Predicting Speedups of Idealized Bounding Cases of Parallel Genetic Algorithms

Erick Cantú-Paz
Illinois Genetic Algorithms Laboratory
Department of Computer Science
University of Illinois at Urbana-Champaign
cantupaz@illigal.ge.uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
deg@illigal.ge.uiuc.edu

Abstract

This paper presents models that predict the speedup of two cases that bound the possible topologies and migration rates of parallel genetic algorithms (GAs). The first bounding case is a parallel GA with completely isolated demes or subpopulations and for this case the model and the experiments show that the speedup is not very significant when more demes are used. The second model predicts the speedup when each deme communicates with every other deme using a maximal migration rate. For this case, we show that when the communication time is not constant there is a combination of number of demes and deme size that maximizes the speedup. The models are validated with computational experiments using functions of varying difficulty.

1 Introduction

Many claims have been made about the practical benefits of parallel genetic algorithms. To quantify these benefits we can compute the expected parallel speedup, comparing a parallel GA with a simple GA that finds a solution of the same quality. This paper presents models that predict the expected speedup for two cases that bound the topologies and migration rates of coarse-grained parallel GAs.

As a first step to compute the parallel speedup it is necessary to compute the computational costs needed by the serial and the parallel GAs to find a solution. The computational cost is largely determined by the population size (Goldberg and Deb, 1991) and we build on a previous study where Cantú-Paz and Goldberg (1997) derived population-sizing equations for the two bounding cases considered in this paper.

The first bounding case is a completely isolated topology where the demes do not communicate with each other. This case is clearly a lower bound of the connectivity of a topology and of the migration rate. The second bounding model considers a fully connected topology, where each deme communicates to all the others, and the migration rate is set to a maximum value. We recognize that users of parallel GAs use intermediate values for the connectivity and migration rate, but these extreme cases serve as indicators of the performance of parallel GAs. Additionally, if the simplifications and assumptions used in this paper are shown to be viable then the models can be specialized later.

The second step to compute the parallel speedup is to consider the time consumed by communications. To this effect we use a general power-law model and show that, under most circumstances, there exists an optimal number of demes and an associated optimal deme size that maximize the parallel speedup.

This study begins with a review of a few representative publications concerning speedups in parallel GAs. Next, section 3 summarizes the models that predict the population size of a simple GA, a parallel GA with isolated demes, and a parallel GA with fully connected demes. In section 4 the population-sizing models are used to predict the speedup for the two bounding cases of parallel GAs, and for each case computational experiments are performed to test the models. Finally, this paper ends with the conclusions of this work and suggestions for further research.

2 Background

Several authors have reported gains in the time required to find a solution for particular problems with parallel GAs. In this section we revisit some represen-

tative publications about speedups in parallel GAs. It is not our intention to present a comprehensive review of the area, but instead to highlight a few methodological considerations that are used constantly in the literature. A more detailed overview of parallel GAs can be found in (Cantú-Paz, 1995).

Tanese initiated the systematic study of parallel GAs with a set of experiments designed to test the effect of the migration rate and the frequency of migrations on the efficiency of the algorithm and on the quality of the solutions found (Tanese 1987; Tanese 1999a; Tanese 1989b). Her approach was to divide a fixed-sized population into some number of equally-sized demes and her experiments show that "near-linear" speedups can be obtained if migration occurs infrequently and the migration rate is low.

An important methodological problem in Tanese's study is that the questions of efficiency (how fast is the parallel GA?) and of the quality of the solution are treated independently. The same method is followed by many other researchers who study parallel GAs. With this method, it should come as no surprise that reports of linear or quasi-linear speedups appear constantly.

Recently, Belding (1995) confirmed Tanese's findings using different test functions. In addition, he shows superlinear speedups for up to 32 processors for a parallel GA working on the Royal Road functions R1 and R4. As in Tanese's study the quality of the solutions is reported separately from the results on efficiency, and in the case of the superlinear speedup the quality is not reported at all.

This paper distances itself from this traditional methodology and studies together the problems of solution quality and algorithm efficiency. By integrating the two notions of time and quality, the models presented in this paper reach different conclusions than most previous studies on parallel GAs.

We follow Goldberg, Deb, and Thierens (1993) and restrict the notion of a building block (BB) to the minimum-order schemata that contribute to the global optimum. In this paper we use deceptive functions as test cases for our models, and for a fully deceptive function with deception length k, the BBs are the optimal schemata of length k. In this view the juxtaposition of two BBs at a particular string does not lead to a BB of length $2k$, but instead to two separate BBs of length k.

A primary assumption of this paper is that the computation time is dominated by the time consumed in evaluating the objective function. The GA computations (selection, crossover, and mutation) are relatively simple and we ignore the computational effort employed in them. Under this assumption the computation cost of a GA is proportional to the number of function evaluations used.

Since the number of function evaluations is related directly to the population size, in order to determine the computation time we need to find the population size in the serial and parallel cases. In the next section we summarize some recently-developed models for population sizing.

3 Population sizing for serial and parallel GAs

First, this section reviews a result by Harik, Cantú-Paz, Goldberg, and Miller (1997) that allows us to calculate the population size needed by a serial GA to find a solution of a certain quality. Next, we review recent results for the population size required by the two bounding cases of parallel GAs that are studied in this paper.

3.1 Single-population GAs

To obtain a model of the quality of the solution of a GA, Harik, Cantú-Paz, Goldberg, and Miller (1997) modeled the selection process in a GA as a one-dimensional random walk. The model concentrates on only one partition of length k and assumes that decisions are independent across partitions. In the random walk model the number of copies of the correct BB in the partition 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 expected number of copies of the best BB in a randomly initialized population, that is equal to $x_0 = n/2^k$, where k is the order of the BB and n is the population size.

At each step of the random walk there is a probability, p, of getting one additional copy of the correct BB. This probability is different for every problem and represents the probability of deciding well between two competing BBs. For functions composed by adding several uniformly-scaled subfunctions, p was computed by Goldberg, Deb, and Clark (1992) in their study of population sizing as

$$p = N\left(\frac{d}{\sqrt{2m'\sigma_{bb}^2}}\right),$$

where N denotes the cumulative distribution function of a normal distribution with a mean of zero and a standard distribution of one, d is the difference of the fitness contribution between the best and the second best schemata in the partition, $m' = m - 1$, m is the number of subfunctions, and σ_{bb} is the average variance in the k-th order partition.

Using a well-known result from the random walk literature, the probability that the particle will be absorbed ultimately at $x = n$, or equivalently the probability that the GA will converge to the right solution is

$$P_{bb} = \frac{1 - \left(\frac{q}{p}\right)^{n/2^k}}{1 - \left(\frac{q}{p}\right)^{n}},$$

where $q = 1 - p$ is the probability of making the wrong decision between two competing BBs. This probability can be approximated as (Harik, Cantú-Paz, Goldberg, and Miller, 1997)

$$P_{bb} \approx 1 - \left(\frac{q}{p}\right)^{n/2^k}, \quad (1)$$

because p is greater than q and n is greater than $n/2^k$. From this form of P_{bb} it is straightforward to compute the population size. Solving for n results in

$$n = \frac{2^k \ln(1 - \frac{b}{m})}{\ln\left(\frac{q}{p}\right)}, \quad (2)$$

where b is the number of correct BBs desired in the final solution and m is the total number of BBs in the problem. We proceed now to review the calculation of the population size for the first parallel case.

3.2 Isolated demes

The first bound for parallel GAs is when there is no communication between the demes. If there is no communication, then the migration rate is zero, and this is clearly a lower bound. Likewise, using demes in isolation is a lower bound on the connectivity of the topology.

We use a framework set forth by Nakano, Davidor, and Yamada (1994) and consider that the parallel GA succeeds when at least one of the demes finds the solution with the minimum required quality. The quality of a solution is determined as the number of partitions that converge to the correct BB.

To compute the overall probability of success, let P_m be the probability that one deme converges to a solution with at least b partitions correct out of m possible. It follows that $(1 - P_m)^r$ is the probability that none of the demes succeeds (in the paper by Nakano, Davidor, and Yamada (1994) the r stands for the number of independent runs, and in this paper we use r to denote the number of demes). Thus, the overall success probability for multiple demes is:

$$P_s = 1 - (1 - P_m)^r. \quad (3)$$

The number of partitions correct at each deme is a random variable with a binomial distribution (since we are assuming that each partition is independent of the others). We use the probability of absorption from the gambler's ruin problem as the probability P_{bb} that one partition converges to the correct BB and therefore, P_m may be computed as $P_m = 1 - B(b)$ where B denotes the cumulative distribution function (CDF) of a binomial distribution with parameters m and P_{bb}.

It is difficult to solve equation 3 for the population size n that is required to achieve a solution of a minimum quality. However, if we approximate the binomial distribution of P_m with a normal distribution, we can use a standard approximation for the normal distribution and solve for n. The complete derivation is long and cumbersome and is contained in (Cantú-Paz and Goldberg, 1997). With this method we find that the deme size needed to find a solution of a minimum quality with isolated demes is approximately

$$n_d = \frac{2^k \ln(1 - P^*_{bb})}{\ln(q/p)}, \quad (4)$$

where P^*_{bb} is the probability that in each deme a partition converges to the right value of the BB. For $r = 1$, P^*_{bb} is equal to b/m and the equation above is equivalent to the single-population sizing equation. P^*_{bb} decreases as more demes are used, so the deme size required by multiple isolated demes is smaller than the population size required by a simple GA to find a solution of the same quality. In section 4 we show how this reduction in the average quality that is required in each deme translates into savings in computational time.

3.3 Fully connected demes with maximal migration

The second bounding case for parallel GAs is where all demes send migrants to all the other demes. In this case, a fully connected topology represents an upper bound on connectivity. We also bound the migration rate with the highest possible value so that each deme

sends n/r individuals to every other. The model considers that migration occurs once after all demes have converged. After migration every deme resumes execution until it converges again.

The first step is to find the probability that one partition converges to the right BB. For this we use the gambler's ruin model again, but we note that the initial number of BBs is not $x_0 = n/2^k$ as in the case of a randomly initialized population, but instead $x_0 = nP_{bb}$. Therefore, the probability that a partition converges to the correct value after the second iteration is found by computing

$$P_{bb_2} = 1 - \left(\frac{q}{p}\right)^{nP_{bb}}. \tag{5}$$

The overall success criterion is the same as before (at least one deme must find a solution with at least b out of m BBs, see equation 3) and using the same method as before, the deme size required by fully connected demes can be approximated as (Cantú-Paz and Goldberg, 1997):

$$n_d = \frac{\sqrt{-2^k \ln(1 - P^*_{bb})}}{1 - \frac{q}{p}}. \tag{6}$$

Comparing the equation above with equation 4 we see that the deme size now depends on the *square root* of $2^k \ln(1 - P^*_{bb})$, and so the expected gains in computation time are much greater in the case with fully connected demes than with isolated demes.

In the next section we compute the expected parallel speedups using the deme sizing results reviewed above.

4 Parallel speedups

To measure the benefit of using a parallel algorithm we can calculate the gain in execution time that results from using multiple processors. This gain is usually expressed in terms of the parallel *speedup* of the algorithm.

In the GA community the topic of parallel speedups has raised significant controversy. Probably the main reason is that the execution times of the serial and parallel GAs are compared without considering the quality of the solutions found in each case. This may give an unfair advantage to the parallel GA, and it is clearly not the best way to compare the two algorithms.

In general, to make a fair comparison between any serial and parallel programs the two must give exactly the same result, otherwise the comparison is meaningless. However, in the case of stochastic algorithms like GAs, a fair comparison must be based on the *expected* quality of convergence in the serial and parallel cases. To this effect we use the models reviewed in the previous section as they predict the population size needed to obtain *on average* a solution of the desired quality.

This paper considers the conventional definition of the parallel speedup of an algorithm as the ratio of the execution time of the best serial algorithm, T_s, and the execution time of the parallel program, T_p (Almasi and Gottlieb, 1994). Assuming that there is one processor for every deme, the execution time in the parallel case is the time needed by *one* deme to complete its tasks. Therefore, T_p is the sum of the time used by one deme in computations (T_{COMP_p}) and the time it used to communicate information to its neighbors (T_{COMM_1}), so we can compute the speedup as

$$S_p = \frac{T_s}{T_{COMP_p} + T_{COMM_1}}. \tag{7}$$

Another cause for controversy on using the parallel speedup to compare serial and parallel GAs is that when the GA is parallelized (by using several interconnected demes, as we do) the result is an algorithm that is very different from the serial GA. The problem is that the parallel speedup was intended originally to compare the serial and the parallel versions of the same algorithm. However, we believe that the ratio of the execution times of the parallel and serial GAs is a useful measure to compare the two different algorithms.

We proceed to examine the expected speedup for the first bounding case of isolated demes.

4.1 Isolated demes

Since in this case the communication time is null, it is only necessary to determine the computation time for the serial and parallel cases in order to compute the expected speedup. As we mentioned before, we assume that the computation time is dominated by the time used in evaluating individuals, and the execution time is proportional to the number of function evaluations. In every generation, the GA evaluates n individuals and the number of generations needed to reach convergence may be assumed to be a domain-dependent constant, g. Holding the quality constant in the serial and parallel cases, the speedup for isolated demes can be computed as

$$S_p = \frac{gn}{gn_d} = \frac{n}{n_d}. \tag{8}$$

Recall that the population size for the single-

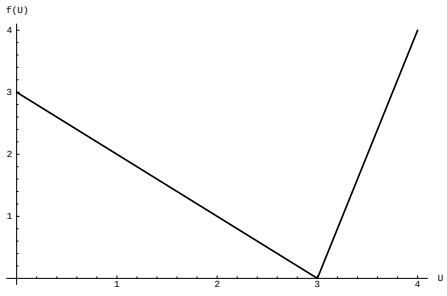

Figure 1: A deceptive 4-bit trap function of unity. The horizontal axis is the number of bits set to 1 and the vertical axis is the fitness value. The first test problem was constructed by concatenating 20 copies of this function.

population GA is given by equation 2 and, when the demes execute in isolation, the deme size n_d is predicted by equation 4. Substituting these population sizes into equation 8 and simplifying terms we obtain a simple equation for the speedup

$$S_p = \frac{\ln(1 - \frac{b}{m})}{\ln(1 - P^*_{bb})}. \tag{9}$$

To test this result we experimented with two functions of different difficulty. All our experiments with parallel GAs were performed on a network of RS/6000 workstations connected by a 10 Mbps Ethernet. The creation of demes and the communications between them were handled using PVM 3.3.

Our first test function is formed by concatenating $m = 20$ copies of a 4-bit trap function (see figure 1). For this function $\sigma_{bb} = 1.215$ and the difference between the optimal and the second best solution is $d = 1$, therefore $p = 0.5585$. The experiments use 2-point crossover (with probability 1.0), binary tournament selection and no mutation because the models assume that the only source of diversity is the initial random population.

For each deme count ($r = 1...16$) we increased the deme size until at least one of the demes found a solution with at least 16 of the 20 BBs. At this point we recorded the times used in computations. The results that we report are the average over 50 independent runs. The theoretical predictions for the parallel speedup along with the experimental results are plotted in figure 2 and it is evident from the figure that there is only a modest advantage in using more isolated demes.

The second function used in the experiments is an 8-

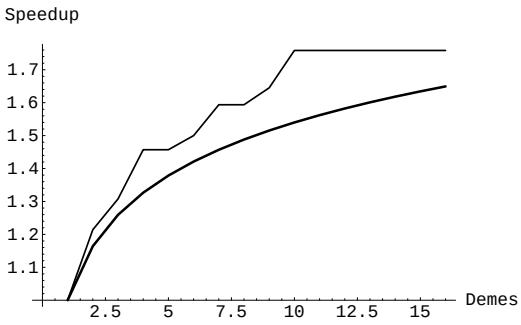

Figure 2: Projected and experimental speedups for a 4-bit trap function with 20 BBs using 1 to 16 isolated demes. The thin line shows the experimental results and the thick line is the theoretical prediction. The quality demanded was 16 BBs correct.

bit trap function similar to the 4-bit trap used before. The difference between the two best solutions is still $d = 1$, but the variance is higher ($\sigma_{bb} = 2.1804$), this causes the probability of deciding well to be smaller ($p = 0.5437$) and therefore we expect that the GA will need larger population sizes to find adequate solutions for this function. The 8-bit trap problem has longer BBs than the previous function and therefore we chose 1-point crossover to avoid excessive disruption (the crossover probability is 1.0 as before). The experiments continue to use binary tournament selection and no mutation. As with the first function, we demand that the solutions have at least 16 BBs correct. The experimental results are plotted in figure 3 along with the theoretical predictions for up to 16 demes. The results for the 8-bit trap function are of the same magnitude as for the previous function, and it appears that the speedup for the case of isolated demes is not very significant.

4.2 Speedups with communication

We proceed to examine the case where there is communication between the demes. To progress in the analysis we must make some simplifying assumptions. First, we assume that the number of demes in the parallel GA is equal to the number of processors available. Also, we assume that the total communication time has a general power-law form:

$$T_{COMM} = Cr^{\gamma},$$

where C and γ are constants. The specific values of C

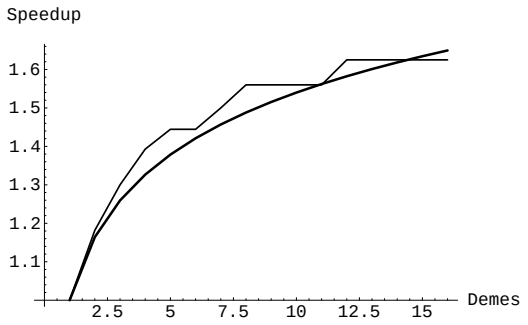

Figure 3: Projected (thick line) and experimental speedups (thin line) for a 8-bit trap function with 20 BBs using 1 to 16 isolated demes. The quality demanded was 16 BBs correct.

and γ depend on the choice of topology and on the particular hardware that is used to implement the parallel GA.

It is possible for the communications time to be a constant ($\gamma = 0$) if the parallel GA is implemented on a computer that has the hardware necessary to allow all inter-deme communications to occur simultaneously. For example, in a ring topology each deme communicates only with one other deme and communications may occur at the same time. However, as more complicated topologies are implemented, this illusion of perfectly parallel communications disappears. The worst case is when a fully connected topology is simulated on multicomputer with a bus-based architecture. In that case, the total communication time will be a quadratic function of the number of demes ($\gamma = 2$), as all r^2 logical links are accommodated on the bus.

The time that *each* deme uses to communicate is $T_{COMM_1} = T_{COMM}/r = Cr^{\gamma'}$, where $\gamma' = \gamma - 1$.

At this point we can make an important observation: note that as more demes are used the computation time decreases, because smaller deme sizes are needed, but at the same time the communication time increases monotonically with r. Therefore, as more demes are added there is a tradeoff between decreasing computation and increasing communication times. This tradeoff entails the existence of an optimal number of demes that minimizes the parallel time and maximizes the speedup. Note that the tradeoff between computation and communication times exists only when $\gamma > 0$. If the communication cost is a constant, then there is no optimal number of demes and performance improves (but only slightly) as more demes are used.

We proceed in our analysis and calculate the optimal number of demes for the more common case where the communication cost is not constant. Using the general model for communications, the parallel time can be expressed as:

$$T_p = T_{COMP_p} + Cr^{\gamma'}. \tag{10}$$

To optimize the parallel time, we set $\frac{\delta T_p}{\delta r}$ to zero and solve to obtain the optimal number of demes. Unfortunately, the derivative cannot be solved analytically for r and to make progress in the optimization of speedups we need to characterize the parallel computation time with a simpler expression.

We observed that plotting the parallel deme size against the number of demes on a log-log scale results in an almost straight line, which means that the deme size can be approximated closely with a general power-law equation: $n = Ar^B$, where A and B are constants.

We obtain the value of B as

$$B = \frac{\ln(n_1/n_2)}{\ln r_1/r_2},$$

where r_1 and r_2 are two deme counts, and n_1 and n_2 are the corresponding parallel population sizes (obtained using equation 6). The value for A can be obtained directly as $A = \frac{n}{r^B}$.

With this characterization of the parallel deme size we can now approximate the optimal number of demes using $T_{COMP_p} = gn = gAr^B$ and setting the derivative of equation 10 to zero. Solving for r,

$$r^* = \left(-\frac{gAB}{\gamma' C}\right)^{\frac{1}{\gamma' - B}}, \tag{11}$$

and we can approximate the optimal deme size as

$$n^* = Ar^{*B}. \tag{12}$$

We performed experiments to validate the models of this section using the same test functions of the previous section. We show in figures 4 and 5 the predictions and the experimental results of the parallel speedups using 4- and 8-bit trap functions, respectively. The parameters for the GAs were the same as for the experiments with isolated demes and we follow the same experimental procedure, except that in this case we record the time used in computation and also the time used in communications.

Both the predictions and the experimental results show the existence of an optimal speedup at approximately the same number of demes. Note that for the

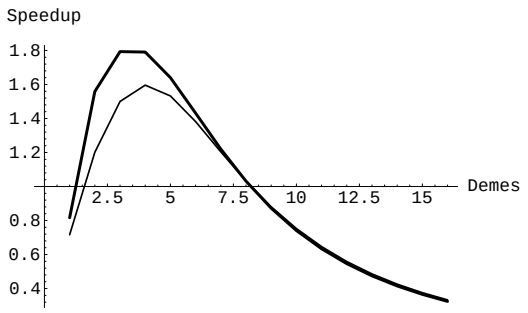

Figure 4: Projected and experimental speedups for a 4-bit trap function with 20 BBs using 1 to 16 fully connected demes. The quality demanded was 16 BBs correct. The thick line is the theoretical prediction and the thin line is the experimental results.

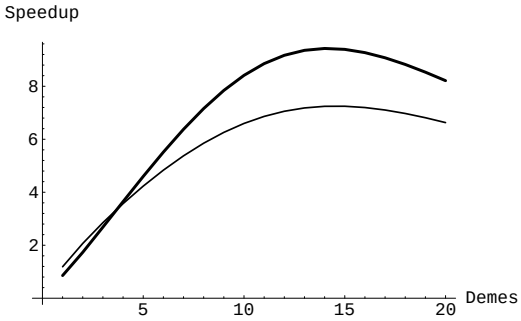

Figure 5: Projected (thick line) and experimental (thin line) speedups for a 8-bit trap function with 20 BBs using 1 to 16 fully connected demes. The quality demanded was 16 BBs correct.

4-bit trap function, the speedup is less than one for several deme counts, in effect this means that the parallel algorithm takes more time to finish than the serial GA. This occurs because our implementation has very expensive communications and because the 4-bit problem can be solved with a relatively small population, so that the savings on computation costs are too small to compensate for the rapidly increasing communications costs.

In the case of the 8-bit trap function the magnitude of the speedups is always greater than one. In fact, the parallel algorithm shows a significant advantage over the serial GA, even in our implementation with expensive communications. For this function the deme sizes required to find a solution with at least 16 BBs are much larger than for the 4-bit function, and the savings on the computation cost are enough to compensate for the expensive communications.

5 Conclusions and future work

This paper presents two models for bounding cases of the topology and the migration rate of parallel GAs. The models depart from the usual practice and integrate the notions of the quality of the solution and the time required to find it.

The first result of our analysis is that the speedup is very modest when there is no communication between the demes. This result may seem counterintuitive if one only considers the savings on computation time that are achieved when a single large population is partitioned into many small demes, but that the quality of the solution degrades rapidly as smaller demes are used.

The second result of the paper shows the existence of an optimal number of demes (and a corresponding optimal deme size) that minimizes the expected execution time when the communication cost is variable. Our analysis provides a simple and accurate method to compute these optimal values.

Moreover, the formulas for the optimal deme size and deme count can be adjusted to consider particular implementations and to predict the effect of improvements in hardware or software in the efficiency of parallel GAs. With these tools it is possible to guide the developer to those aspects of the implementation that will make a greater impact on performance.

There is an important question that arises from the two conclusions of the paper: how is it possible to get greater speedups when more processors are available? The problem is that using more isolated demes only improves slightly the execution time, and adding more demes to an optimal-sized set of communicating demes results in a performance loss. A possible solution is to "hybridize" the parallel GA and distribute the evaluation of the subpopulations among multiple processors. Another alternative to get greater speedups is to use tightly connected clusters of subpopulations as a single deme and exchange individuals occasionally across the clusters.

In this paper we bound the expected speedups of parallel GAs with the analysis of two cases that bound the ranges of possible topologies and migration rates. However, we recognize that this work may be more ap-

pealing to practitioners if the analysis is specialized to predict the expected speedups using other topologies and reduced migration rates. Work is already underway to construct models for those cases.

Acknowledgments

This study was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants number F49620-94-1-0103, F49620-95-1-0338, and F49620-97-1-0050. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

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

Erick Cantú-Paz was supported by a Fulbright-García Robles-CONACYT Fellowship.

References

Almasi, G., & Gottlieb, A. (1994). *Highly parallel computing* (2nd ed.). New York: Benjamin/Cummings Publishing Company.

Belding, T. C. (1995). The distributed genetic algorithm revisited. In Eschelman, L. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 114–121). San Mateo, CA: Morgan Kaufmann.

Cantú-Paz, E. (1995). *A summary of research on parallel genetic algorithms* (IlliGAL Report No. 95007). Urbana, IL: University of Illinois at Urbana-Champaign.

Cantú-Paz, E., & Goldberg, D. (1997). Modeling idealized bounding cases of parallel genetic algorithms. In Koza, J., Deb K., Dorigo, M., Fogel, D., Garzon, M., Iba, H., and Riolo, R. (Eds.) *Genetic Programming: Proceedings of the Second Annual Conference*. San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G. (Ed.) *Foundations of Genetic Algorithms*, *1*, 69–93.

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

Goldberg, D. E., Deb, K. & Thierens, D. (1993). Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, *32*, 10–16.

Harik, G., Cantú-Paz, E., Goldberg, D., & Miller, B. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In Bäck, T., Michalewicz, Z., and Yao, X. (Eds.) *1997 IEEE Proceedings of the International Conference of Evolutionary Computation* (pp. 7–12). New York: IEEE Press.

Nakano, R., Davidor, Y., & Yamada, T. (1994). Optimal population size under constant computation cost. In Davidor, Y., Schwefel, H.-P., & Männer, R. (Eds.), *Parallel Problem Solving from Nature, PPSN III* (pp. 130–138). Berlin: Springer-Verlag.

Tanese, R. (1987). Parallel genetic algorithm for a hypercube. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 177–183). Hillsdale, NJ: Lawrence Erlbaum Associates.

Tanese, R. (1989a). Distributed genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 434–439). San Mateo, CA: Morgan Kaufmann.

Tanese, R. (1989b). *Distributed genetic algorithms for function optimization*. Unpublished doctoral dissertation, University of Michigan, Ann Arbor.