Parallel Genetic Algorithms Based on Coevolution

Dana Vrajitoru
IUSB, Math & Computer Science Department
1700 Mishawaka Ave, P.O. Box 7111
South Bend, IN 46634, USA
danav@cs.iusb.edu
Phone: +1-219-237-4525

Abstract

Most of the parallel implementations of genetic algorithms divide the population into several nests or niches that evolve independently and exchange information periodically. In this paper we propose a different approach, where instead of the population, we split the problem to be solved among the processes. The new model has shown an interesting performance concerning both the speedup and the quality of the solution.

1 INTRODUCTION

The genetic algorithms (GAs) [Holland, 1975], [De Jong, 1975], [Goldberg, 1989] are easy to parallelize, because most of the operations they include are independent of each other. An exhaustive review of the most used models for parallel GAs can be found in [Cantú-Paz, 1998], [Gordon and Whitley, 1993].

For most parallel implementations of the GAs, several niches of individuals solving the same problem evolve independently and exchange information from time to time [E. Cantú-Paz, 1997], [Harvey and Pettey, 1999], [Sawai and Adachi, 1999]. These algorithms are designed to minimize the amount of communication between processes. The population is divided into several independent nests (deme, niches, etc.) on which the selection and crossover operations act locally. The global performance is insured by periodic migration of some individuals between subpopulations.

Although not equivalent to serial GAs, the nested GAs show interesting speedups due to their low communication rate. Recent studies concentrate on the size of the nests and on the design of the communication network so that the performance is comparable to the performance of a sequential GA [E. Cantú-Paz, 1997], [Sarma and De Jong, 1996].

In this paper we introduce a parallel model that combines the advantages of the two categories of the parallel GA previously described: the interesting speedup from the nested models, and the good performance of the sequential models. In our scheme, each niche contains individuals solving a different part of the problem by coevolution. These parts can be combined to solve the entire problem.

To achieve this goal, we divide each individual into subindividuals that will evolve separately. Each process works with the entire population, but only on the subindividuals corresponding to particular locations. The difficulty of this model is represented by the evaluation of the subindividuals, for which the processes exchange information periodically in a similar way to the nested algorithms.

The paper is structured in the following way: Section 3 introduces the experimental context and test problems we have used. Section 2 describes our parallel model for GAs, and Section 4 presents the experimental results considering both the fitness performance and the speedup.

2 COEVOLUTION BASED PARALLEL MODEL

In the classical approach of the nested parallel GAs, each process solves the same problem and constructs a niche of whole individuals. We propose a different approach, where each process solves its own part of the problem by building a whole population of partial individuals. This model is inspired from coevolutionary models, [Potter and De Jong, 2000], [Rosin and Belew, 1997] but is different from them by the fact that it can be applied to any problem, even those that are not implicitly divisible.

2.1 MODEL DESCRIPTION

Let L be the size of the individual, and np the number of processes. In our model, the process number i is evolving the genes corresponding to the places from $i * L/np$ to $(i+1) * L/np - 1$, where both the process numbers and the gene positions start from 0. Each process works with the same population size as that of an equivalent sequential GA, so that the total number of genes produced in a generation is the same.

If we represent a generation as a matrix where each line is an individual, the nested parallel approaches divide this matrix in horizontal blocks, while our model divides it in vertical blocks.

The main challenge of this model is represented by the fitness evaluation of the partial individuals. We propose an evaluation scheme based on the schemata theory.

A *schema* is a class of individuals having part of the genes in common. We denote a schema as an individual for which the genes can take an extra value, generally marked by $*$, which represents the genes that can take any value in all of the individuals in the class. For example, for binary individuals of length 5, the schema denoted by $10**1$ represents the class composed by $\{10001, 10011, 10101, 10111\}$.

According to the schemata theorem [Goldberg, 1989], we can define the expected fitness of a schema as the average fitness of all the individuals in the class. The theorem states that the chances of survival of a schema in the next generation are directly influenced by its expected fitness, among other factors.

To evaluate partial individuals, we assimilate them with schemata. Thus, we estimate their fitness by selecting a number of individuals belonging to their class and by computing the average of their fitness values.

For example, suppose that the process number 0 computes the first two genes in an individual of size 6. To evaluate the partial individual that we represent as the schema $10****$, we select 2 (or more) individuals in this class and compute the average of their fitness. In our case, the two individuals could be 101101 and 100110. These individuals represent the intersection between the given schema and the schemata $**1101$ and $**0110$ respectively. We denote these by *complementary schemata*. If f is the fitness function, and f_e the expected fitness of a schema, we have

$$f_e(10****) = 0.5(f(101101) + f(100110)) \qquad (1)$$

In many cases, the computation of the fitness function is the most time-consuming part of the execution of a GA. To achieve a good performance, we must carefully choose the individuals representing each schema or partial individual such that we can use a minimal number of individuals and that they give us a good estimation of the best expected fitness of the partial individuals.

The schemata or partial individuals must evolve so that their intersection is an optimal individual. For example, if the best individual is 110110, the process number 0 should produce the schema $11****$, the process number 1 the schema $**01**$, and the process number 2 the schema $****10$. So, we must not only insure that each process finds a schema that contains an optimal individual, but that the intersection (or concatenation) of the resulting optimal schemata from all the processes is an optimal individual.

For this, the exchange of information between processes is essential. In our model, we insure the global evolution of the partial individuals generated by each process to a common optimal individual, by using complementary schemata from the best partial individuals found so far by all the other processes. In the beginning, these complementary schemata are built randomly. Periodically, each process exchanges with all the others the best partial individuals achieved so far to form their complementary schemata. This way, each process is in principle capable to find an optimal solution to the problem, which should be formed by the intersection of the schema representing the best partial individual with the best complementary schema.

The communication is organized in a ring where each process receives information from the previous one and sends information to the next one. During this operation, each process sends to the next one the best partial individuals in its possession, as well as the partial individuals received from the previous process. The last process sends to the first one an entire individual except for the part that the first individual owns. Thus, for each complementary schema, the average amount of information sent and received by each process is equal to

$$\frac{L}{2}\frac{(np-1)(np+2)}{np^2} \sim \frac{L}{2}$$

where L is the size of a complete individual, and np is the number of processes. We can conclude that each complementary schema requires about half an individual to be sent and received by each process.

We have performed experiences with 1 and 2 complementary schemata, and with a communication step of 50 and 100 generations. The results suggest that these small numbers can already be sufficient to achieve a performance comparable to the sequential GA.

In some cases, the choice of the number of processes and of how to divide each problem imposes itself in a natural way. Although our model allows us to split the individuals any way we choose, some particular choice may be an advantage according to the nature of each problem.

2.2 IMPROVING THE EVALUATION TIME

The main drawback of our model is the evaluation time. For each process, the global amount of computation required by the evaluation compared to a sequential algorithm is multiplied by the number of complementary schemata. For example, in Equation 1 we had to evaluate two individuals in order to estimate the fitness of the schema.

We can notice that the evaluation of a partial individual such as we have described it has to repeat an important amount of computations for each new individual to evaluate. The complementary schemata are constant in the niche between each two phases of process communication, and they represent a large part of the individual. This information could be used to improve the evaluation speed. Although there is no general method to do it, most of the problems can be represented in such a way that an improvement is possible.

For example, if parts of the individual can be evaluated independently, as it is the case for functions depending on several variables, then they can be assigned to different processes. Thus, the values obtained by evaluating the complementary schemata can be recorded and used between each two communication steps. This can explain one case in Section 4.2 where the execution time is lower than the sequential user time divided by the number of processes, which is the ideal case.

An interesting case is represented by the SAT problems: given a Boolean expression depending on some variables, find an assignment to those variables such that expression is evaluated to true. This problem requires a complex evaluation scheme for our model to be successfully applied to it.

Let S be a SAT instance, represented as a Boolean tree, and Ind a potential solution for S. Each node leaf in S is a variable whose value is given by a gene belonging to Ind. In our case, Ind is composed by $PInd$, the partial individual computed by the current process, and $CInd$, the complementary schema that is periodically communicated between processes. We propose to replace $CInd$ by a Boolean tree CS where each subtree that can be evaluated using only genes belonging to $CInd$ is replaced by the real value to which it is evaluated.

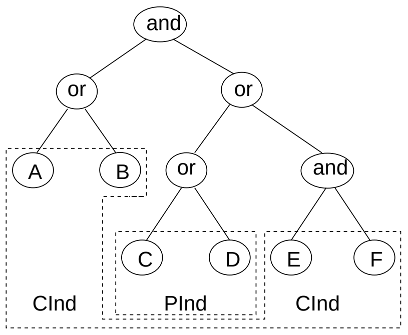

Figure 1: SAT instance

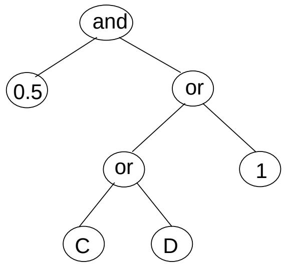

Figure 2: Complementary schema tree

For example, if S is the tree in Figure 1, $PInd$ contains the genes for variables C and D, and the complementary schema corresponding to the variables A, B, E, and F is $10**11$, then CS would be the tree in Figure 2.

This procedure reduces the evaluation time to the part of the Boolean tree that has genes belonging to $PInd$ and can be applied to many problems where the solution has a tree kind of representation.

3 EXPERIMENT DESCRIPTION

This section presents the test functions and the parameter settings that we have used for the experiment.

3.1 TEST PROBLEMS

To test and benchmark the new model, 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, and a combination of them can give us a better idea of the performance of each operator.

3.1.1 Standard functions set

We have started our experiments with the set of 10 standard functions used in many cases to test GAs [Whitley et al., 1996]. Generally, we must minimize a function $F_{1\dots10}(x_1, x_2, \dots) : R \rightarrow R$ in a given interval.

For each function, we have chosen the genetic representation of the variables x_i such that the optimal individual is neither fully composed of 0 values, nor of 1 values. Each problem is based on 2 to 30 independent variables, for which we allocate an equal number of genes. For these problems, we have chosen the number of processes such that each variable is computed by one process. Thus, the number of processes is equal to a factor of the number of variables.

3.1.2 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 et al., 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 certain number of 3-bit functions. These problems are a slight variation based on [Deb and Goldberg, 1994]. 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, and the optimal individual shows a performance of 3000.

For these problems, the individual can be divided among the processes in any way we want, since each group of 3 genes can evolve independently of all the others. In this case, we have performed experiences with 4 processes.

3.1.3 Hamiltonian circuit (HC)

Given an oriented graph, does there exist a circuit that passes once and only once through each node? This problem is known to be NP-complete [Brassard and Bratley, 1994], and is equivalent to a traveling salesman problem where all the arcs would have a weight of 1.

We have performed our experiences with 10 HC problems with graphs of 9 to 150 nodes and up to 3000 arcs. The direct representation of a HC problem for the GAs is difficult. De Jong and Spears [De Jong and Spears, 1989] suggest to transform the HC instances into SAT instances, that are easier to represent in a genetic form.

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 arc, and is assigned the true value if the arc belongs to the circuit. The SAT expression represents the fact that, for each node, one and only one of the entering arcs and of the exiting arcs must belong to the circuit.

The genetic representation of SAT is straightforward, each variable being converted into a binary gene where the 0/1 values can be interpreted as false/true. In the classical evaluation of a Boolean expression, an individual can only be evaluated to false or true. Thus, as long as an individual does not represent an exact solution to the problem, it is evaluated to 0. This makes it difficult for the GA to improve the individual performance, because it cannot decide whether an individual is far from or close to the researched solution. To evaluate an expression to more than true or false, we have used fuzzy logic measures, also proposed by De Jong and Spears [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.

In the case of the Hamiltonian circuit, or of any SAT problem in general, the evolution of a schema is very much dependent on the choice of the complementary schemata. These problems are the most challenging for our model as the dependency between the information belonging to each process is much more important than for the other problems. As we have discussed it in Section 2.2, we can improve the evaluation time by assigning entire subtrees of the SAT expression to each process, which is easier if the variables are ordered according to their places in the tree. The potential for improvement increases with the size of the individual, which can be observed in Section 4.2. We have also used 4 processes in this case.

3.2 PARAMETER SETTINGS

We have performed for each problem 40 runs of the GA, half of which without mutation, and the other 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].

We have based our evaluation on the best fitness value achieved in the last generation, considered as an average over 40 runs. We have defined the score of each scheme as its number of occurrences on the top first position in the classification provided by this measure.

3.3 THE CROSSOVER OPERATOR

We have used a crossover operator called *combined balanced* that combines four variations of the crossover in each generation : the 1-point, 2-point, uniform and dissociated. For each operation, one of the four crossover forms is used by a random choice giving each of them equal chances. We will briefly recall the functionality of these operators.

Let L be the length of the individual.

The *1-point* crossover [Holland, 1975] cuts each parent at a random cross site from 1 through $L-1$, and swaps the resulting right hand sides of the parents.

The *n-point* crossover [De Jong, 1975] is equivalent to n independent 1-point crossovers applied in sequence to the same parents. For our experiments, we have chosen $n = 2$.

The *uniform* crossover [Syswerda, 1989] swaps each of the parent genes with a probability $p_{swap} \leq 0.5$ independently of each other. We have chosen $p_swap = 0.5$ for our research, which means that about $L/2$ of the parent genes will be randomly exchanged.

The *dissociated* crossover [Vrajitoru, 1999] splits each parent in two at a different cross site, and swaps the resulting right hand sides of the parents by applying the logical conjunction and disjunction respectively on the parent genes in between the cross sites.

4 EXPERIMENTAL RESULTS

Generally, the main goal of parallel algorithms is to provide the same performance as sequential ones, but in less time. In this section we will show that our model achieves both these goals.

4.1 ACHIEVED PERFORMANCE

We have performed some experiences with 2 complementary schemata exchanged each 50 and 100 generations, and with 1 complementary schema exchanged

Table 1: Results on the standard test functions

Problem	Seq	P2x50	P1x50	P2x100
F1	0.139	0.003	0.003	0.003
F2	0.428	13.723	13.644	21.418
F3	11.825	11.375	11.4	11.325
F4	3.817	1.042	1.47	0.909
F5	4.033	2.183	2.619	2.183
F6	3.748	1.272	1.249	1.272
F7	656.477	641.226	641.226	641.226
F8	2.586	0.732	0.863	0.756
F9	0.121	0.122	0.139	0.199
F10	1.507	1.249	1.248	1.252

Table 2: Results on deception problems

Problem	Seq	P2x50	P1x50	P2x100
d1	2678.95	2792.4	2803.6	2792.4
d2	2848.45	2917	2914	2917
d3	2499.6	2766.4	2768.8	2766.4
d4	2514.15	2761.8	2762.4	2761.8
d5	2766.2	2857.6	2863.2	2857.6
d6	2764.85	2811.8	2925	2863.8
d7	2647.9	2791.4	2798.8	2791.4
d8	2669.75	2799.2	2800.8	2799.2

every 50 generations, denoted by P2x50, P2x100 and P1x50 respectively. We compare our model with a sequential algorithm (Seq) for which the population size and the size of the individual are such that the total number of generated genes during each generation is the same. The performance measure we have chosen is the average over 40 runs of the best fitness obtained in 500 generations.

Tables 1, 2, and 3 present the performance of each scheme on the set of standard functions, on the deception problems, and on the HC problems respectively.

From Table 1 we can notice that the average fitness of the parallel schemes is in general lower than that of the sequential one, which is a positive fact since these are minimization problems. The contrary observation can be made about Tables 2 and 3, that contain results on optimization problems. To be more precise in these observations, we have computed the general score of each scheme as the number of problems for which it has shown the best performance.

The scores presented in Table 4 show that both schemes where the information is exchanged every 50 generations are better than the sequential model. As

Table 3: Results on HC problems

Problem	Seq	P2x50	P1x50	P2x100
hc9	0.967	0.92	0.934	0.922
hc10	0.964	0.937	0.909	0.933
hc11	0.967	0.94	0.942	0.932
hc12	0.958	0.945	0.939	0.943
hc13	0.959	0.907	0.948	0.941
hc14	0.958	0.954	0.952	0.952
hc15	0.959	0.956	0.954	0.951
hc20	0.957	0.965	0.962	0.952
hc25	0.951	0.962	0.96	0.947
hc30	0.942	0.961	0.956	0.949
hc50	0.935	0.951	0.949	0.942
hc60	0.937	0.948	0.945	0.942
hc70	0.936	0.947	0.945	0.941
hc80	0.936	0.947	0.944	0.941
hc90	0.94	0.949	0.947	0.947
hc100	0.939	0.95	0.951	0.949
hc110	0.939	0.955	0.952	0.953
hc120	0.939	0.951	0.952	0.949
hc130	0.936	0.951	0.95	0.95
hc140	0.933	0.95	0.951	0.95
hc150	0.938	0.943	0.942	0.942

Table 4: Summary of results

	F1-10	deception	HC	Total
Seq	1	0	7	8
P2x50	2	1	11	14
P1x50	3	7	3	13
P2x100	4	0	0	4

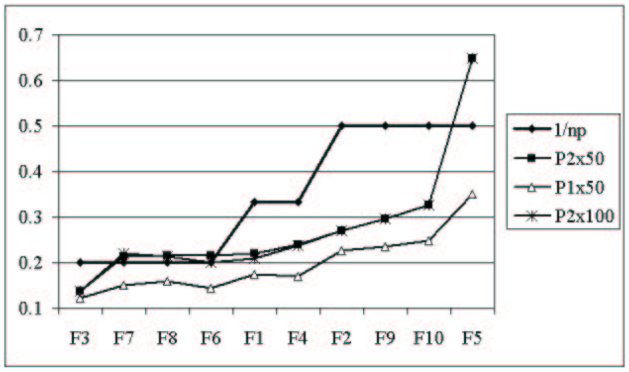

Figure 3: Execution time on the set of standard functions

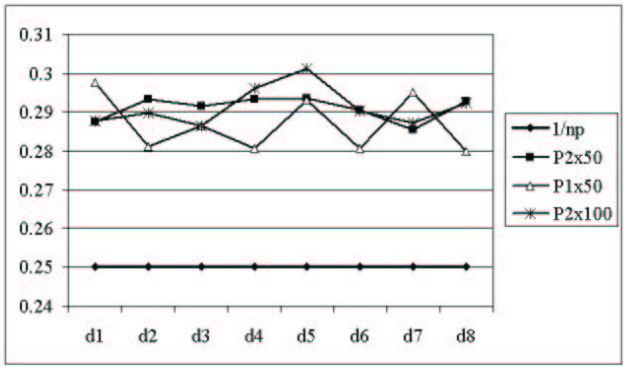

Figure 4: Execution time on the deception problems

expected, the use of two complementary schemata for evaluation proves to be more efficient than the use of only one, but not by much. We can conclude that the first goal of our parallel GA implementation, which is to perform at least as well as the sequential one, is accomplished.

4.2 SPEEDUP

The second performance measure for a parallel implementation is the speedup. We have performed our experiences on a SUN workstation with 4 processors using the MPI library for communication. Figures 3, 4, and 5 show the normalized user time which is the execution time from the user's point of view divided by the sequential execution time. According to this measure, the normalized execution time should not be superior to 1.0, and should be equal to one over the number of processes ($1/np$) in the ideal case, which is also plotted on these figures.

From Figures 3, 4, and 5 we can note that the speedup is influenced by the difficulty of the problem measured both by the complexity of the evaluation and by the

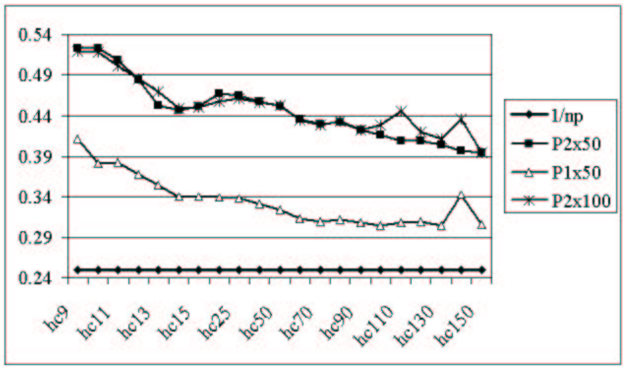

Figure 5: Execution time on the HC problems

length of the individual.

Thus, the set of standard functions represents an easy class of problems both concerning the evaluation complexity, which is linear, and the length of the individual, which varies from 20 to 300 in our case. In this case, the parallel user time goes even below the ideal curve, which is better than expected.

The deception problems are easy to evaluate (linear complexity), but the length of the individual is higher than before. For these problems we have achieved a user time no more than 0.06% higher than the ideal case.

The HC problems are hard from both points of view. Following the P2x50 line in Figure 5, which represents the scheme that gave the best results in this case, we can notice that the speedup increases with the problem size, which is interesting because the large problems are more likely to create a need for a parallel implementation. For these problems we have achieved a user time varying between 40% and about 50% of the sequential user time.

From the above observations we can conclude that our parallel implementation of GAs is also interesting from the point of view of the speedup.

5 CONCLUSIONS AND FUTURE WORK

The goal of this paper has been to introduce a new parallel implementation of the GAs and to show that it can compare to a sequential GA concerning the quality of the results and that it reduces the speedup considerably. This model has been presented in Section 2.

Section 4 has concentrated on the experimental results and has shown that both requirements stated before have been achieved. Thus, Tables 1 to 4 show that the quality of the evolved solutions by a parallel scheme can be better than the one resulting from an equivalent sequential GA. Moreover, Figures 3 to 5 show a significant reduction of the execution time from the user's point of view.

The present research can be extended to develop new evaluation improvement schemes for other difficult problems and to test the parallel model on other types of parallel machines or networks.

Acknowledgements

This work has been partially accomplished at the $LCVM^2$ laboratory from the EPFL, Switzerland, http//lcvmwww.epfl.ch/.

References

[Brassard and Bratley, 1994] Brassard, G. and Bratley, P. (1994). *Fundamentals of Algorithmics.* Prentice-Hall.

[Cantú-Paz, 1998] Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171. Paris: Hermes.

[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.

[E. Cantú-Paz, 1997] E. Cantú-Paz, D. E. G. (1997). Modeling idealized bounding cases of parallel genetic algorithms. In J. Koza, e. a., editor, *Proceedings of the Second Annual Conference on Genetic Programming*, pages 353–361, San Francisco. Morgan Kaufmann Publishers.

[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.

[Gordon and Whitley, 1993] Gordon, S. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In Forrest, S., editor, *Proceedings of the International Conference on Genetic Algorithms*, pages 177–. Morgan Kaufmann Publishers.

[Harvey and Pettey, 1999] Harvey, K. and Pettey, C. (1999). The outlaw method for solving multimodal functions with split ring parallel genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 274–280, Orlando (FL). Morgan Kaufmann Publishers.

[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).

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

[Rosin and Belew, 1997] Rosin, C. and Belew, R. (1997). New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29.

[Sarma and De Jong, 1996] Sarma, J. and De Jong, K. (1996). An analysis of the effects of neighborhood size and shape on local selection algorithms. In *Proceedings of Parallel Problem Solving from Nature IV*, pages 236–244.

[Sawai and Adachi, 1999] Sawai, H. and Adachi, S. (1999). Parallel distributed processing of a parameter-free ga by using hierarchical migration methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 274–280, Orlando (FL). Morgan Kaufmann Publishers.

[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.

[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.

[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.