Determining the Best Parent Selection Method for a Genetic Algorithm through Varying Problem Sizes and Complexities

Daman Oberoi[1], Bart Rylander[1]

[1] School of Engineering, University of Portland,
5000 N. Willamette Blvd. Portland, OR 97203, U.S.A.
{doberoi, rylander }@up.edu

Abstract. We conduct an experiment to investigate which of three parent selection methods (elitism, fitness proportionate, and tournament) generates the greatest fitness in the fewest number of generations using a genetic algorithm (GA). The parent selection methods were applied to the problems of Maximum Ones, 3-Processor Scheduling, and Sorting, while in each case the problem size varied from 4 to 22. We show that for nearly all problem sizes and types, Tournament selection produces the best results.

1 Introduction

The purpose of this research is to identify a parent selection algorithm independent of problem size, type, and complexity that produces the greatest fitness in the fewest number of generations. Typically a practitioner will use an algorithm with which he is most familiar, but it may not produce the best results. The hope of this experiment is to declare a universally superior parent selection method.

It is important to distinguish the difference between a problem and a problem instance. A problem is a mapping from problem instances onto solutions. For example, the GA is applied to three types of problems in this experiment: Maximum Ones, 3-processor scheduling, and Sorting. These are merely three types of problems to which a GA can be applied. A problem instance is a particular version of a problem with a specific set of parameters. This experiment is being performed on 19 instances of each type of problem, where the varying parameter is the number of tasks (from 4 to 22). Problem instance and problem size are used interchangeably throughout this paper.

2 Test Parameters

The GA was applied to three types of problems: Maximum Ones, 3-Processor Scheduling, and Sorting. Maximum Ones is a problem in which the population achieves optimal fitness when all alleles are 1. The 3-Processor Scheduling problem assigns a scheduling duration to each task for each processor. In this experiment, the schedule values were arbitrarily assigned for the first four tasks. All schedule values were repeated after four tasks. Sorting is a problem in which the optimal population contains integers represented in binary in a pre-determined order (ascending order for

this research). For this experiment, the binary values were matched bit-by-bit to the pre-determined order (as opposed to matching a set of binary values representing one integer to the pre-determined order).

While the number of tasks varied for each problem type from 4 to 22, the following parameters remained constant throughout the experiment: 100 chromosomes, at least 90% convergence, survival of the fittest 50% of chromosomes, and 1% mutation rate. The 90% convergence requirement means the algorithm exited its loop when the fitness was within 90% of the optimal value. The survival rate indicates that the best 50% of chromosomes according to fitness carried on to the next generation. A bit was mutated from 0 to 1 or vice versa 1% of the time for all newly created chromosomes in a generation.

Crossover was performed on every generation with parents selected from the top 50% of chromosomes according to greatest fitness. Each crossover event produced one chromosome of the next generation for each selected pair of parents. The crossover point, where the influence of one parent ended and the other parent began, was randomly chosen between 0 and one less than the problem size.

The method of selecting parents was the point under investigation. One hundred trials were conducted for each selection method given the problem size and problem type; hence, 17,100 trials were conducted all together (19 problem sizes x 3 selection methods x 3 problem types x 100 trials = 17,100 trials). Elitism selection refers to the method by which two parents, not necessarily distinct, per crossover event were randomly selected from the top 50% of chromosomes. Fitness Proportionate selection refers to the process by which two parents, not necessarily distinct, per crossover event were randomly selected from the top 50% of chromosomes, but the randomization was weighted proportionately according to the fitness of each chromosome. Tournament selection refers to the method in which two parents, not necessarily distinct, per crossover event were randomly selected from the top 50% of chromosomes through a “tournament face-off.” This face-off involves selection of three distinct parents from the top 50% of parental chromosomes through which the chromosome with the greatest fitness emerges as the parent.

3 Results

Upon completion of the experiment, the results were tabulated and grouped by problem type. Although data was collected on a variety of parameters (including minimum, maximum, median, and average values for fitness, generations, and mutation), we have chosen to report only on median values for fitness and generations because they seem most representative of the overall results.

3.1 Maximum Ones Problem

Fitness. For the Maximum Ones problem, the higher the fitness, the greater the performance. Therefore, Figure 1 shows that Elitism performs better than Fitness Proportionate roughly half of the time (9 instances). However, the amount by which Elitism performs better is much more consistent than the amount by which Fitness Proportionate performs better. There does not appear to be any pattern regarding the problem size in which one selection method performs better than the other.

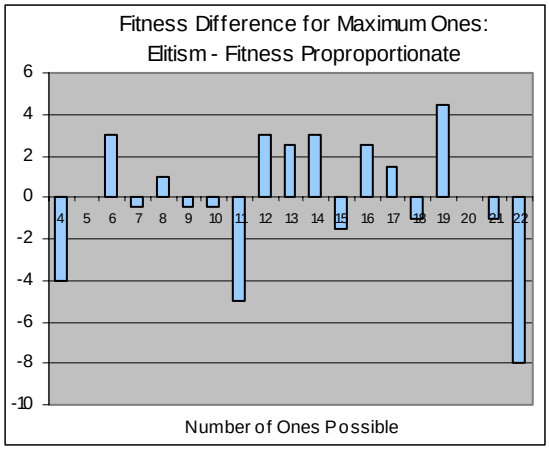

Fig. 1. Comparison of Elitism to Fitness Proportionate regarding their effects on fitness for Maximum Ones

Figure 2 and Figure 3 show that, with one exception, Tournament selection always performs better than Elitism and Fitness Proportionate. Interestingly, there appears to be some correlation between the rising and falling trends between the two graphs, with a notable dip at 14. Also, the minimum on both figures occurs at 4 and the maximum occurs at 18.

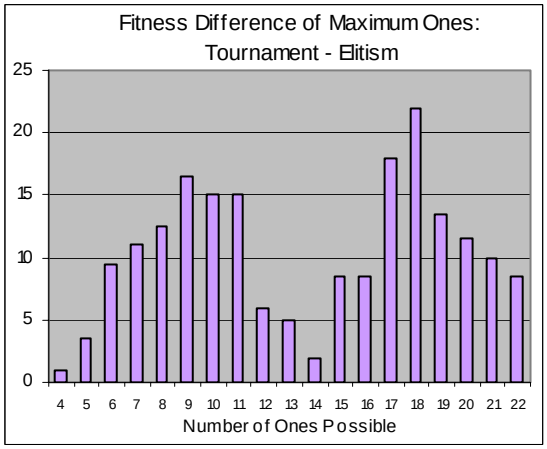

Fig. 2. Comparison of Tournament to Elitism regarding their effects on fitness for Maximum Ones

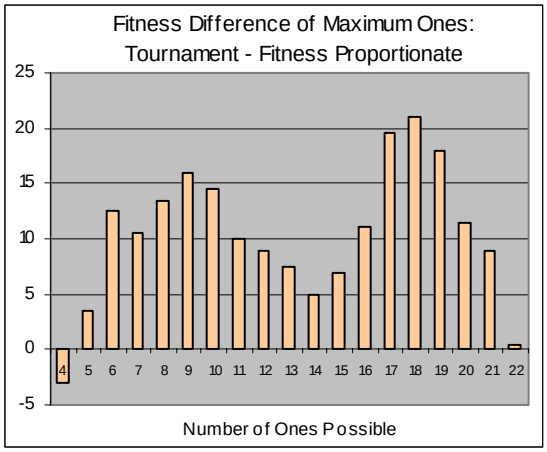

Fig. 3. Comparison of Tournament to Fitness Proportionate regarding their effects on fitness for Maximum Ones

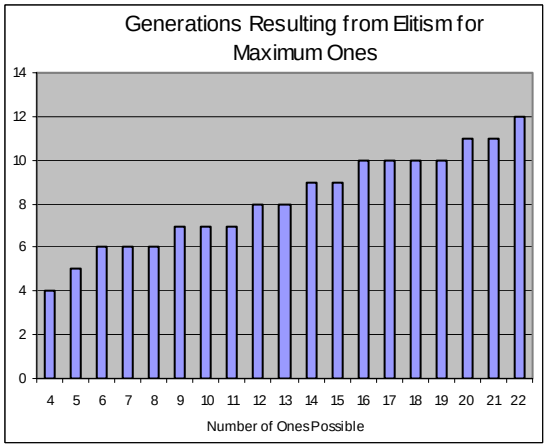

Fig. 4. The number of generations resulting from Elitism applied to Maximum Ones

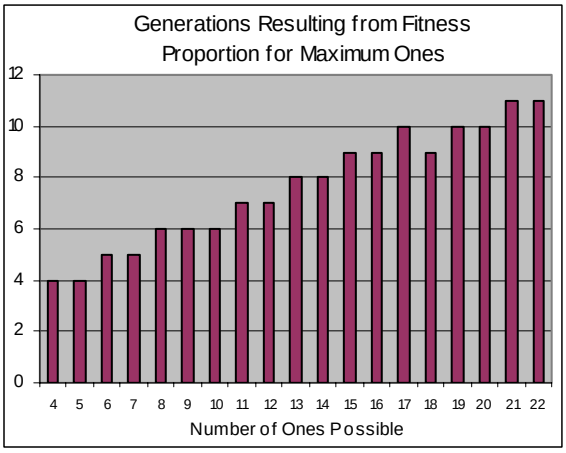

Fig. 5. The number of generations resulting from Fitness Proportionate applied to Maximum Ones

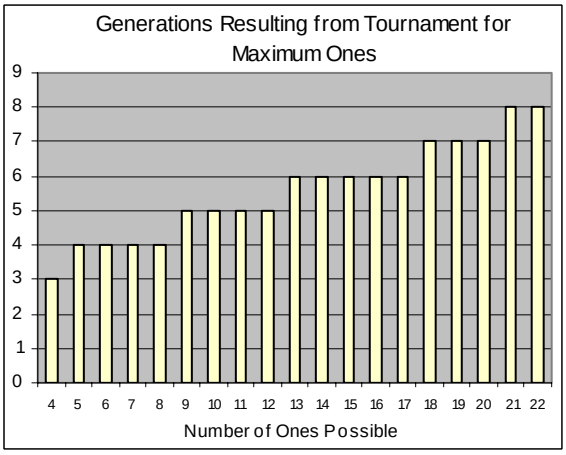

Fig. 6. The number of generations resulting from Tournament applied to Maximum Ones

3.2 3-Processor Scheduling

Fitness. For 3-Processor Scheduling, a smaller fitness value suggests a stronger performance because the goal is to minimize the time required to complete the set of tasks by each processor. Accordingly, Figure 7 shows that Elitism and Fitness Proportionate outperform each other approximately half of the time. However, with a couple of exceptions, Elitism typically outperforms Fitness Proportionate by a greater amount than Fitness Proportionate outperforms Elitism.

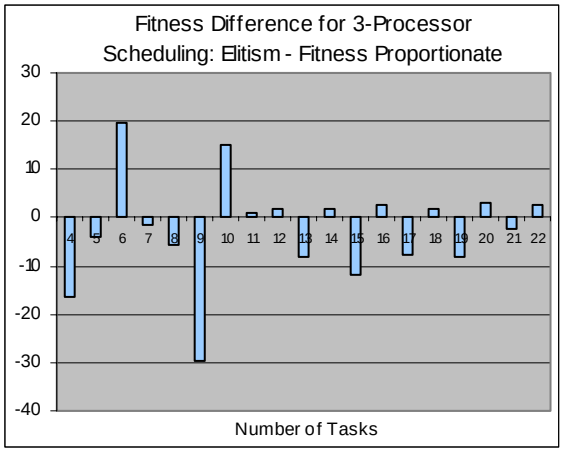

Fig. 7. Comparison of Elitism to Fitness Proportionate regarding their effects on fitness for 3-Processor Scheduling

Out of the 38 charted instances between Tournament selection and the other two selection schemes, Tournament performs better in all but 10 scenarios. Moreover, the amounts by which Tournament typically performs better is significantly greater than the amounts by which it does not. See Figure 8 and Figure 9.

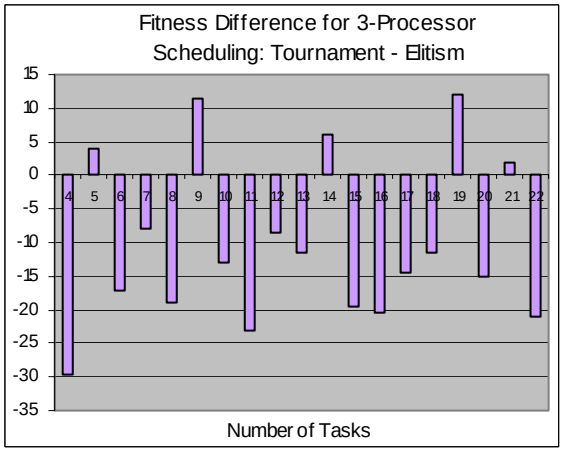

Fig. 8. Comparison of Tournament to Elitism regarding their effects on fitness for 3-Processor Scheduling

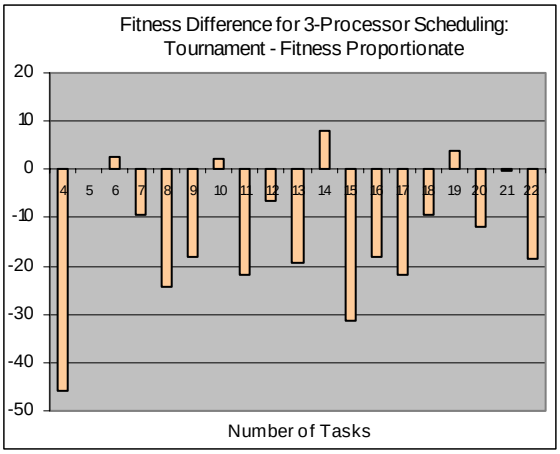

Fig. 9. Comparison of Tournament to Fitness Proportionate regarding their effects on fitness for 3-Processor Scheduling

Generations. For every problem size, Tournament selection resulted in the fewest number of generations, while Elitism produced the second fewest. There were no instances in which the number of generations were equal for any of the selection methods. See Figure 10, Figure 11 and Figure 12.

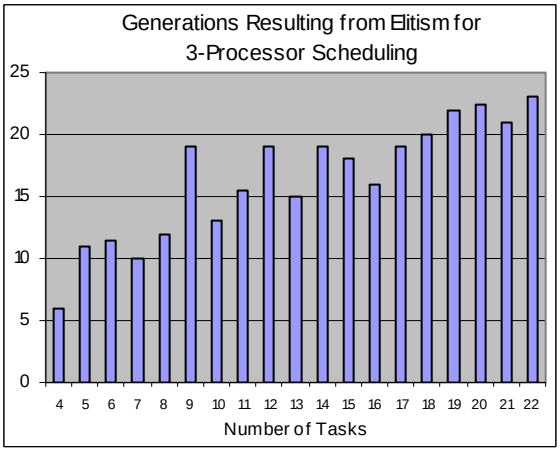

Fig. 10. The number of generations resulting from Elitism applied to 3-Processor Scheduling

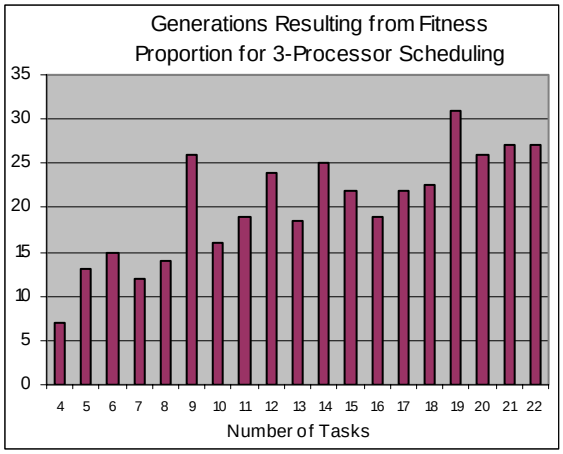

Fig. 11. The number of generations resulting from Fitness Proportionate applied to 3-Processor Scheduling

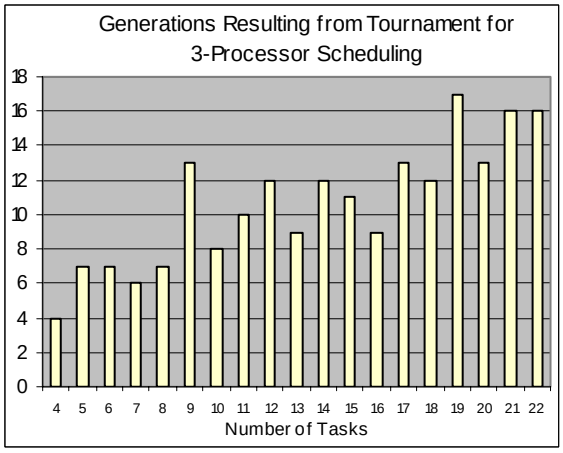

Fig. 12. The number of generations resulting from Tournament applied to 3-Processor Scheduling

3.3 Sorting Problem

Fitness. Since a higher fitness value is beneficial for the Sorting problem, Fitness Proportionate had a slight edge in the number of times it performed better than Elitism. However, the amounts by which Fitness Proportionate performs better are not very large in most instances. Moreover, the amounts by which Elitism and Fitness Proportionate outperform each other are pretty similar for most problem sizes.

Interestingly, there are groupings of problem instances where Fitness Proportionate performs better and where Elitism performs better. Refer to Figure 13.

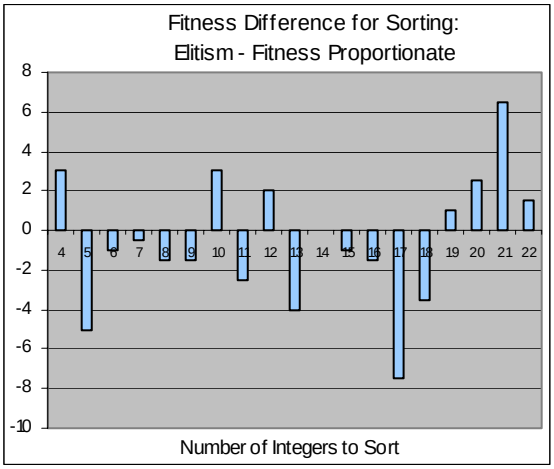

Fig. 13. Comparison of Elitism to Fitness Proportionate regarding their effects on fitness for Sorting

Apparent from Figure 14 and Figure 15, Tournament selection produces greater fitness values over Elitism and Fitness Proportionate for all problem sizes. However, there seems to be little correlation between the two graphs regarding the rising and falling trends or locations of minimums and maximums.

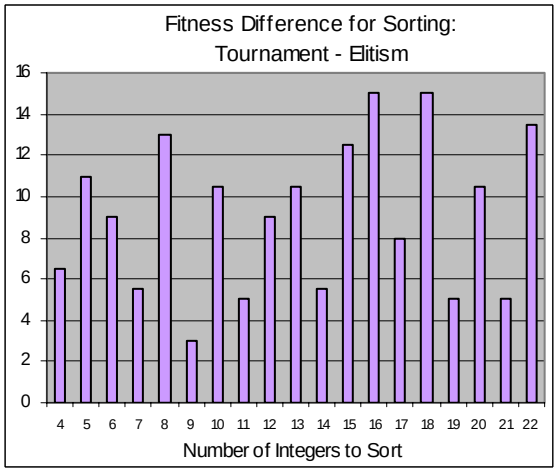

Fig. 14. Comparison of Tournament to Elitism regarding their effects on fitness for Sorting

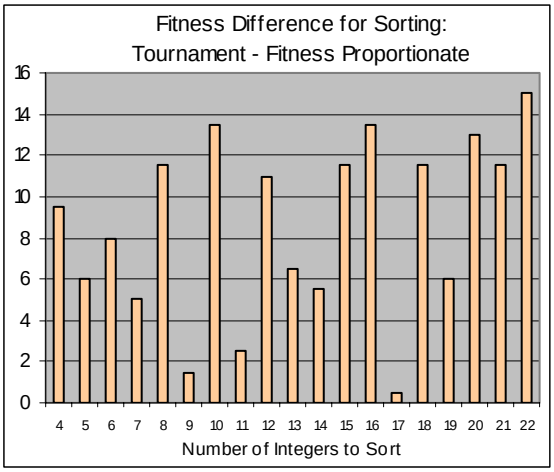

Fig. 15. Comparison of Tournament to Fitness Proportion regarding their effects on fitness for Sorting

Generations. Figure 16, Figure 17, and Figure 18 combine to show that Tournament selection always had the fewest number of generations for all problem sizes. Elitism and Fitness Proportionate produced the same number of generations on more than half (11) of the occasions. In nearly all the instances where Elitism and Fitness Proportionate did not result in the same number of generations, Fitness Proportionate had fewer generations. The only exception occurs at problem size 22.

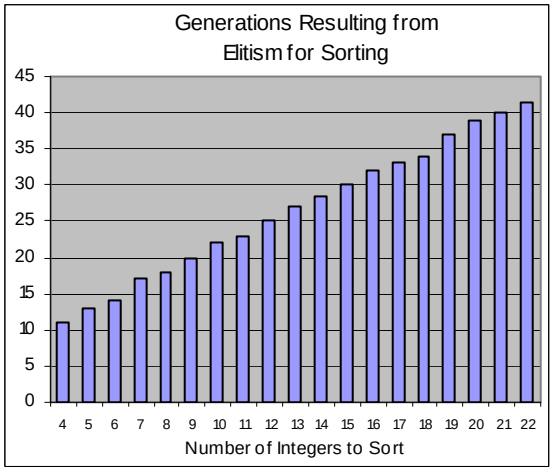

Fig. 16. The number of generations resulting from Elitism applied to Sorting

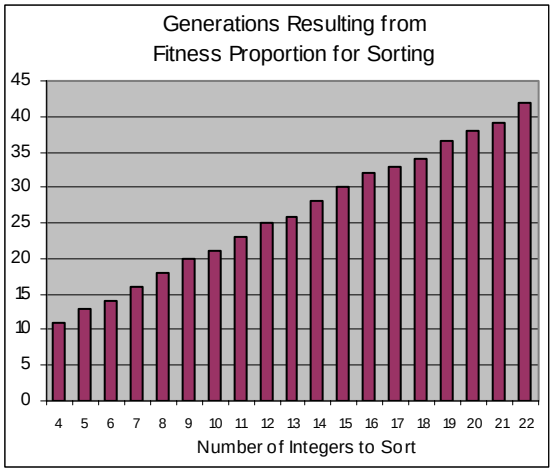

Fig. 17. The number of generations resulting from Fitness Proportionate applied to Sorting

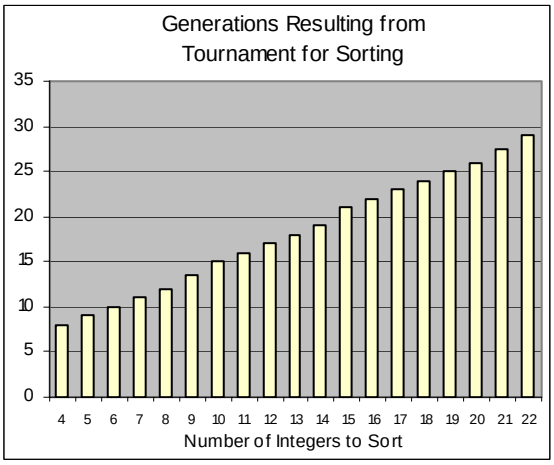

Fig. 18. The number of generations resulting from Tournament applied to Sorting

4 Conclusions

The goal of this experiment was to find a parent selection method that would perform better than any other method regardless of the problem type, size, or complexity to which it was applied. Although none of the three methods investigated here emerged as the best in every scenario, it seems warranted to declare Tournament selection as the best overall method. It resulted in the best fitness values and by the greatest margins than either of the other two selection methods. At the same time, Tournament selection always completed with the fewest generations.

Declaring the second best overall method is a much more formidable task. For Maximum Ones and 3-Processor Scheduling, the fitness graphs indicate a draw between Elitism and Fitness Proportionate. In addition, Fitness Proportionate holds only a slight edge in the fitness for Sorting. With regard to generations, Fitness Proportionate performs very slightly better for Sorting, a little better for Maximum Ones, and significantly worse for 3-Processor Scheduling. Further research is required to distinguish these two selection methods.

This experiment opens up avenues for further research in a variety of related areas. One idea is to alter the current algorithms slightly and see if the results still hold. For example, instead of applying the parent selection schemes on only the top 50% of the chromosomes, investigate the effects of applying the methods to the entire population. Another possibility is to introduce other selection methods and compare them to those researched here in an experiment similar to this one. The results of this type of research could lead to a decisive conclusion on whether a superior selection method exists without regard to problem size, type, and complexity.

References

1. Goldberg, D. E., Deb, K., A Comparative Analysis of Selection Schemes used in Genetic Algorithms. *Foundations of Genetic Algorithms*, pp. 69-93, Morgan Kaufman, 1991.
2. Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI, University of Michigan Press, 1975.
3. Hancock, P., An Empirical Comparison of Selection Methods in Evolutionary Algorithms. *Evolutionary Computing: AISB Workshop* Leeds, U.K. April 1994, Selected Papers, Springer-Verlag, 1994.
4. Rawlins, G.J.E. (1991). Introduction. *Foundations of Genetic Algorithms*, Morgan Kaufman. pp. 1-10.
5. Bovet, D., and Crescenzi, P. (1994). *Computational Complexity*, Prentice Hall.
6. Vose, M., The Simple Genetic Algorithm, Morgan Kaufman, 1999.
7. Mitchell, M., An Introduction to Genetic Algorithms, Prentice Hall, 1996.
8. Rylander, B., Foster, J., GA-hard Problems, *Proc. on Genetic and Evolutionary Computation Conference*, 2000.