curves for GAs with different crossover on Royal Road function R_1.

uniform crossover on both R_1 and R_2. This is further proved by the observation that SANUX outperforms traditional crossover operators much better on R_2 than on R_1. This happens because the introduction of intermediate schemas into R_2 increases the epistasis on which SANUX shows more advantage than traditional operators due to its implicit gene linkage capability.

5.3 RESULTS ON L-SAT PROBLEMS

The experiment results on L-SAT problems with low, medium and high epistasis are given in Figure 7, Figure 8, and Figure 9 respectively. From these figures it can be seen that during low epistasis all crossover operators work equally as well with uniform crossover slightly better than the other operators. However, as epistasis is increased to medium and high, SANUX outperforms all traditional crossover operators. This further shows that SANUX has more advantage over traditional crossover on problems with high epistasis due to its implicit gene linkage capability.

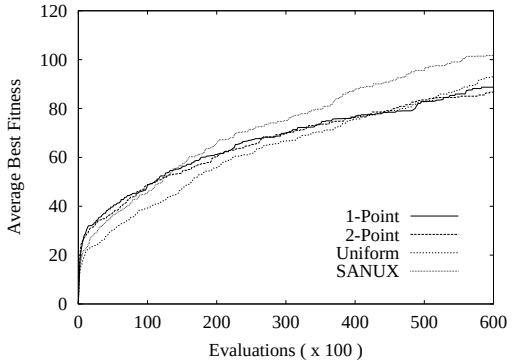


Figure 6: Average best curves for GAs with different crossover on Royal Road function R_2.

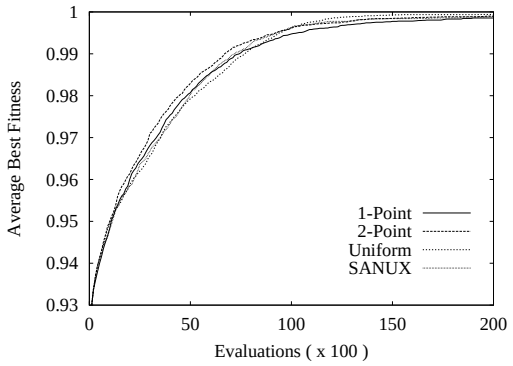


Figure 7: Average best curves for GAs with different crossover on L-SAT problems with low epistasis.

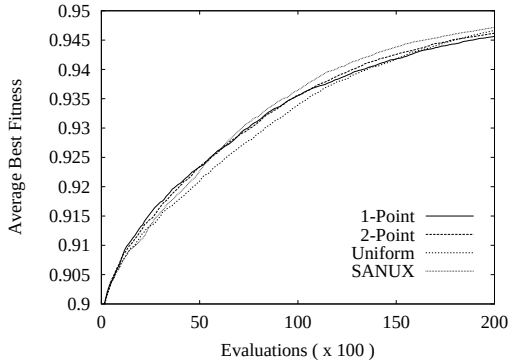


Figure 8: Average best curves for GAs with different crossover on L-SAT problems with medium epistasis.

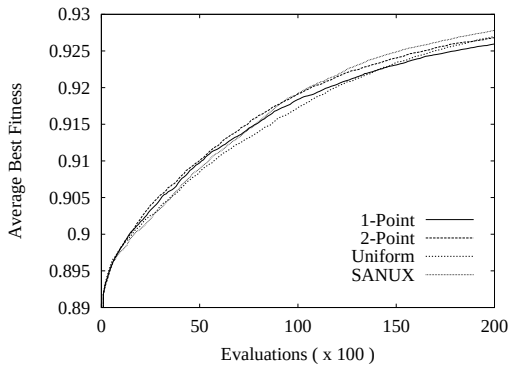


Figure 9: Average best curves for GAs with different crossover on L-SAT problems with high epistasis.

6 CONCLUSIONS

In this paper, a new statistics-based adaptive non-uniform crossover operator, SANUX, is proposed. The motivation of SANUX is to make use of the statistics information implicitly contained in the population explicitly to guide the crossover operation. SANUX achieves this by using the statistics information of the allele distribution in the current population to adjust the swapping probability for each gene locus adaptively during the evolutionary progress of the GA. An important property of SANUX is its capability of implicitly linking gene groups. SANUX is also computationally efficient through saving crossings.

The preliminary experiment results of this study show that SANUX performs better than traditional one-point, two-point and uniform crossover operators on a set of typical GA's test problems. The experiment results indicate that SANUX may be a good candidate as a crossover operator for GAs. Since SANUX works at the bottom-level of crossover, it can be easily combined into the other two levels of adaptation in crossover. Additionally, SANUX is dual to those probability-based algorithms in the sense of using probability vector and thus may act as the basis for analyzing and designing new related algorithms.

References

J. E. Baker (1987). Reducing bias and inefficiency in the selection algorithms. In J. J. Grefenstelle (ed.), *Proc. 2nd Int. Conf. on Genetic Algorithms*, 14-21.

S. Baluja (1994). Population-based incremental learning. Technical Report CMU-CS-95-193, Carnegie Mellon University.

L. B. Booker (1992). Recombination distributions for genetic algorithms. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, 29-44.

D. Corne, P. Ross and H.-L. Fang (1994). GA research note 7: fast practical evolutionary timetabling. Technical Report, Department of Artificial Intelligence,

University of Edinburgh, UK.

L. Davis (1989). Adapting operator probabilities in genetic algorithms. In D. Schaffer (ed.), *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 60-69.

K. A. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Abor.

K. A. De Jong, M. A. Potter and W. M. Spears (1997). Using problem generators to explore the effects of epistasis. In T. Bäck (ed.), *Proc. of the 7th Int. Conf. on Genetic Algorithms*, 338-345.

A. E. Eiben, R. Hinterding, and Z. Michalewicz (1999). Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation* **3**(2):124-141.

L. Eshelman, R. Caruana, and J. D. Schaffer (1989). Biases in the crossover landscape. In *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 10-19.

L. Eshelman and J. D. Schaffer (1994). Productive recombination and propagating and preserving schemata. In M. Vose and D. Whitley (eds.), *Foundations of Genetic Algorithms 3*, 299-313.

A. S. Fraser (1957a). Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Sciences* **10**:484-491.

A. S. Fraser (1957b). Simulation of genetic systems by automatic digital computers. II. Effects of linkage or rates of advance under selection. *Australian Journal of Biological Sciences* **10**:492-499.

S. Forrest and M. Mitchell (1992). Relative building-block fitness and the building-block hypothesis. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2.*

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley,

G. E. Goldberg, B. Korb and K. Deb (1989). Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems* **3** (5):493-530.

G. Harik and G. E. Goldberg (1996). Learning linkage. In *Foundations of Genetic Algorithms 4*, 247-272.

G. Harik, F. Lobo and G. E. Goldberg (1998). The compact genetic algorithm. In *Proc. of the 1998 IEEE Conference on Evolutionary Computation*, 523-528.

J. H. Holland (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

B. Julstrom (1995). What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In L. J. Eshelman (ed.), *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 81-87.

J. Levenick (1995). Metabits: genetic endogenous crossover control. In L. J. Eshelman (ed.), *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 88-95.

J. Reed, R. Toombs, and N. A. Barricelli (1967). Simulation of biological evolution and machine learning: I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology* **17**:319-342.

R. S. Rosenberg (1967). *Simulation of Genetic Populations with Biochemical Properties.* PhD Thesis, University of Michigan, Ann Abor.

J. D. Schaffer and A. Morishima (1987). An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstelle (ed.), *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, 36-40.

J. E. Smith and T. C. Fogarty (1996). Recombination strategy adaptation via evolution of gene linkage. In *Proc. of the 3rd IEEE Int. Conf. on Evolutionary Computation*, 826-831.

W. M. Spears and K. A. De Jong (1991). On the virtues of parameterized uniform crossover. In *Proc. 4th Int. Conf. on Genetic Algorithms*, 230-236.

W. M. Spears (1995). Adapting crossover in evolutionary algorithms. In *Proc. of the 4th Annual Evolutionary Programming Conference*, 367-384.

W. M. Spears (1997). Recombination parameters. In T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*, E1.3.1-E1.3.11. Oxford University Press.

G. Syswerda (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (ed.), *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 2-9.

A. Tuson and P. Ross (1998). Adapting operator settings in genetic algorithms. *Evolutionary Computation* **6**(2):161-184.

K. Vekaria and C. Clarck (1999). Biases introduced by the adaptive recombination operators. In T. Bäck (ed.), *Proc. of the 1999 Genetic and Evolutionary Computation Conference*, 670-677. San Mateo, CA: Morgan Kaufmann.

T. White and F. Oppacher (1994). Adaptive crossover using automata. In Y. Davidor, H. Schwefel and R. Männer (eds.), *Proc. of the 3rd Int. Conf. on Parallel Problem Solving from Nature*, 229-238.

An Enhanced Annealing Genetic Algorithm for Multi-Objective Optimization Problems

Zhong-Yao Zhu
Dept. of Computer Science & Engineering
Chinese University of Hong Kong
Hong Kong

Kwong-Sak Leung
Dept. of Computer Science & Engineering
Chinese University of Hong Kong
Hong Kong

Abstract

In this paper, we present a new algorithm — an Enhanced Annealing Genetic Algorithm for Multi-Objective Optimization problems (MOPs). The algorithm tackles the MOPs by a new quantitative measurement of the Pareto front coverage quality — Coverage Quotient. We then correspondingly design an energy function, a fitness function and a hybridization framework, and manage to achieve both satisfactory results and guaranteed convergence.

1 Introduction

Many real world decision-making problems involve simultaneous optimization of several incommensurable and often competing objectives. Usually, there is no single optimal solution, but rather a set of alternative solutions. These solutions are known as *Pareto-optimal* solutions, to which no other solutions in the search space are superior in all the objectives.

Often, for a multi-objective optimization problem (MOP), a full perception of all the *Pareto-optimal* solutions would be highly desirable (even imperative) to the decision-makers so that a comprehensive and high quality decision can be made. To achieve this, a practical algorithm should have the capability of simultaneously searching a set of *Pareto-optimal* solutions. Among various candidates, Evolutionary Algorithms (EAs) are particularly suitable for this purpose.

In this paper, we present a new algorithm — an Enhanced Annealing Genetic Algorithm (eAGA) for MOPs. The algorithm tackles the MOPs by introducing a quantitative measurement of the Pareto front coverage quality — Coverage Quotient (CQ). Based on CQ we can derive the energy function in the Simulated Annealing Algorithm (SAA) and the fitness function in the Genetic Algorithm (GA). In the algorithm, a population is interpreted as a state in the search space. The proposed algorithm explores the search space from state to state by means of genetic operations (selection, crossover, and mutation) and converges to a minimal energy state containing only *Pareto optima*. Both satisfactory results and guaranteed convergence can be achieved with the eAGA.

The paper is organized as follows: Section 2 gives a brief introduction to multi-objective optimization problems and an overview of the EA implementation. Our algorithm is detailed in section 3, followed by the theoretical analysis in section 4. The simulation results are presented in section 5. The last section is the conclusion and the future work.

2 Background

In this section, the basic concepts of multi-objective optimization problems are introduced. A brief survey of EAs implementation in MOPs and their strength and weakness are briefed as the motivation of our algorithm.

2.1 Multi-objective Optimization

A general multi-objective optimization problem is to optimize a set of objectives subject to some constrains. Mathematically, a MOP may be stated as in (Rao, 1991):

$$\begin{array}{lll} \text{Min/Max} & f_i(x) & i = 1, 2, ..., N \\ \text{Subject to} & g_j(x) \leq 0 & j = 1, 2, ..., J \\ & h_k(x) = 0 & k = 1, 2, ..., K \end{array}$$

where alternative x is a p-dimensional vector having p design or decision variables, $f_i(x)$ $(i = 1, 2, ..., N)$ are the objective functions, and $g_j(x)$ $(j = 1, 2, , ..., J)$ and $h_k(x)$ $(k = 1, 2, ..., K)$ are the constraint functions.

Solutions to the MOP are mathematically characterized in terms of the non-dominated alternatives. Let $F(x) = (f_1(x), f_2(x), ..., f_N(x))$ represent the objective vector. In a minimization problem, for instance, alternative $x^{(1)}$ is said to be partially less than alternative $x^{(2)}$ (denoted by $x^{(1)} \prec x^{(2)}$), if no component of $F(x^{(2)})$ is less than that of $F(x^{(1)})$ and there is at least one component of $F(x^{(2)})$ is strictly greater than that of $F(x^{(1)})$. If $x^{(1)}$ is partially less than $x^{(2)}$, alternative $x^{(1)}$ is said to *dominate* $x^{(2)}$. Any alternative that is not dominated by others is said to be a non-dominated point. Any non-dominated points associated with the MOP are called optimal solutions (more precisely, *Pareto-optimal* solutions or non-dominated solutions) of the MOP. Usually, the image of all *Pareto optimal* solutions in objective space is called the Pareto front.

2.2 Multi-objective Evolutionary Algorithms

According to Zitzler (Zitzler, 1998), the current EA implementation can be categorized as plain aggregation approaches, population-based non-Pareto approaches and Pareto-based approaches.

Plain aggregation approaches combine the objectives into a higher scalar function which is used for fitness calculation; they produce one single solution and require profound domain knowledge which is often not available. Population-based non-Pareto approaches, however, are able to evolve multiple non-dominated solutions in parallel; thereby, the population is mostly monitored for *Pareto optima*. However in contrast to the Pareto-based approaches, they do not make direct use of the concept of Pareto dominance. Such designed algorithms are effective to some extent, but they usually suffer from prematuring to some special areas (Schaffer, 1985). Pareto-based EAs compare solutions according to the $\prec$ relation in order to determine the reproduction probability of each individual. This strategy can meet our requirement well. Nevertheless identifying the $\prec$ relation among individuals usually brings higher running time consumption (Srinivas, 1994).

More comprehensive overviews of EAs in MOPs can be found in (Zitzler, 1999) (Carlos, 1999),

3 The Enhanced Annealing Genetic Algorithm

In this section, we present the new algorithm — an Enhanced Annealing Genetic Algorithm (eAGA) for MOPs. As most of non-dominated sorting GAs, we design our algorithm based on two considerations: 1) the non-dominance of solutions, and 2) the coverage of the Pareto front.

As suggested by its name, the new algorithm is a hybridization of the Simulated Annealing Algorithm (SAA) (Laarhoren, 1989) and the Genetic Algorithm (GA) (Goldberg, 1989). The algorithm tackles the MOPs by introducing a quantitative measurement of the Pareto front coverage quality — Coverage Quotient (CQ). The energy function in SAA and the fitness function in GA are derived from this measurement correspondingly. In the algorithm, a population is interpreted as a state in the search space. The neighborhood relationship between states is defined in terms of their individual discrepancy. The proposed algorithm explores the search space from state to state by means of genetic operations (selection, crossover, and mutation). Ultimately, the exploration converges to a minimal energy state which can be proved to contain only *Pareto optima*. The details are given in the following subsections.

3.1 Uniform Expression of Population

For clarity and simplicity, we define an alternative representation of populations. In this representation, all populations that differ from each other only in the order of individuals are treated as the same and expressed in a unique format.

Suppose the individuals are all expressed as L bits binary strings. Then the individual space is given by $\{0,1\}^L$ and can be isomorphic to the finite-state space:

$$\{0,\ 1, \cdots,\ i, \cdots, 2^L - 1\}$$

where i $(0 \leq i \leq 2^L - 1)$ is the index of individuals (Reeves, 1993). From this point of view, a population Φ with size n can then be represented as $\Phi = (\phi_0\ \phi_1\ \cdots\ \phi_i \cdots\ \phi_{2^L-1})$ where ϕ_i is the number of times of individual i appearing in the population Φ. It is clear that there are at most n nonzero elements in Φ and $\sum_{i=0}^{2^L-1} \phi_i = n$.

The set consisting of all the populations with size n is denoted as S_n. Each population is interpreted as a state in S_n. The neighbors of a state $\Phi = (\phi_0\ \phi_1\ \cdots\ \phi_{2^L-1})$ are defined as all the populations $\Phi' = (\phi'_0\ \phi'_1\ \cdots\ \phi'_{2^L-1})$ which satisfy: $\sum_{i=0}^{2^L-1} |\phi_i - \phi'_i| = 2$. All the neighbors of Φ are denoted as $N(\Phi)$.

3.2 The Pareto Front Coverage Quality Measurement

The Coverage Quotient (CQ) gives a quantitative measurement of the Pareto front coverage quality. It is based on the idea that a good coverage of the Pareto front by a population Φ should minimize the potential that a non-Pareto point is wrongly judged as a Pareto point when compared against Φ. The formal definition is given as follows:

Definition. 1: Given F is a MOP and N is the number of objectives:.

- Let $Pmin_i, Pmax_i$ be the minimum and maximum of all the Pareto points in objective i ($1 \leq i \leq N$), the hyper-cube

  $$U = [Pmin_1, Pmax_1] \times ... \times [Pmin_N, Pmax_N],$$

  which contains all the Pareto points, is defined as P-Cube of F.

- Given a population $\Phi = \{d_1, d_2, ..., d_n\}$,

  $$\begin{array}{ll} D(d_i) & = [F(d_i)_1, \infty] \times ... \times [F(d_i)_N, \infty], \text{ and} \\ D_n(\Phi) & = D(d_1) \cup D(d_2) \cup ... \cup D(d_n) \end{array} \quad (1)$$

  are defined as the dominating region of individual d_i ($1 \leq i \leq n$) and dominating region of population Φ respectively.

- The Coverage Quotient (CQ) of Φ is defined as

  $$\mathrm{CQ}(\Phi) = |U| - |U \cap D_n(\Phi)|. \quad (2)$$

- The population Φ is said to be an optimal coverage of the Pareto front if

  $$\mathrm{CQ}(\Phi) = \min_{\Phi'}\{\mathrm{CQ}(\Phi')|\ \Phi' \in S_n\}.$$

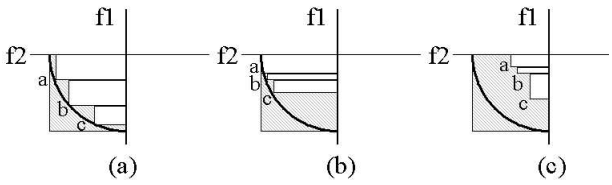


Figure 1: The Coverage Quotient in 3 cases: (a) evenly distributed Pareto points, (b) crowded Pareto points, and (c) non-Pareto points

A graphic illustration of Def.1 is given in Figure 1. Assuming the curve is the Pareto front, the shadowed regions represent the Coverage Quotient in 3 cases respectively.

In Def.1, when we identify an individual's non-dominance by comparing it with population Φ, it can be correctly justified only if it is located inside $D_n(\Phi)$. Thus we can reduce the risk of misjudgment by maximizing the dominating region $D_n(\Phi)$ or equivalently, minimizing the Coverage Quotient $CQ(\Phi)$. However, a direct calculation of CQ from Formulas (1) and (2) is difficult for high dimension objectives. Thus in Def.2, an alternative definition of the Coverage Quotient is presented:

Definition 2: Given F, N, Φ, U and $D(\cdot)$ as in Def.1:

- Let $P(\Phi) = \{d^1, d^2, .., d^{n_p}\} \subset \Phi$ ($n_p \leq n$) be all the individuals which satisfy;
  1. $d^i \neq d^j$ ($1 \leq i, j \leq n_p$);
  2. $\forall\ d^i \in P(\Phi)$, $\nexists\ d_j \in \Phi$, s.t. $d_j \prec d^i$ ($1 \leq i \leq n_p$, $1 \leq j \leq n$).

  The Coverage Quotient (CQ) of Φ is defined as

  $$\mathrm{CQ}(\Phi) = -\sum_{d^i \neq d^j \in P(\Phi)} |(D(d^i) \cup D(d^j)) \cap U|.$$

In Def.2, the calculation of CQ just needs n^2 running time complexity. Meanwhile, with such defined CQ, the proposed algorithm can be guaranteed to converge to a population consisting of only distinct Pareto individuals as proved in section 4.

Correspondingly, we define the energy function of a state Φ as:

$$E(\Phi) = \mathrm{CQ}(\Phi),$$

and the fitness function of an individual $d_i \in \Phi$ ($1 \leq i \leq n$) as:

$$fit(d_i, \Phi) = e^{\frac{CQ(\Phi \setminus d_i) - MinCQ(\Phi)}{MaxCQ(\Phi) - MinCQ(\Phi)}},$$

in which:

$$\begin{array}{ll} MinCQ(\Phi) & = \min_{d \in \Phi}\{CQ(\Phi \setminus d)\} \\ MaxCQ(\Phi) & = \max_{d \in \Phi}\{CQ(\Phi \setminus d)\} \end{array}$$

3.3 State Transformation (ST) Operation

In the eAGA, the ST operation is the primary search technique. A GA-like evolutionary process is adopted to form the backbone of the ST operation. Furthermore, an additional acceptance procedure is employed to guarantee the global convergence of our algorithm. The framework of this operation is given as follows: For a population $\Phi = \{\phi_0\ \phi_1 ..... \phi_{2^L-1}\}$:

1. Perform Roulette selection on Φ to choose two individuals d_1 and d_2 as the parents.
2. Perform crossover and mutation on d_1 and d_2 to produce two children d_1' and d_2'.

3. Randomly select a parent $d \in \{d_1, d_2\}$ and a child $d' \in \{d'_1, d'_2\}$.
4. Replace d in Φ with d' to form a new population Φ'.
5. Accept Φ' to be the new state at probability $\min\{1, \frac{P(d,\Phi')}{P(d',\Phi)}\}$

in which, $P(k, \Phi)$ is the probability at which individual k ($k \in \{0, 2^L - 1\}$) is produced by performing selection, crossover, and mutation on population Φ. It is calculated as follows:

Given the indices of the nonzero elements in Φ are

$$0 \le p_1 < p_2 < p_3 < ....p_{n'} \le 2^L - 1 \quad (n' \le n),$$

then

$$P(k, \Phi) = \widetilde{F}^T(\Phi)\widetilde{R}(k)\widetilde{F}(\Phi), \tag{3}$$

where $\widetilde{F}(\Phi) = (\widetilde{F}_0(\Phi), ..., \widetilde{F}_{n'}(\Phi)) \in R^{n'}$ with

$$\widetilde{F}_i(\Phi) = \frac{fit(p_i, \Phi)\phi_{p_i}}{\sum_{j=1}^{n'} fit(p_j, \Phi)\phi_{p_j}} \qquad i = 1, ..., n'. \tag{4}$$

and $\widetilde{R}(k) = (\widetilde{R}(k)_{i,j}) = (r_{p_i,p_j}(k)) \in R^{n' \times n'}$.

$r_{i,j}(k)$ $(i, j = p_1, \ldots, p_{n'})$ is the probability at which individual k is produced from the crossover and mutation of the individuals i and j. It is computed by means of the transformation:

$$r_{i,j}(k) = r_{i\oplus k, j\oplus k}(0) \tag{5}$$

and

$$\begin{aligned} r_{i,j}(0) \quad &= \tfrac{1}{2}\sum_{k=0}^{L} \tfrac{R_c}{L+1}((1-R_m)^{L-H(m_1(i,j,k))}R_m^{H(m_1(i,j,k))} \\ &+(1-R_m)^{L-H(m_2(i,j,k))}R_m^{H(m_2(i,j,k))}) \\ &+\tfrac{1}{2}(1-R_c)((1-R_m)^{L-H(i)}R_m^{H(i)} \\ &+(1-R_m)^{L-H(j)}R_m^{H(j)}) \end{aligned}$$

where

$$\begin{aligned} m_1(i,j,k) &= i \otimes (2^k - 1) \oplus j \otimes \overline{(2^k - 1)}, \\ m_2(i,j,k) &= i \otimes \overline{(2^k - 1)} \oplus j \otimes (2^k - 1), \quad 0 \le k \le L, \end{aligned}$$

with R_c being the crossover rate; R_m being the mutation rate; $\oplus$ being the *exclusive-or* operator; $\otimes$ being the *logical-and* operator; $-$ being the *inverse* operator; and H(m) being the Hamming distance between individuals m and 0 (Vose, 1991) (Vose, 1995).

3.4 P-Cube Approximation

Notice that identifying P-Cube requires pre-knowledge of all the *Pareto optima*, which is usually unavailable in most cases. In the eAGA, a dynamic estimation approach is adopted which can approximate the P-Cube as the algorithm proceeds. In the algorithm, two arrays $Bmin_i$ and $Bmax_i$ and $2N$ binary strings $Imin_i$ and $Imax_i$ $(1 \le i \le N)$ are maintained. The $Bmin_i$ and $Bmax_i$ record the minimal and maximal values in the ith objective of all the Pareto points ever found and the $Imin_i$ and $Imax_i$ $(1 \le i \le N)$ record the individuals which attain $Bmin_i$ and $Bmax_i$ in the ith objective respectively. For $Imin_i/Imax_i$, it is replaced by a new Pareto point d iff

- $d \prec Imin_i/Imax_i$; or
- d and $Imin_i/Imax_i$ are non-dominated to each other, but $F(d)_i < Bmin_i/F(d)_i > Bmax_i$.

(Note: / denotes or).
Once an $Imin_i/Imax_i$ is modified, $Bmax$ and $Bmin$ must be updated correspondingly by:

$$\begin{aligned} Bmin_i \quad &= \min \quad \{F(Imin_k)_i, F(Imax_k)_i |\ k = 1, ..., N\} \\ Bmax_i \quad &= \max \quad \{F(Imin_k)_i, F(Imax_k)_i |\ k = 1, ..., N\} \\ &\quad\ (1 \le i \le N.) \end{aligned} \tag{6}$$

It is proved in section 4 that this estimation can approximate and eventually converge to the P-Cube as the algorithm progresses.

3.5 Algorithm Framework

The overall framework of our algorithm is given as follows:

1. Initialization
   - **1.1** Select an initial population Φ
   - **1.2** Select an initial temperature T
   - **1.3** Select an annealing function $S(T) = \alpha \cdot T$
   - **1.4** Perform non-dominated sorting on Φ and fill the $Bmin$, $Bmax$, $Imin$ and $Imax$
2. State Transformation
   - **2.1** Perform Roulette selection on Φ to choose two individuals d_1 and d_2 as the parents.
   - **2.2** Perform crossover and mutation on d_1 and d_2 to produce two children d'_1 and d'_2.
   - **2.3** Randomly select a parent $d \in \{d_1, d_2\}$ and a child $d' \in \{d'_1, d'_2\}$.
   - **2.4** Replace d in Φ with d' to form a new population Φ'.
   - **2.5** Update the $Bmin$, $Bmax$, $Imin$ and $Imax$ with individual d'.
   - **2.6** Accept Φ' to be the new state at probability

     $$\min(1, \frac{P(d,\Phi')}{P(d',\Phi)}) \min\{1, e\frac{E(\Phi) - E(\Phi')}{T}\}$$

   - **2.7** If Φ' is accepted, then $\Phi := \Phi'$,
3. $T := S(T)$

4. If stop criterion is not met, then goto 2.1
5. Exit.

Note: α is an annealing parameter which satisfies $0 < \alpha < 1$.

4 Convergence Analysis

In this section, the theoretical analysis of the proposed algorithm is presented.

Lemma 1: For MOP F, suppose $Bmin_i^g$ and $Bmax_i^g$ $(1 \le i \le N)$ are minimal and maximal values of all Pareto points in the ith objective; $Imin_i^g$ and $Imax_i^g \in \{0,1\}^L$ $(1 \le i \le N)$ are the individuals which attain $Bmin_i^g$ and $Bmax_i^g$ in the ith objective respectively. Given mutation rate R_m is nonzero, the estimations $Bmin_i$ and $Bmax_i$ will converge to $Bmin_i^g$ and $Bmax_i^g$ $(1 \le i \le N)$ as the algorithm progresses.

Proof: We prove the Lemma in two steps:

1. We will prove that $Imin_i^g$ $(1 \le i \le N)$ will be found as the algorithm proceeds. Let p_i^n be the probability that $Imin_i^g$ is not found in iteration n. It is obvious that $p_i^n \le 1 - R_m^L < 1$. Then the probability P_i^n that $Imin_i^g$ $(1 \le i \le N)$ is not found in the first n iteration is:

$$P_i^n = \prod_{j=1}^{n} p_i^j < (1 - R_m)^n \longrightarrow 0 \ (n \to \infty).$$

2. We will prove that once an $Imin_i^g$ $(1 \le i \le N)$ is found, it cannot be replaced by other individuals. Suppose $Imin_i^g$ is found, then an individual d will replace it iff

   Case 1: $d \prec Imin_i^g$; or

   Case 2: d and $Imin_i^g$ are non-dominated to each other, but $F(d)_i < Bmin_i^g$.

   Because $Imin_i$ is a Pareto point, Case 1) would not happen. If Case 2) happens, this means that there exists another non-dominated individual d' satisfies: $F(d')_i < Bmin_i^g$ which contradicts with the assumption.

The $Imax_i^g$ $(1 \le i \le N)$ case can be proved in the same way.

Combining 1, 2 and Formula (6), we finish the proof of Lemma 1.

Lemma 2: Given population $\Phi = \{d_1, d_2, ..., d_n\}$ is the optimal coverage in Def.1, then d_i $(1 \le i \le n)$ are all Pareto individuals. (Suppose there exist more than n Pareto individuals).

Proof: (By contradiction). Suppose d_k $(1 \le k \le n)$ is not a Pareto individual. Then there exists a Pareto individual d which dominates d_k.

1. If $d \in \Phi$, then replace d_k with a Pareto individual $d' \notin \Phi$ to form a new population Φ'. It is easy to see that:
   $$E(\Phi) > E(\Phi'),$$
   which contradicts with the definition of optimal coverage.
2. If $d \notin \Phi$, then replace d_k with d to form a new population Φ'. It is the same as in 1 that:
   $$E(\Phi) > E(\Phi'),$$
   which contradicts with the definition of optimal coverage.

This finishes our proof of Lemma 2.

Lemma 3: Given population $\Phi = \{d_1, d_2, ..., d_n\}$ is the optimal coverage in Def.2, then d_i $(1 \le i \le n)$ are all distinct Pareto individuals. (Suppose there exist more than n Pareto individuals).

Proof: The proof of Lemma 3 is similar to that of Lemma 2 with minor modification.

Theorem : As T $\to$ 0, the eAGA converges to a population consisting of only distinct Pareto individuals.

Proof: It is clear that for any fixed temperature T, as the population evolves, the eAGA defines a homogeneous finite state population Markov chain. Let M be the number of total populations. Then the probability transition matrix of the Markov chain can be expressed as (Iosifescu, 1980): For states Φ^i and Φ^j,

$$P_{i,j}(T) = \begin{cases} G_{i,j}(T) A_{i,j}(T) & j \ne i \\ 1 - \sum\limits_{l=1, l \ne i}^{M} G_{i,l}(T) A_{i,l}(T) & j = i \end{cases}. \quad (7)$$

Here

$$G_{i,j}(T) = \begin{cases} P(\Phi^i, \Phi^j) min\{1, \frac{P(d,\Phi^i)}{P(d',\Phi^j)}\} & \Phi^j \in N(\Phi^i) \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

and $$A_{i,j}(T) = min\{1, e^{(\frac{E(\Phi^i) - E(\Phi^j)}{T})}\}, \quad (9)$$

in which: $d \in \Phi^j$ and $d \notin \Phi^i$; $d' \in \Phi^i$ and $d' \notin \Phi^j$.

From Formula 8, we can see that $G_{i,j}(T) = G_{j,i}(T)$.

From Formula 9, it follows that whenever $E(\Phi^i) \le E(\Phi^j) \le E(\Phi^k)$,

$$A_{i,k}(T) = min\{1, e^{(\frac{E(\Phi^i) - E(\Phi^k)}{T})}\} = A_{i,j}(T) A_{j,k}(T)$$

and whenever $E(\Phi^i) \le E(\Phi^j)$, $0 \le A_{i,j}(T) \le 1$ and

$$\lim_{T \to 0} e^{(\frac{E(\Phi^i) - E(\Phi^j)}{T})} = 0.$$

Accordingly, $\lim_{T\to 0} A_{i,j}(T) = 0$. By the Folklore's lemma (Laarhoren, 1989), the stationary distribution $q(T)$ of the Markov chain exists and satisfies

$$\lim_{T\to 0} q_i(T) = \frac{1}{|S_{opt}|}\chi_{S_{opt}}(i), \tag{10}$$

in which $S_{opt} = \{\Phi | E(\Phi) \text{ attains the minimum}\}$, and

$$\chi_{S_{opt}}(i) = \begin{cases} 1 & i \in S_{opt} \\ 0 & \text{otherwise.} \end{cases}$$

from Formula 10, $\lim_{T\to 0} \Phi(T) \in S_{opt}$ follows.

This finishes the proof of the Theorem.

5 Simulation

In this section, we show the effectiveness and efficiency of the eAGA in a set of simulations. The test problems are given as follows:

$$\begin{aligned} Min.\ T_1(x) &= (f_1(x), f_2(x)) \\ f_1(x) &= 1 - exp(-\textstyle\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2) \\ f_1(x) &= 1 - exp(-\textstyle\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2) \\ \text{where} &\quad -4 \le x_i \le 4,\ n = 3 \end{aligned}$$

$$\begin{aligned} Min.\ T_2(x) &= (f_1(x), f_2(x)) \\ f_1(x) &= x_1 \\ f_2(x) &= g(x)h(f_1(x), g(x)) \\ g(x) &= 1 + 9 \times \textstyle\sum_{i=2}^{n} x_i/(n-1) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \\ \text{where} &\quad 0 \le x_i \le 1,\ n = 30 \end{aligned}$$

$$\begin{aligned} Min.\ T_3(x) &= (f_1(x), f_2(x)) \\ f_1(x) &= x_1 \\ f_2(x) &= g(x)h(f_1(x), g(x)) \\ g(x) &= 1 + 9 \times \textstyle\sum_{i=2}^{n} x_i/(n-1) \\ h(f_1, g) &= 1 - (f_1/g)^2 \\ \text{where} &\quad 0 \le x_i \le 1,\ n = 30 \end{aligned}$$

$$\begin{aligned} Min.\ T_4(x) &= (f_1(x), f_2(x)) \\ f_1(x) &= x_1 \\ f_2(x) &= g(x)h(f_1(x), g(x)) \\ g(x) &= 1 + 9 \times \textstyle\sum_{i=2}^{n} x_i/(n-1) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_i) \\ \text{where} &\quad 0 \le x_i \le 1,\ n = 30 \end{aligned}$$

The simulations are carried out to verify: 1) the efficiency of the eAGA, and 2) the effectiveness of the Coverage Quotient in both definitions. For these purposes, the execution results of eAGA-I, eAGA-II, SPEA (Zitzler, 1999) and NSGA (Srinivas, 1994) are compared in terms of: 1) the non-dominance of resulting solutions, and 2) the coverage of the Pareto front, which are two main considerations in most of the current EA-based MOP algorithms. (**Note**: eAGA-I and eAGA-II are abbreviations for the eAGA with CQ defined by Def.1 and Def.2 respectively).

To make the comparisons fair, the algorithms are executed 30 times on each of the test problems. In each run, all algorithms begin from the same initial population. The final results are taken as the average of these 30 runs. All algorithms are executed for the same length of time.

Independent of the algorithms and the test problems, each simulation is carried out using the following parameters:

Population size	:	100
Crossover rate	:	0.8
Mutation rate	:	0.01
Individual length	:	12
Niching parameters σ_{share}	:	0.48862
Elitist population size	:	100

Particularly, in the eAGA we take:

Initial Temperature T	:	1000
Annealing Parameter α	:	0.97

In Figures 2 to 5, the Pareto fronts achieved by the different algorithms are displayed.

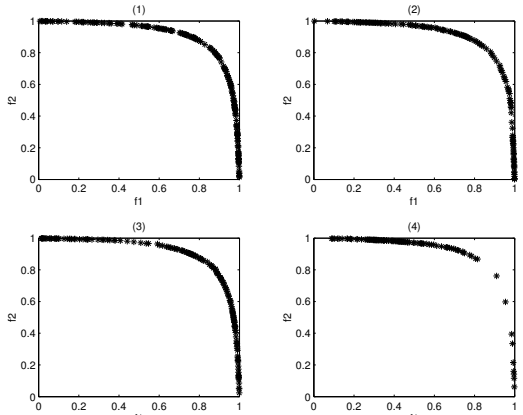


Figure 2: Test function $T1$

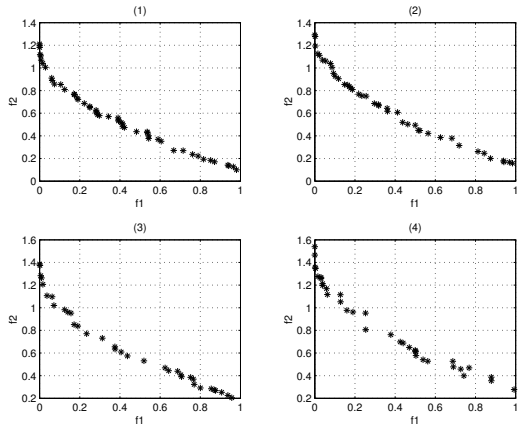


Figure 3: Test function $T2$

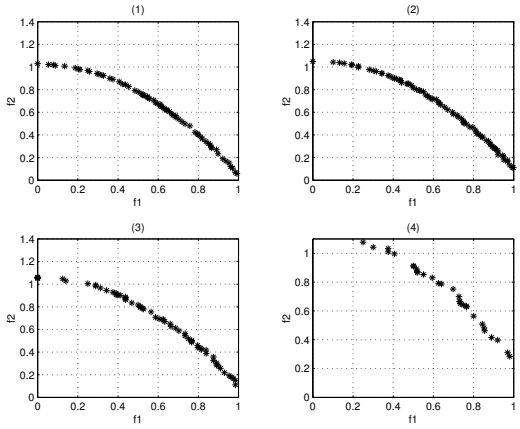


Figure 4: Test function $T3$

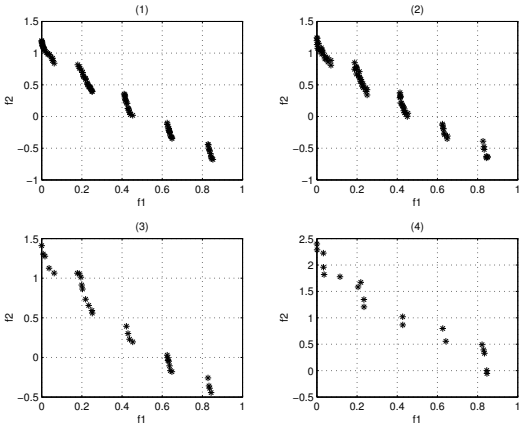


Figure 5: Test function $T4$
(1): eAGA-I (2): eAGA-II (3): SPEA (4): NSGA

It can be observed from the figures that eAGA-I has the best performance in all the four test problems. In T2, T3, and T4, eAGA-II can yield similar performance as eAGA-I. However, in T1, SPEA slightly outperforms eAGA-II which misses some *Pareto optima* close to the boundaries.

In measuring the non-dominance of results, we adopt a quantitative metric, *C metric*, presented in (Zitzler, 1999). To identify the non-dominance of a solution, we need to compare it with all the other individuals in the search space. This is definitely unrealistic. Thus, instead of identifying the absolute non-dominance of a solution, *C metric* compares the non-dominance relationship between the outcomes of two algorithms. Given a pair of algorithms A_1 and A_2, *C metric* estimates the non-dominance of A_1 by calculating the percentage of the solutions of A_1 which are dominated by those of A_2. Mathematically, *C metric* is defined as follows:

Let $X', X'' \subseteq X$ be two sets of decision vectors. The function C maps the ordered pair (X', X'') to the interval [0,1]:

$$C(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a' \prec a''\}|}{|X''|}.$$

The value $C(X', X'') = 1$ means that all solutions in X'' are dominated by or equal to solutions in X'. The opposite, $C(X', X'') = 0$ represents the situation that none of the solutions in X'' are covered by the set X'. Note that both $C(X', X'')$ and $C(X'', X')$ have to be considered, since $C(X', X'')$ is not necessarily equal to 1-$C(X'', X')$.

The comparison results of *C metric* are given in Table 1. For each ordered algorithm pair, there are 30 C values according to the 30 runs performed. Each C value is computed on the basis of the non-dominated sets achieved by the pair of algorithms with the same initial population. The final result is taken as the average of these 30 C values.

Table 1: Comparison of $C(X', X'')$ and $C(X'', X')$, in which X'' is the outcomes of eAGA-I and X' is the outcomes of eAGA-II, SPEA, and NSGA respectively.

$C(x', x'')$	$T1$	$T2$	$T3$	$T4$
eAGA-II	0.1791	0.3492	0.2646	0.4751
SPEA	0.2388	0.1384	0.4101	0.2214
NSGA	0.1194	0.0776	0.1124	0.1041

$C(x'', x')$	$T1$	$T2$	$T3$	$T4$
eAGA-II	0.3024	0.7152	0.5490	0.6964
SPEA	0.2928	0.8412	0.4483	0.7108
NSGA	0.7137	0.9927	0.8334	0.9174

In measuring the coverage of the Pareto front, we adopted the metric presented by (Deb, 2000). This metric is based on the consecutive distances among the non-dominated solutions. The non-dominated solutions are compared with a uniform distribution and the deviation is computed as follows: Given a set of non-dominated solutions P,

$$\Delta = \sum_{i=1}^{|P|} \frac{|d_i - \overline{d}|}{|P|},$$

in which, d_i is the Euclidean distance between two consecutive solutions in P in the phenotype space and $\overline{d}$ is the average of all the d_is. In order to ensure that this calculation takes into account of the spread of solutions in the entire region of the Pareto front, the boundary solutions are included in P. In our implementation, the boundary solutions are the individuals which attend minimum in at least one objective function.

The deviation measure Δ of these consecutive distances is then calculated for each run. An average of these deviations over 30 runs is calculated as the measure ($\overline{\Delta}$) for comparing different algorithms. Thus, it is clear that an algorithm having a smaller ($\overline{\Delta}$) is better, in terms of its ability to widely and evenly spread solutions in the Pareto front.

Table 2 shows the average deviation, ($\overline{\Delta}$) in all the test problems.

Table 2: Comparison of average deviation $\overline{\Delta}$ obtained using eAGA-I, eAGA-II, SPEA, and NSGA.

	$T1$	$T2$	$T3$	$T4$
eAGA-I	0.0064	0.0175	0.0093	0.0222
eAGA-II	0.0072	0.0211	0.0101	0.0338
SPEA	0.0069	0.0261	0.0144	0.0482
NSGA	0.0174	0.0318	0.0295	0.0765

The quantitative comparison in Tables 1 and 2 conforms with our observation in Figures 2 to 5. In all the four test problems, eAGA-I is observed to have the best performance in both the non-dominance of solutions and the coverage of the Pareto front. In T2 and T4, the results of eAGA-I can cover more than 70% of those of SPEA. Nevertheless, the results of SPEA can only cover less than 23% of the those of eAGA-I. Similar performances are yielded by eAGA-I and SPEA in T2 and T3. But the results of eAGA-I have much more even distribution along the Pareto front as shown in Table 2. As eAGA-II, in T2 and T3, eAGA-I outperforms it by covering more than 53% of its results. Meanwhile, it can only cover less than 35% of those of eAGA-I. In T1 and T4, eAGA-I still has better performance, even the superiority is not so remarkable. In measuring the coverage of the Pareto front, eAGA-I and eAGA-II outperform SPEA and NSGA in most of the cases. However, in T1 and T4, eAGA-II fails to find the *Pareto optima* in the regions near the boundaries. We believe that this failure is caused by the limitation of Def.2 which assigns less reproduction potential to the individuals in these regions. We acknowledge the existence of such limitation and will focus our attention to improve this weakness in the future work.

6 Conclusion

In this paper, we have presented an Enhanced Annealing Genetic Algorithm (eAGA) for Multi-Objective optimization problems. We have also proved its convergence. On four difficult test problems borrowed from the literatures, it is found that the proposed eAGA-I and eAGA-II outperform SPEA and NSGA — two well known multi-objective EAs in the explicit goals of the non-dominance of the solutions and the coverage of the Pareto front. With the properties of high effectiveness and superior performance, the eAGA should find increasing attention and applications in the near future.

Acknowledgments

This research is partially supported by a Hong Kong Government RGC Earmarked Grant, Ref. No. CUHK 4212/01E.

References

Carlos, A. C. (1999), A Comprehensive Survey of Evolutionary-based Multi-objective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269-308.

Deb, K., Agrawal, S., Pratap, A. & Meyarivan, T (2000), A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. *KanGAL Reports* No. 200001, Indian Institute of Technology Kanpur.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. MA: Addison-wesley.

Iosifescu, M. (1980), *Finite Markov Processes and Their Application*, New York: John Wiley & Sons.

Laarhoren, P. J. M. Van., & Aarts, E. H. L. (1989). *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers.

Rao, S. S., (1991), *Optimization Theory and Application*, New Delhi: Wiley Eastern Limited.

Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problem*. Halsted Press, John Wiley & Sons.

Schaffer, J.D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In J.J. Grefenstette (Eds.), *Genetic algorithms and their applications: Proceedings of the First International Conference on Genetic Algorithms*, 93-100.

Srinivas, N., & Deb, K. (1994). Multi-objective Optimization using Non-dominated Sorting in Genetic Algorithms. *Evolutionary Computation*. 2(3):221-248.

Vose, M. D.,& Liepins, G. E. (1991), Punctured equilibria in genetic search, *Complex system*, Vol.5:31-44.

Vose, M. D. (1995), Modeling simple genetic algorithms. *Evolutionary computation*, 33(4):453–472.

Zitzler, E., & Thiele, L. (1998), Multi-objective Optimization Using Evolutionary Algorithm — A Comparative Case Study. *Parallel Problem Solving from Nature V*:292-301, Netherlands: Springer.

Zitzler, E., & Thiele, L. (1999), Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *Evolutionary Computation*, 3(4): 257-271.