Are Multiple Runs of Genetic Algorithms Better than One?

Erick Cantú-Paz[1] and David E. Goldberg[2]

[1] Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550
cantupaz@llnl.gov
[2] Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Avenue Urbana, IL 61801
deg@uiuc.edu

Abstract. There are conflicting reports over whether multiple independent runs of genetic algorithms (GAs) with small populations can reach solutions of higher quality or can find acceptable solutions faster than a single run with a large population. This paper investigates this question analytically using two approaches. First, the analysis assumes that there is a certain fixed amount of computational resources available, and identifies the conditions under which it is advantageous to use multiple small runs. The second approach does not constrain the total cost and examines whether multiple properly-sized independent runs can reach the optimal solution faster than a single run. Although this paper is limited to additively-separable functions, it may be applicable to the larger class of nearly decomposable functions of interest to many GA users. The results suggest that, in most cases under the constant cost constraint, a single run with the largest population possible reaches a better solution than multiple independent runs. Similarly, a single large run reaches the global faster than multiple small runs. The findings are validated with experiments on functions of varying difficulty.

1 Introduction

Suppose that we are given a fixed number of function evaluations to solve a particular problem with a genetic algorithm (GA). How should we use these evaluations to maximize the expected quality of the solution? One possibility would be to use all the evaluations in a single run of the GA with the largest population possible. This approach seems plausible, because it is well known that, in general, the solution quality improves with larger populations. Alternatively, we could use a smaller population and run the GA multiple times, keeping the best solution found by the different runs. Although the quality per run is expected to decrease, we would have more chances of reaching a good solution.

This paper examines the tradeoff between increasing the likelihood of success of a single run vs. using more trials to reach the goal. The first objective is to

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 801–812, 2003.

determine what configuration reaches solutions with the highest quality. The paper also examines the question of single vs. multiple runs removing the constant cost constraint. The objective in this case is to determine what configuration reaches the solution faster.

It would be desirable to find that multiple runs are advantageous, because they could be executed concurrently on different processors. Multiple independent runs are a special case of island-model parallel GAs, and have been studied in that context before with conflicting and controversial results [1,2,3,4,5]. Some results suggest that multiple runs can reach solutions of similar or better quality than a single run in a shorter time, which implies that superlinear speedups are possible.

Most of the previous work on this topic has been experimental, which makes it difficult to identify the problem characteristics that give an advantage to multiple runs. Instead of trying to analyze experimental results from a set of arbitrarily-chosen problems, we use simple mathematical models and consider only additively separable functions. The paper clearly shows when one approach can be superior, and reveals that, for the functions considered, multiple runs are preferable only in conditions of limited practical value.

The paper also considers the extreme case when multiple runs with a single individual—which are equivalent to random search—are better in terms of expected solution quality than a single GA. Although it is known than in some problems random search must be better than GAs [6], it is not clear on what problems this occurs. This paper sheds some light on this topic.

The next section summarizes related work on this area. The gambler's ruin (GR) model [7] is summarized in section 3 and extended to multiple independent runs in section 4. Section 5 presents experiments that validate the accuracy of the models. Section 6 lifts the total cost constraint and discusses multiple short runs. Finally, section 7 presents a summary and the conclusions.

2 Related Work

Since multiple runs can be executed in parallel, they have been considered by researchers working with parallel GAs. Tanese [1] found that, in some problems, the best overall solution found in any generation by multiple isolated populations was at least as good as the solution found by a single run. Similarly, multiple populations showed an advantage when she compared the best individual in the final generation. However, when she compared the average population quality at the end of the experiments, the single runs seemed beneficial.

Other studies also suggest that multiple isolated runs can be advantageous. For example, Shonkwiler [2] used a Markov chain model to argue that multiple small independent GAs can reach the global solution using fewer function evaluations than a single GA. He suggested that superlinear parallel speedups are possible if the populations are executed concurrently on a parallel computer.

Nakano, Davidor, and Yamada [8] proved that, under the fixed cost constraint, there is an optimal population size and corresponding run count that

maximizes the chances of reaching a solution of certain quality, if the single-run success probability increases with larger populations until it reaches a saturation point (less than 1). The method used in the current paper can be used to find this optimum, but a numerical optimization would be required, because efforts to characterize the optimal configuration in closed form have been unsuccessful.

Cantú-Paz and Goldberg [3] compared multiple isolated runs against a single run that reaches a solution of the same expected quality. They determined that—even without a fixed time constraint—the savings on execution time seemed marginal when compared against a single GA, and recommended against using isolated runs. The findings in the present paper, however, show that with the cost constraint there are some cases where multiple runs are advantageous.

Recently, Fuchs [4] and Fernández et al. [5] studied empirically multiple isolated runs of genetic programming. They found that in some cases it is advantageous to use multiple small runs. Luke [9] studied the tradeoff between executing a single run for many generations or using multiple shorter runs to find solutions of higher quality given a fixed amount of time. In two out of three problems, his experiments showed that multiple short runs were preferable.

There have been several attempts to characterize the problems in which GAs perform better than other methods [10,11]. However, without relating the performance of the algorithms to properties of the problems it is difficult to make predictions and recommendations for unseen problems, even if they belong to the same class. This paper identifies cases where random search reaches better solutions based on properties that describe the difficulty of the problems.

3 The Gambler's Ruin Model

It is common in GAs to encode the variables of the problem using a finite alphabet Σ. A schema is a string over $\Sigma \cup \{*\}$ that represents the set of individuals that have a fixed symbol $\mathsf{F} \in \Sigma$ in exactly the same positions as the schema. The $*$ is a "don't care" symbol that matches anything. For example, in a domain that uses 10-bit binary strings, the individuals that start with 1 and have a 0 in the second position are represented by the schema $10********$.

The number k of fixed positions in a schema is its order. Low-order highly-fit schemata are sometimes called building blocks (BBs) [12]. Following Harik et al. [7], we refer to the lowest-order schema that consistently leads to the global optimum as the correct BB. In this view, the correct BB must (1) match the global optimum *and* (2) have the highest average fitness of all the schemata in the same partition. All other schemata in the partition are labeled as incorrect.

Harik, Cantú-Paz, Goldberg, and Miller [7] modeled selection in GAs as a biased random walk. The number of copies of the correct BB in a population of size n is represented by the position, x, of a particle on a one-dimensional space. Absorbing barriers at $x = 0$ and $x = n$ bound the space, and represent ultimate convergence to the wrong and to the right solutions, respectively. The initial position of the particle, x_0, is the number of copies of the correct BB in the initial population.

At each step of the random walk there is a probability, p, of obtaining one additional copy of the correct BB. This probability depends on the problem that the GA is facing, and Goldberg et al. [13] showed how to calculate it for functions composed of m uniformly-scaled subfunctions. The probability that a particle will eventually be captured by the absorbing barrier at $x = n$ is [14]

$$P_{bb}(x_0, n) = \frac{1 - \left(\frac{q}{p}\right)^{x_0}}{1 - \left(\frac{q}{p}\right)^{n}} \tag{1}$$

where $q = 1 - p$. Therefore, the expected probability of success is

$$P_s(n) = \sum_{x_0=0}^{n} P_0(x_0) \cdot P_{bb}(x_0, n), \tag{2}$$

where $P_0(x_0) = \binom{n}{x_0} \left(\frac{1}{\chi^k}\right)^{x_0} \left(1 - \frac{1}{\chi^k}\right)^{n-x_0}$ is the probability of having exactly x_0 correct BBs in the initial population, and $\chi = |\Sigma|$ is the cardinality of Σ.

The GR model makes several assumptions, but it has been shown that it accurately predicts the solution quality of artificial and real-world problems [7, 15]. For details, the reader is referred to the paper by Harik et al. [7], but one assumption affects the experiments in this paper: Having absorbing walls bounding the random walk implicitly assumes that mutation and crossover do not create or destroy BBs. The only source of BBs is the random initialization of the population. This is why the experiments described below do not use mutation.

4 Multiple Small Runs

We measure the quality, Q, of the solution as the number of partitions that converge to the correct BBs. The probability that one partition converges correctly is given by the GR model, $P_s(n)$ (Equation 2). For convenience, we use $P_1 = P_s(n_1)$ to denote the probability that a partition converges correctly in one run with population size n_1 and $P_r = P_s(n_r)$ for the probability that a partition converges correctly in one of the multiple runs with a population size n_r.

4.1 Solution Quality

Under the assumption that the m partitions are independent, the quality has a binomial distribution with parameters m and $P_s(n)$. Therefore, the expected solution quality of a single run is $\mathrm{E}(Q) = mP_s(n)$. Of course, some runs will reach better solutions than others, and when we use multiple runs we consider that the problem is solved when one of them finds a solution of the desired quality. Let $Q_{r:r}$ denote the quality of the best solution found by r runs of size n_r. We are interested in its expected value, which can be calculated as [16]

$$\mathrm{E}(Q_{r:r}) = \sum_{x=0}^{m-1} 1 - F^r(x), \tag{3}$$

where $F(x) = P(Q \leq x) = \sum_{j=0}^{x} \binom{m}{j} P_r^j (1 - P_r)^{m-j}$ is the cumulative distribution function of the solution quality. Unfortunately, there is no closed-form expression for the means of maximal order statistics of binomial distributions. However, there are approximations for the extreme order statistics of the Gaussian distribution, and we can use them to make some progress in our analysis. We can approximate the binomial distribution of the quality with a Gaussian, and normalize the number of correct partitions by subtracting the mean and dividing by the standard deviation: $Z_{r:r} = \frac{Q_{r:r} - mP_r}{\sqrt{mP_r(1-P_r)}}$. Let $\mu_{r:r} = \mathrm{E}(Z_{r:r})$ denote the expected value of $Z_{r:r}$. We can approximate the expected value of the best quality in r runs as

$$\mathrm{E}(Q_{r:r}) \approx mP_r + \mu_{r:r}\sqrt{mP_r(1-P_r)}. \tag{4}$$

If there are no restrictions on the total cost, adding more runs to an experiment results in a higher quality. The problem is that $\mu_{r:r}$ increases very slowly as more runs are used: $\mu_{r:r} \approx \sqrt{\sqrt{2}\ln r}$. Therefore, the increase in quality is marginal, and multiple isolated runs seem unappealing [20].

However, the situation may be different if the total cost is constrained. Equation 4 shows an interesting tradeoff: $\mu_{r:r}$ grows as r increases, but P_r decreases because the population size per run must decrease to keep the cost constant. Multiple runs would perform better than a single one if the quality degradation is not too pronounced. In fact, the tradeoff suggests that there is an optimal number of runs and population size that maximize the expected quality. Unfortunately, we cannot obtain a closed-form expression for these optimal parameters.

The quality reached by multiple runs is better than one run if

$$mP_r + \mu_{r:r}\sigma_r > mP_1, \tag{5}$$

where $\sigma_r = \sqrt{mP_r(1-P_r)}$. We can bound the standard deviation as $\sigma_r = 0.5\sqrt{m}$ to obtain an upper bound on the quality of the multiple runs. Substituting this bound into the inequality above, dividing by m, and rearranging we obtain

$$\frac{\mu_{r:r}}{2\sqrt{m}} > P_1 - P_r. \tag{6}$$

This equation shows that multiple runs are more likely to be beneficial on short problems (small m), everything else being equal. This is bad news for the case of multiple runs, because interesting problems in practice may be very long.

The equation above also shows that multiple runs can be advantageous if the difference between the solution qualities is small. This may happen at very small population sizes where the quality is very poor, even for a single run. This case is not very interesting, because normally we want to find high-quality solutions. However, the difference is also small when the quality does not improve much after a critical population size. This is the case that Nakano et al. [8] examined, and represents an interesting possibility where multiple runs can be beneficial. The optimum population size is probably near the point where there is no further improvement: Using a larger population would be a waste of resources, which would be better used in multiple runs to increase the chance of success.

4.2 Models of Convergence Time

We can write the fixed number of function evaluations that are available as

$$T = rgn_r, \tag{7}$$

where g is the domain-dependent number of generations until the population converges to a unique value, r is the number of independent runs, and n_r is the population size of each run. GAs are often stopped after a fixed number of generations, with the assumption that they have converged by then. In the remainder we assume that the generations until convergence are constant. Therefore, to maintain a fixed total cost, the population size of each of the multiple runs must be $n_r = n_1/r$, where n_1 denotes the population size that a single run would use.

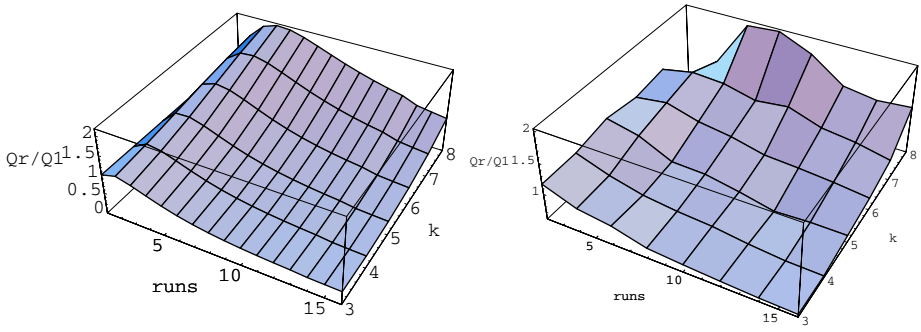
Assuming that g is constant may be an oversimplification, since it has been shown that the convergence time depends on factors such as the population size and the selection intensity, I. For example, under some conditions, the generations until convergence are given by $g \approx \frac{\pi}{2}\frac{\sqrt{n}}{I}$ [17]. In general, if the generations until convergence are given by the power-law model $g = \kappa n^{\theta}$, the population size of each of the multiple runs would have to be $n_r = n_1/r^{1/(\theta+1)}$ to keep the total cost constant (e.g., in the previous equation, $\theta = 1/2$ and n_r would be $n_1/r^{2/3}$). This form of n_r would give an advantage to the multiple runs, because their sizes (and the quality of their solutions) would not decrease as much as with the constant g assumption, so this assumption is a conservative one.

4.3 Random Search

Using all the available computation time in one run with a large population is clearly one extreme. The other extreme are multiple runs with the smallest population, which is one individual. The latter case is equivalent to random search, because there is no evolution possible (we are assuming no mutation). The models above account for the two extreme cases. When the population size is one, $P_r = \frac{1}{\chi^k}$, because only one term in equation 2 is different from zero. The quality of the best solution found by r runs of size one can be calculated with equation 3.[1]

To identify when random search can outperform a GA, we calculated the expected solution quality using equation 3 varying the order of the BBs, k, and the number of runs. The next section will define the functions used in these calculations; for now we only need to know that k varied. Figure 1 shows the ratio of the quality obtained by random search over the quality found by a simple GA with a population size of $n_1 = r$. Values over 1 indicate that multiple runs perform better. The figure shows that random search has an advantage as the problems become harder (with longer BBs). However, this peculiar behavior occurs only at extremely low population sizes, where the solution quality is so low

[1] Taking $Q_{r:r} = m[1-(1-\frac{1}{\chi^k})^r]$ may seem tempting, but it greatly overestimates the true quality. This calculation implicitly assumes that the final solution is formed by correct BBs that may have been obtained in different runs.



(a) Theory (b) Experiments

Fig. 1. Ratio of the quality of multiple runs of size 1 (random search) vs. a single run varying the order of the BBs and the number of runs.

that it is of no practical importance. When we increase the population size (and the number of random search trials), the GA moves ahead of random search.

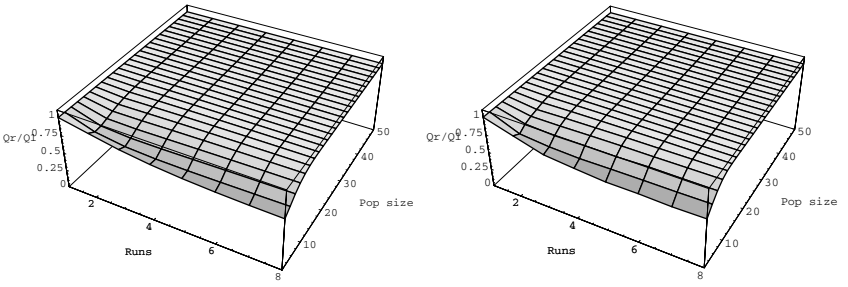
These results suggest that superlinear speedups can be obtained if random trials are executed in parallel and the simple GA is used as the base case. Interestingly, Shonkwiler [2] used very small population sizes (≈ 2 individuals) and at least two of his functions are easily solvable by random search.

5 Experiments

The GA in the experiments used pairwise tournament selection without replacement, one-point crossover with probability 1, and no mutation. All the results presented in this section are the average of 200 trials.

The first function is the one-max function with a length of $m = 25$ bits. We varied the population size n_r from 2 to 50 individuals. For each population size, we varied the number of runs from 1 to 8 and recorded the quality of the best solution found in any of the runs, $Q_{r:r}$. Figure 2 shows the ratio of $Q_{r:r}$ over the quality Q_1 that a GA with a population size $n_1 = rn_r$ reached. The experiments match the predictions well, and in all cases the larger single runs reached solutions of better quality than the multiple smaller runs.

To illustrate that multiple runs are more beneficial when m is small, we conducted experiments varying the length of the problem to $m = 100$ and $m = 400$ bits. The population size per run was fixed at $n_r = 10$, and the number of runs varied from 1 to 8. The results in figure 3 clearly show that as the problems become longer, the single large runs find better solutions than the multiple runs.



(a) Theory (b) Experiments

Fig. 2. Ratio of the quality of multiple runs vs. a single run for the one-max with $m = 25$ bits.

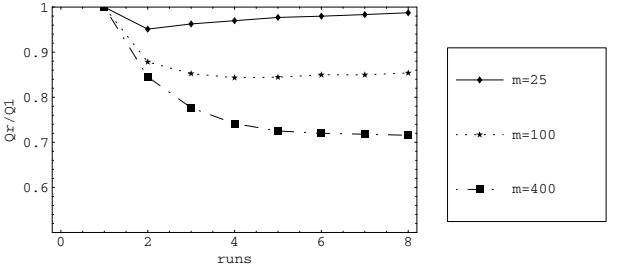
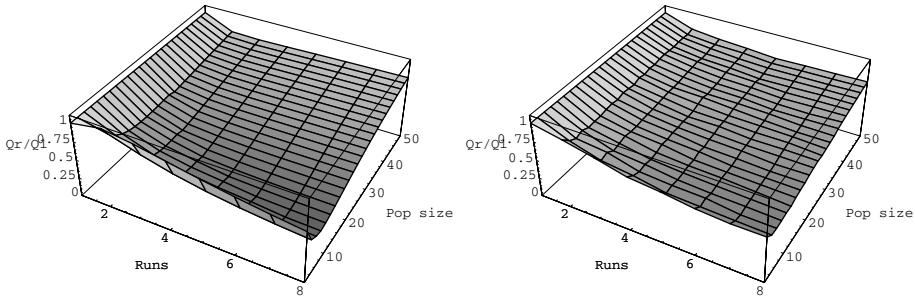


Fig. 3. Ratio of the quality of multiple runs vs. a single run varying the problem size.

The next two test functions are formed by adding fully-deceptive trap functions [18]. The order-k traps are defined as

$$f_{\text{dec}}^{(k)}(u) = \begin{cases} k-u-1 & \text{if } u < k, \\ k & \text{if } u = k. \end{cases} \tag{8}$$

Two deceptive test function were formed by concatenating $m = 25$ copies of $f_{\text{dec}}^{(3)}$ and $f_{\text{dec}}^{(4)}$. Figures 4 and 5 show the ratio $Q_{r:r}/Q_1$, varying the run size from 2 to 100 individuals and the number of runs from one to eight. The experimental results are very close to the predictions, except with very small population sizes, where the GR model is inaccurate. In most cases, the ratio is less than one, indicating that a single large run reaches a solution with better quality than multiple small runs. The exceptions occur at very small population sizes, where even random search performs better.



(a) Theory (b) Experiments

Fig. 4. Ratio of the quality of multiple runs vs. a single run for the order-3 trap.

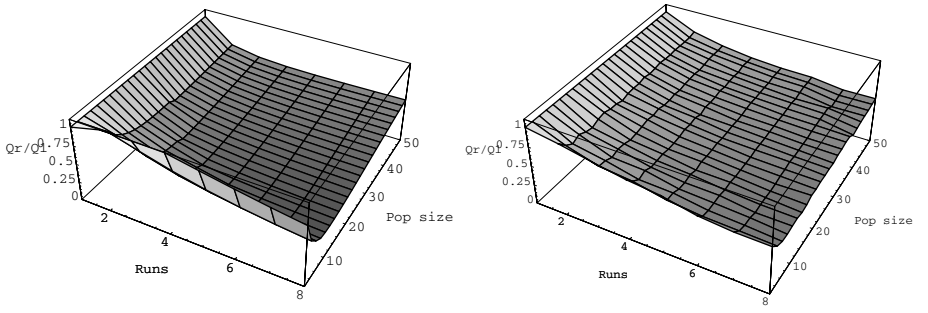
We performed experiments to validate the results about random search. Figure 1b shows the ratio of the quality of the solutions found by the best of r random trials and the solution obtained by a GA with a population size of r. For each value of k from 3 to 8, the test functions were formed by concatenating $m = 25$ order-k trap functions. The experiments show the same general tendency as the predictions (figure 1a).

6 Multiple Short Runs

Until now we have examined the solution quality under the constant cost constraint and after the population converges to a unique solution. However, in practice it is common to stop a GA run as soon as it finds a solution that meets some quality criterion. The framework introduced in this paper could be applied to this type of experiment, if we had a model that predicted the solution quality as a function of time: $P_s(n,t)$. In any generation (or any other suitable time step), the expected solution quality in one run would be $mP_s(n,t)$, but again we would be interested in the expected value of the best solution in the r runs, which can be found by substituting the appropriate distribution in equation 3.

There are existing models of quality as a function of time, but they assume that the population is sized such that the GA will reach the global solution and that recombination of BBs is perfect [17]. If we adopt these assumptions, we could use the existing models, but we would not be able to reduce the population size to respect the constraint of fixed cost. Mühlenbein and Schlierkamp-Voosen [17] derived the following expression for the one-max function:

$$P_s(n,t) = \frac{1}{2}\left(1 + \sin(\frac{I}{\sqrt{n}}t)\right), \tag{9}$$



(a) Theory (b) Experiments

Fig. 5. Ratio of the quality of multiple runs vs. a single run for the order-4 trap.

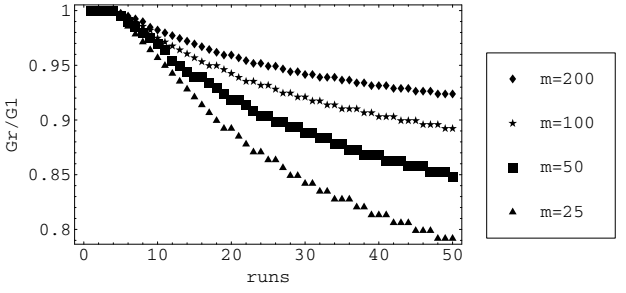


Fig. 6. Ratio of the generations until convergence of multiple over single runs. The total cost is not constant.

and Miller and Goldberg [19] used it successfully to predict the quality of deceptive functions. If we abandon the cost constraint, we can show that the best of multiple runs of the same size (that is at least large enough to reach the global optimum) reaches the solution in fewer generations than a single run of the same size. This argument has been used in the past to support the use of multiple parallel runs [2].

Figure 6 shows the ratio of the number of generations until convergence (to the global) of multiple runs over the number of generations of convergence of a single run. The figure shows that the time decreases as more runs are used, and the advantage is more pronounced for shorter problems. If each run was executed concurrently on a different processor of a parallel machine, the elapsed time to reach the solution would be reduced (assuming that the cost to determine convergence by any run is negligible, which may not be the case). However, this

scheme offers a relatively small advantage, and it is probably not the best use of multiple processors since we can obtain almost linear speedups in other ways [20].

7 Summary and Conclusions

There are conflicting reports of the advantage of using one or multiple independent runs. This problem has consequences on parallel GAs with isolated populations and also to determine when random search can outperform a GA. This paper presented an analytical study that considered additively-separable functions. Under a constraint of fixed cost and assuming no mutation, the analysis showed that the expected quality of the solution reached by multiple independent small runs is higher than the quality reached by a single large run only in very limited conditions. In particular, multiple runs seem advantageous at very small population sizes, which result in solutions of poor quality, and close to a saturation point where the solution quality does not improve with increasingly larger populations. In addition, the greatest advantage of multiple independent runs is on short problems, and the advantage tends to decrease with higher BB order. The results suggest that for difficult problems (long and with high-order BBs), the best alternative is to use a single run with the largest population possible. Small independent runs should be avoided.

Acknowledgments. We would like to thank Hillol Kargupta, Jeffrey Horn, and Georges Harik for many interesting discussions on this topic. UCRL-JC-142172. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48. Portions of this work were 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 the National Science Foundation under grant DMI-9908252.

References

1. Tanese, R.: Distributed genetic algorithms. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989) 434–439
2. Shonkwiler, R.: Parallel genetic algorithms. In Forrest, S., ed.: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann (1993) 199–205
3. Cantú-Paz, E., Goldberg, D.E.: Modeling idealized bounding cases of parallel genetic algorithms. In Koza, J., et al., eds.: Proceedings of the Second Annual Genetic Programming Conference, Morgan Kaufmann (1997) 353–361
4. Fuchs, M.: Large populations are not always the best choice in genetic programming. In Banzhaf, W., et al., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (1999) 1033–1038

5. Fernández, F., Tomassini, M., Punch, W., Sánchez, J.M.: Experimental study of isolated multipopulation genetic programming. In Whitley, D., et al., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2000) 536
6. Wolpert, D., Macready, W.: No-free-lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82
7. Harik, G., Cantú-Paz, E., Goldberg, D., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation **7** (1999) 231–253
8. Nakano, R., Davidor, Y., Yamada, T.: Optimal population size under constant computation cost. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving fron Nature, PPSN III, Berlin, Springer-Verlag (1994) 130–138
9. Luke, S.: When short runs beat long runs. In Spector, L. et al., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2001) 74–80
10. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing? In Advances in Neural Information Processing Systems 6 (1994) 51–58
11. Baum, E., Boneh, D., Garrett, C.: Where genetic algorithms excel. Evolutionary Computation **9** (2001) 93–124
12. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA (1989)
13. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. Complex Systems **6** (1992) 333–362
14. Feller, W.: An Introduction to probability theory and its applications. 2nd edn. Volume 1. John Wiley and Sons, New York, NY (1966)
15. van Dijk, S., Thierens, D., de Berg, M.: Scalability and efficiency of genetic algorithms for geometrical applications. In Schoenauer, M., et al., eds.: Parallel Problem Solving from Nature—PPSN VI, Berlin, Springer-Verlag (2000) 683–692
16. Arnold, B., Balakrishnan, N., Nagaraja, H.N.: A first course in order statistics. John Wiley and Sons, New York, NY (1992)
17. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. Evolutionary Computation **1** (1993) 25–49
18. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In Whitley, L.D., ed.: Foundations of Genetic Algorithms 2, Morgan Kaufmann (1993) 93–108
19. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation **4** (1996) 113–131
20. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Boston, MA (2000)