On Decentralizing Selection Algorithms *

Kenneth De Jong
Computer Science Department
George Mason University
Fairfax, VA 22030
kdejong@gmu.edu

Jayshree Sarma
Computer Science Department
George Mason University
Fairfax, VA 22030
jsarma@gmu.edu

Abstract

The increasing availability of parallel computing architectures provides an opportunity to exploit this power as we scale up evolutionary algorithms (EAs) to solve more complex problems. To effectively exploit fine grained parallel architectures, the control structure of an EA must be decentralized. This is difficult to achieve without also changing the semantics of the selection algorithm used, which in turn generally produces changes in an EA's problem solving behavior. In this paper we analyze the implications of various decentralized selection algorithms by studying the changes they produce on the characteristics of the selection pressure they induce on the entire population. This approach has resulted in significant insight into the importance of selection variance and local elitism in designing effective distributed selection algorithms.

1 INTRODUCTION

One of the frequently stated virtues of evolutionary algorithms (EAs) is their "natural" parallelism. The increasing availability of parallel computing in both coarsely and finely grained architectures provides a tantalizing opportunity to exploit this power as we scale p EAs to solve larger and more complex classes of problems. However, most of the commonly used EAs today (i.e., the ones for which we have the most theoretical and experimental results) have fairly strong explicit and implicit forms of centralized control.

Considerable work has already been done in adapting EAs to both coarsely and finely grained parallel architectures. The most natural adaptation for coarsely grained parallelism is an "island" model in which there are a number of centralized EAs running in parallel. The focus is on designing and implementing useful "migration" mechanisms which allow exchange of information between the independently evolving local populations (Tanese 1989; Cohoon, Martin, and Richards 1991; Whitley and Starkweather 1990).

This paper is the result of our interest in adapting EAs to effectively exploit fine grained architectures. In this case the focus is on parallelizing the algorithm itself and many candidate methods for doing so have been proposed and tested (Spiessens and Manderick 1991; Collins and Jefferson 1991; McInerney 1992). To get a quick sense of the issues involved, consider the following EA pseudo-code:

1. Randomly produce an initial population of individuals and evaluate them.
2. DO until some stopping criterion is met
   (a) Select parents
   (b) Produce children
   (c) Evaluate children
   (d) Select members of the next generation
3. end DO

One can certainly map this onto a finely grained architecture directly by identifying a "master control process" which assigns various tasks such as mating and evaluation to "slave processes". However, the master-slave communication costs are generally high enough that such simple and direct implementations produce little if any performance improvements.

Consequently, the focus has been to identify ways to reduce communication overhead by reducing the role of the master process by decentralizing and distributing control to the slave processes. Unfortunately, this is difficult to achieve without changing the underlying semantics of the algorithms themselves, thus shifting the emphasis from parallelizing existing algorithms to that of inventing new ones.

*To appear in L. Eshelman, ed., Genetic Algorithms: Proceedings of the 6th International Conference (ICGA95), Morgan Kaufmann, San Francisco, CA.

The primary causes of this are the traditional selection algorithms such as fitness proportional, rank proportional, or truncation selection. In each case global calculations are required which involve high communication overhead. Most attempts to decentralize them produce different selection pressures than the centralized versions, and thus result in different evolutionary behavior.

The one exception to this is tournament selection. Since there is no need to keep any global statistics or assign ranks, it is an obvious candidate to use in decentralized evolutionary algorithms (Fogel 1994; Goldberg and Deb 1991). However, as we will see in the next section, even in this case there are subtle issues that must be dealt with.

The goal of this work is to understand at a more fundamental level the implications of various approaches to decentralizing selection so that more effective parallel EAs can be developed. In this paper we present some initial results based on analyzing the changes in the characteristics of the selection pressure induced by decentralization.

2 DECENTRALIZED SELECTION WITH GLOBAL POOLS

A selection algorithm involves two elements: a selection pool, and a selection probability distribution over that pool. In most centralized EAs, the pool is the entire population (i.e., a global pool). The selection probability distributions are typically defined in terms of one's fitness relative to the fitness of the other individuals in this pool, and require ranking or fitness averages to be maintained.

The one exception to this is tournament selection in which k individuals are randomly picked with uniform probability from the selection pool, and the one with highest fitness is declared to be the winner and selected to be a parent. It is quite straightforward to show that binary tournament selection ($k = 2$) is equivalent in expectation to the standard linear ranking scheme in which the best individual gets 2 offspring and the worst gets none (Goldberg and Deb 1991).

Hence, binary tournament selection is an attractive candidate for decentralized selection. It is easily implemented by assigning each member of the population to a separate processor. Code can be executed (in parallel) on each processor which uses binary tournament selection on the entire processor pool to select parents. The appropriate number of parents can then produce an offspring to (potentially) replace the current individual assigned to that processor.

When one implements this decentralized form of selection, two issues immediately arise. The first is that communication costs, though lower than before, are still high. This is a result of the fact that on most finely grained architectures communication costs to distant processors are significantly higher than to neighboring ones. This raises the question as to whether any form of a global selection pool can be efficiently implemented on such architectures and is addressed in more detail in the next section.

The second, and more surprising result is that, if we ignore communication costs and make comparisons via the traditional "best so far" curves, binary tournament selection appears to perform worse than the equivalent linear ranking scheme, even though they have identical selection pressure.

These observations came from an initial set of experiments using a modified version of GENESIS 5.0. The test suite consisted of De Jong's functions F3, F4, Peak problems (Peak 1, Peak 2, and Peak 6) and Hamiltonian circuit (HC) problems (HC12 and HC10). (See (De Jong and Spears 1989) for a description of these functions.) We used the standard settings for crossover (0.60) and mutation (0.001), no elitism, a population size of 100, and averaged the results over 100 independent runs. Figures 1 and 2 give typical examples of the results obtained.

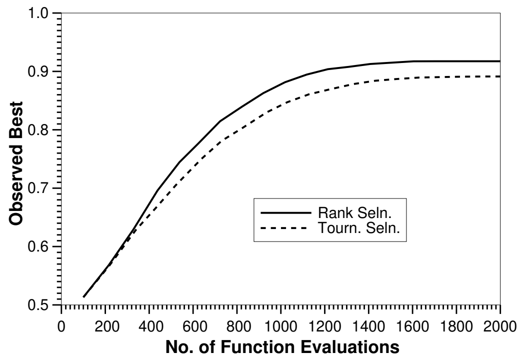

Figure 1: Best-so-far Performance Curves for function HC10 using linear rank and binary tournament selection schemes and population size 100.

To understand these results better, we analyzed these selection algorithms in more detail in the following way. For each selection method, we calculated the expected number of offspring as well as statistics on the actual number of offspring produced by each individual parent in generation 1 of a generational GA. Reliable statistics were obtained by using the same initial population and, for each selection method, monitoring the actual number of offspring produced by each member of the population. This process was repeated 100 times with different random number seeds in order to estimate the variance due to sampling error. The results are given in Figures 3-4 in which the individuals were sorted by fitness from best to worst for plotting

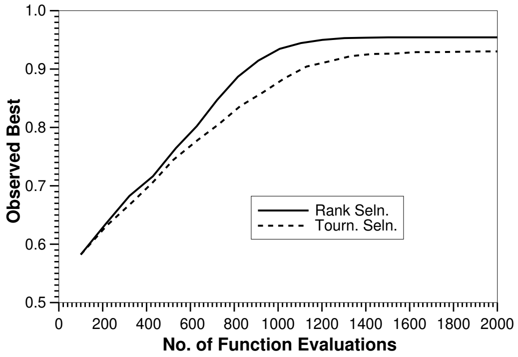

Figure 2: Best-so-far Performance Curves for function Peak6 using linear rank and binary tournament selection schemes and population size 100.

purposes. For comparative purposes we also analyzed the effects of proportional selection and included the results in Figure 5.

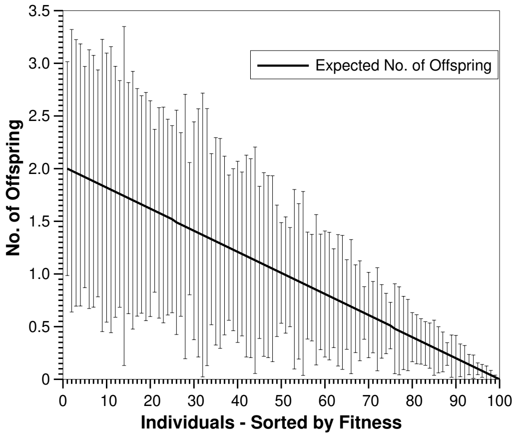

Figure 3: Variance in the expected number of offspring using binary tournament selection, population size 100 for function HC10.

The most striking observation from this analysis is the fact that, although both ranking and binary tournament selection have the same expected selection pressure, binary tournament selection exhibits considerably higher variance as illustrated in Figures 3 and 4. This is because linear ranking as well as proportional selection (Figure 5) is usually implemented as an "expected value" model which minimizes the variance due to sampling (De Jong 1975; Baker 1987). In a centralized environment, tournament selection can also be implemented as an expected value model by guaranteeing that all individuals participate in exactly k tournaments (Goldberg, Korb, and Deb 1989). However, this is difficult to implement in any efficient way in a distributed environment, resulting in much higher variance in the actual number of offspring produced.

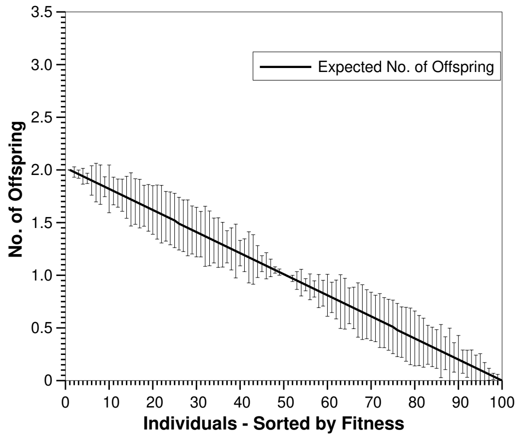

Figure 4: Variance in the expected number of offspring using linear rank selection, population size 100 for function HC10.

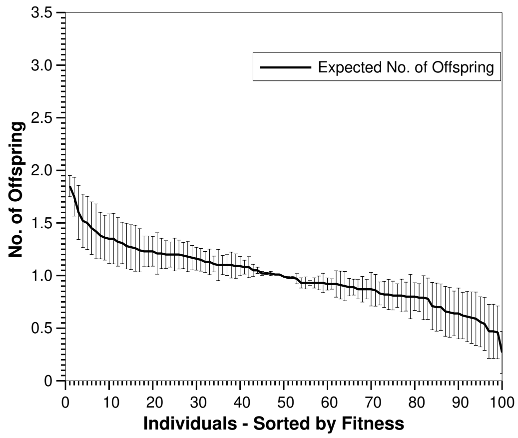

Figure 5: Variance in the expected number of offspring using proportional selection, population size 100 for function HC10.

The larger variance of the distributed form of binary tournament selection provides the most plausible explanation for why it is consistently outperformed by a linear ranking scheme which induces a selection pressure that is identical in expectation, but lower in variance. Just as in the centralized cases, increased selection variance increases genetic drift in finite populations which in turn can have negative effects on search performance (De Jong 1975). This suggests that these "anomalous" results should diminish with increasing population size, and that is what was observed when the experiments were rerun with larger population sizes (Figure 6).

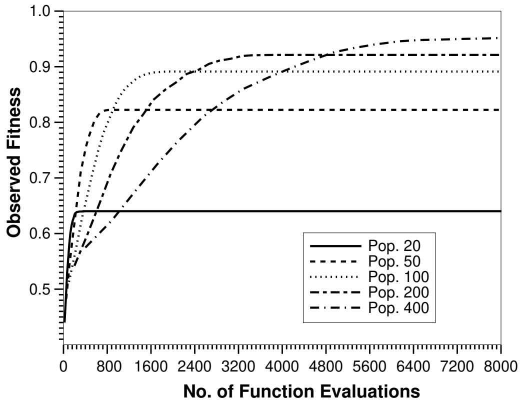

Figure 6: Best-so-far Performance Curves using binary tournament selection with different population sizes for function HC10.

Since one typically uses much larger population sizes (> 1000) on finely grained parallel architectures, the effects of the higher variance are not likely to have too much of a negative impact on search performance. However, larger populations also result in higher communication overhead when global selection pools are used. This has motivated many studies involving the use of local neighborhood selection pools in order to minimize communication overhead (Gorges-Schleuter 1989; Spiessens and Manderick 1991; McInerney 1992). This, in turn, raises the question as to what kind of selection algorithm should be used with local pools. We address this issue in the remaining sections.

3 DECENTRALIZED SELECTION WITH LOCAL POOLS

The idea of using local neighborhoods as selection pools is attractive from both a communication overhead point of view and its biological plausibility. This requires introducing some sort of distance metric and/or topology on the population so that the concept of a neighborhood can be defined. Metrics involving distance in genotype or phenotype spaces (such as sharing functions (Deb and Goldberg 1989)) generally require global statistics and are not easy to implement efficiently in a decentralized form. For finely grained parallel architectures a more natural approach is to introduce a topology in which individuals live on grid points and neighborhoods defined in terms of nearby grid points.

Various studies have proposed different topologies (Mühlenbein 1991; Collins and Jefferson 1991; Baluja 1993). In this paper we focus on one of the more popular topologies, a two-dimensional toroidal grid. In the experiments reported here the grids are square, although rectangular grids have been proposed and studied as well. Neighborhoods surrounding a particular grid point are defined in terms of the number of steps taken (up, down, left, right) from that grid point. Every grid point has a neighborhood which overlaps with the neighborhoods of nearby grid points.

To achieve full decentralization, a modified EA is run in parallel on each grid point with code to select parents from its neighborhood, produce offspring, and (possibly) replace the current individual assigned to that grid point. The overlapping neighborhoods provide an implicit mechanism for migration of genetic material around the grid. If the neighborhoods are too large, we incur the same high communication overhead discussed in the previous section. Consequently, most studies consider only small neighborhoods.

In this paper we focus on threee small neighborhoods of increasing size, the geometries of which are illustrated in Figure 7. Each such neighborhood defines a local selection pool. Our goal is to understand better the implications of choosing a particular selection algorithm for use with these small neighborhoods. In this paper we analyze three alternatives: proportional, ranking, and binary tournament selection.

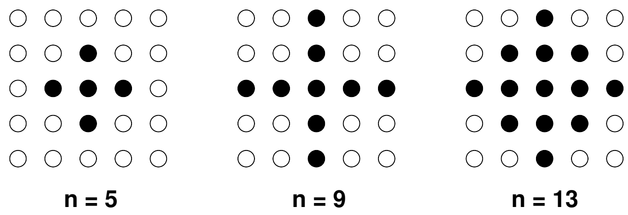

Figure 7: Neighborhood Geometry

The dynamics of parallel EAs operating in overlapping neighborhoods is extremely complex and difficult to analyze formally. We provide some initial insights into the choice of local selection algorithms by analyzing them empirically on the same test suite as the one used in the previous section. Figures 8-10 illustrate the typical performance curves obtained using a 32×32 grid and a neighborhood of size 5, 9, and 13 respectively.

For all three neighborhoods, we see that the performance of proportional selection is significantly below the others. Ranking selection appears to produce the best results, although as the pool size increases, binary tournament selection produces search behavior equivalent to ranking. It was not clear to us why either of these observations should have been expected.

To understand this better, we analyzed the characteristics of the *emergent selection pressure* induced on the entire population by a local selection algorithm. As in

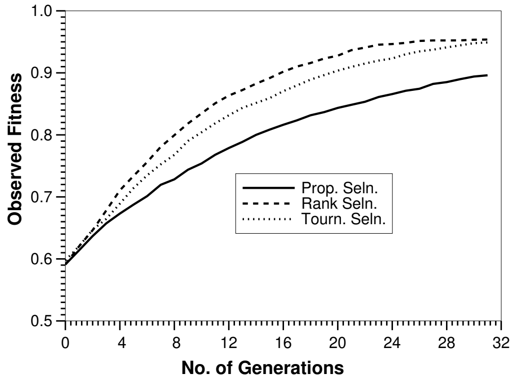

Figure 8: Best-so-far Performance Curves for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 5 for function HC10.

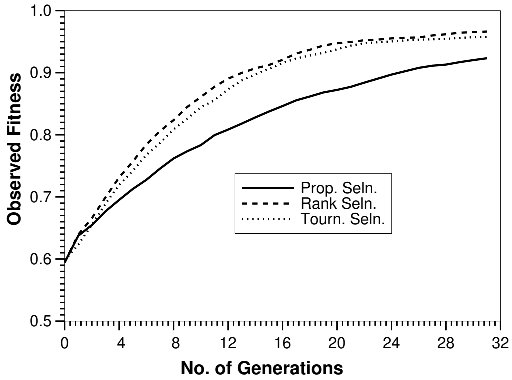

Figure 10: Best-so-far Performance Curves for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 13 for function HC10.

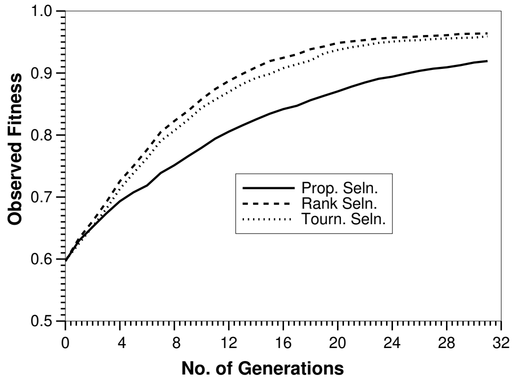

Figure 9: Best-so-far Performance Curves for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 9 for function HC10.

the previous section we use the same initial population and, for each selection method, monitored the actual number of offspring produced by each member of the population. This process was repeated 100 times with different random number seeds in order to estimate the variance due to sampling error, and the results plotted as if there was a single global pool with members sorted by fitness from best to worst. Figures 11-13 illustrate this for a 32×32 grid and a neighborhood size of 5.

Several characteristics are immediately apparent when comparing these with Figures 3-5 involving a global pool. The first observation is that, under local ranking and binary tournament selection, members with similar fitness can have quite different expected number of offspring as indicated by the rather jagged expected value lines. By contrast, local proportional selection induces a much more uniform but weaker selection pressure than either local ranking or local binary tournament selection.

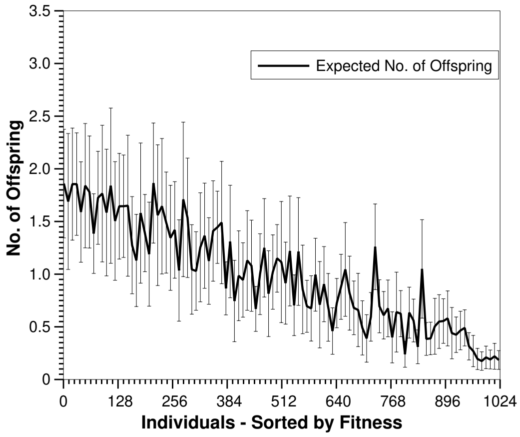

Figure 11: Variance in the expected number of offspring using binary tournament local selection, a 32×32 toroidal grid and a neighborhood size of 5 for function HC10.

With a little thought one can see that this is due to the effects of individuals having to compete only in small local neighborhoods. We know that ranking and binary tournament selection produce a constant selection pressure that is independent of the actual fitness values, while proportional selection is quite sensitive to them. This insensitivity is a virtue in the case of these small local neighborhoods and results in an induced selection pressure on the whole population which is roughly equivalent to the centralized case.

In addition to insight into local selection methods, these results also provide some initial hints about the role of neighborhood size. Notice that there is a performance improvement in the case of all the three selection schemes when neighborhood size is increased from 5 to 9 (Figures 8 and 9). The improvement seen when the neighborhood size is increased from 9 to 13 is quite negligible, suggesting that additional performance improvements are unlikely to be obtained via larger neighborhoods sizes even if we ignore communication costs.

Figures 8-11 also suggest another interesting tradeoff. Note that the performance of local tournament selection is very similar to linear ranking selection and better than proportional selection in the larger neighborhoods. Hence, by increasing the neighborhood size (and communication costs) we can obtain the performance of local ranking selection without the overhead of ranking.

Is it possible to improve search performance without increased overhead? This analysis suggests a likely approach: combine local tournament selection on small neighborhoods with an elitist policy which replaces the individual assigned to a grid point by an offspring only if it has higher fitness. The effect is to strengthen local selection differentials which, as we have seen, results in better global search performance. This is illustrated in Figure 14. Using binary tournament with local elitism produces behavior similar to linear ranking when a neighborhood size of 5 is used.

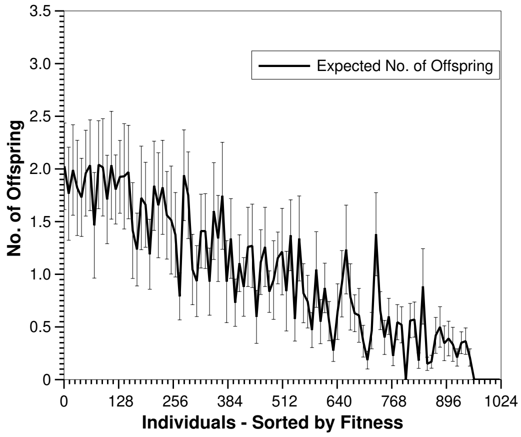

Figure 12: Variance in the expected number of offspring using linear rank local selection, a 32×32 toroidal grid and a neighborhood size of 5 for function HC10.

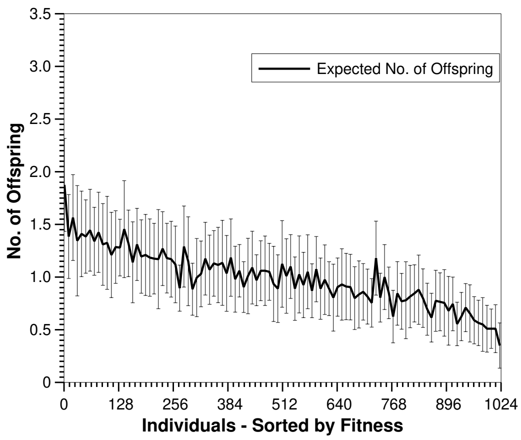

Figure 13: Variance in the expected number of offspring using proportional local selection, a 32×32 toroidal grid and a neighborhood size of 5 for function HC10.

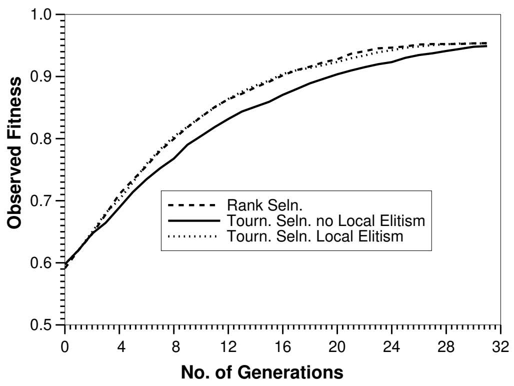

Figure 14: Best-so-far Performance Curves for linear rank and binary tournament (with and without local elitism) local selection schemes using a 32×32 toroidal grid, neighborhood size of 5, for function HC10

4 CONCLUSIONS AND FURTHER WORK

The results presented here concerning effective decentralization of selection algorithms in order to exploit finely grained parallel architectures are clearly preliminary, but are already providing some important insights.

These results emphasize the importance of an analysis of the variance of selection schemes. Without it one can fall into the trap of assuming that selection algorithms that have equivalent expected selection pressure produce similar search behavior. Since higher variance is generally more strongly correlated with poor search performance when small population sizes are involved, to improve performance one must reduce selection variance and/or increase population size.

The analysis of the three local selection schemes pointed out the need for stronger local selection pressure to induce appropriate global selection pressure. Of the three local selection schemes studied, binary tournament selection appears to have the most desirable properties from both a global search perspective and communication overhead. In addition, the insights gained from studying the global selection distributions induced by the local selection algorithms show how an elitist policy can be combined with the binary tournament selection to improve performance and reduce communication costs.

In addition to completing the preliminary results presented here, we are currently exploring two new directions: understanding the effects of softening the local deterministic elitist policy to a probabilistic one, and studying the effects of other topologies.

References

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA, pp. 14–21. Lawrence Erlbaum Associates.

Baluja, S. (1993). Structure and performance of fine-grained parallelism in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana–Champaign, IL, pp. 155–162. Morgan Kaufmann.

Cohoon, J. P., W. N. Martin, and D. S. Richards (1991). A multi–population genetic algorithms for solving the k-partition problem on hypercubes. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 244–248. Morgan Kaufmann.

Collins, R. J. and D. R. Jefferson (1991). Selection in massively parallel genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 249–256. Morgan Kaufmann.

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

De Jong, K. A. and W. M. Spears (1989). Using genetic algorithms to solve NP-Complete problems. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 124–132. Morgan Kaufmann.

Deb, K. and D. E. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 42–50. Morgan Kaufmann.

Fogel, D. B. (1994). Evolutionary programming: an introduction and some current directions. *Statistics and Computing 4*, 113–129.

Goldberg, D. E. and K. Deb (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, San Mateo, CA, pp. 69–93. Morgan Kaufmann.

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

Gorges-Schleuter, M. (1989). ASPARGOS an asynchronous parallel genetic optimization strategy. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 422–427. Morgan Kaufmann.

McInerney, J. (1992). *Biologically Influenced Algorithms and Parallelism in Non–linear Optimization.* Ph. D. thesis, University of California, San Diego.

Mühlenbein, H. (1991). Evolution in time and space – the parallel genetic algorithm. In *Foundations of Genetic Algorithm*, San Mateo, CA, pp. 316–337. Morgan Kaufmann.

Spiessens, P. and B. Manderick (1991). A massively parallel genetic algorithm: Implementation and first analysis. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 279–287. Morgan Kaufmann.

Tanese, R. (1989). Distributed genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 434–439. Morgan Kaufmann.

Whitley, D. and T. Starkweather (1990). Genitor II: A distributed genetic algorithm. *Journal Expt. Theor. Artificial Intelligence 2*, 189–214.