Convergence Variability and Population Sizing in Micro-Genetic Algorithms

Ghassan Abu-Lebdeh & Rahim F. Benekohal*

Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA

Abstract: The issues of micro-genetic algorithm (micro-GA) convergence, population size, internal variability, and performance are examined. Procedures are developed to determine best population size when micro-GAs are used for optimization of real-world problems. Results show that the best population size cannot be determined before convergence has occurred and that convergence may occur without reaching an optimal solution. To determine the best population size, this article proposes the use of consistency of the final fitness values in addition to the fitness value at convergence. Internal variability must be considered in finding the best population size, and that best population size depends on the level of computational resources used. Smaller populations with generous computational resources performed as good as and sometimes better than larger populations. Midsized populations consistently exhibited lower variation in fitness value. For a given population size, the increase in fitness value may not be worth the added computation cost beyond a certain point. Three approaches are proposed to determine the best population size: (1) a random selection approach, (2) a detailed approach, and (3) a simplified approach. We used the detailed approach with our traffic problem and found that, with sufficient computational resources, the micro-GA performed best with population size around the square root of the string length. We tested this proposition on standard simple and deceptive problems and found it to hold true.

1 INTRODUCTION

Genetic algorithms (GAs) are emerging as a powerful tool for optimization of different engineering problems. Their increased use is attributed to their simplicity, minimal problem restrictions, global perspective, and implicit parallelism. However, implementing GAs for a real-world problem is difficult for several reasons. First, the user must select a combination of genetic operators (e.g., mutation, crossover, reproduction) and define values for several parameters (e.g., population size, mutation rate, crossover rate). For example, what population size should be used? What is the optimal crossover rate and mutation rate? What is the best selection scheme? What is the best crossover scheme? There are no established practical rules to optimally select these operators or their values. It is difficult for a GA user to determine which combination of these operators and parameter values will maximize the GA’s performance. It is possible that several combinations of these operators and parameter values would produce a fitness function that is pareto optimal in terms of computation cost and solution quality.

Second, the GA must identify a solution within a certain amount of computation time. This means that the GA will not be allowed to run until ultimate convergence but instead will be halted after a fixed amount of time. The best individual in the ending population will be selected as the final solution. Third, often the exact form of the fitness function is not known, which introduces additional noise that must be accounted for. The use of GAs is further complicated by their inherent internal variability, which must be accounted for in order to correctly select the best population size and number of generations. Among these issues, selecting a population size for a GA is a fundamental decision. Another important and closely related decision—once a population size is chosen—is the number of generations the GA should run.

Misleading, low-quality, and costly solutions may be obtained if the preceding issues are not addressed properly. This article addresses the question of population size used with a class of simple GAs known as micro-GAs (to be introduced later). The results presented in this article are based on our experience in optimizing traffic signal control systems formu-
lates as discrete-event, time-varying dynamic systems. Since there are no readily available analytical procedures to determine the optimal population size for micro-GAs, we opted for an experimental approach. First, we examine the convergence behavior of a micro-GA vis-à-vis population size. In particular, we want to see if we can determine the combination of population size and processing time such that the GA solution is optimal/near optimal. Also, we want to see what effect, if any, this choice has on the outcome. Finally, we propose a procedure that shows how to select the population size for a micro-GA taking into account the variability attributed to the particular choice of initial populations (henceforth referred to as internal variability).

This article deals with micro-GAs as opposed to regular, or full-sized GAs. Micro-GAs and regular GAs differ in the population size and the mechanisms both use to introduce and to maintain the genetic diversity necessary to reach optimal solutions. A micro-GA takes a small population (say $n$), converges it, keeps the best individual, randomly generates $n - 1$ individuals, converges the new population, and so on.

## 2 GENETIC ALGORITHMS

GAs are robust search and optimization techniques based on the mechanics of natural selection and natural genetics. GA-based optimization works by first coding the parameter space as $n$-bit chromosomes, where each chromosome represents a feasible solution, and then using the GA to generate an initial pool of solutions, represented as string structures ($n$-bit chromosomes). With the aid of genetic operators, this initial pool is allowed to evolve through a genetic selection process to eventually give a good solution. Better solutions, as determined by the values of the objective function (known as their fitness value), would be retained, whereas weaker ones are discarded. GAs differ from a conventional iterative improvement algorithm in at least three respects: (1) GAs use a population of points to start the procedure instead of a single design point. Since several points are used as candidate solutions, GAs are less likely to get trapped at a local optimum. (2) GAs do their search and optimization in parallel. The power of a GA derives from its ability to exploit, in a near-optimal fashion, information about the utility of a large number of structural configurations without the computational burden of explicit calculation and storage. This leads to a focused exploration of the search space wherein attention is concentrated in regions that contain structures of above-average utility. However, the population is widely distributed over the search space, insulating the space from susceptibility to stagnation at a local optimum. (3) GAs make much more extensive use of the information provided by the evaluation of candidates than most other heuristic search methods. For example, a hill-climbing algorithm tests several structures and keeps the most promising one. In the process, hill climbing discards a vast amount of information concerning the combinations of features present in the unsuccessful structures. In GAs, on the other hand, combinations of features in unsuccessful structures may still be passed along to other more successful structures.

### 2.1 Micro-genetic algorithms versus regular genetic algorithms

Regular GAs have been proven to be useful tools for many optimization problems. However, they have some limitations. A serious limitation of regular GAs is the time penalty involved in evaluating the fitness functions for large populations, particularly in complex problems. Goldberg noted that small populations could be used successfully with GAs (hence called micro-GAs) if the population is restarted a sufficient number of times. This is mainly due to the fact that smaller populations converge in fewer generations than do large populations. Thus it takes less time to evaluate a given generation than do large populations. A faster convergence in real time provides the opportunity to restart the micro-GA more often than the regular GA. Notwithstanding the computational value of this feature, it is very important to note that convergence does not necessarily mean that an optimal solution has been attained, as will be demonstrated later.

## 3 PROBLEM STATEMENT

The problem we are trying to solve is that of optimizing traffic flow along oversaturated arterial systems. The decision variables are the green times of arterial and cross-street approaches and the offsets.

Before we proceed, the problem is formulated as a dynamic discrete-event system. We decided to optimize it using micro-GAs because of their robustness and ability to effectively deal with complex dynamic problems. As with regular GAs, there are a number of decisions that we had to make regarding genetic operators (e.g., selection and crossover schemes to use) and the specific values that we must assign to the different parameters (e.g., what crossover rate, mutation rate, population size, etc. should be used). Even if we assume that all operators and parameter values, except the population size, have been optimally selected, there are a number of population size–related questions that we have to deal with. They include the following: What population size should we use? For how many generations should we run the system to obtain the optimal or near-optimal solution? Should we select a large population size and run it for fewer generations, or should we select a smaller population size and run the system for more generations? Is the gain in fitness at later generations worth the corresponding computation cost? Finally, can the answers to these questions be generalized?

Few studies have addressed the issue of population sizing.
All those studies dealt with regular GAs, not micro-GAs. De Jong\textsuperscript{4} experimented with different functions and noted that larger populations result in better ultimate off-line performance, whereas smaller populations are better for initial on-line performance. Goldberg\textsuperscript{9} provided a theoretical analysis of the optimal population size and noted that relatively small populations are appropriate for serial implementation and large populations are appropriate for perfectly parallel GAs. Goldberg and Rudnik\textsuperscript{13} gave the first population-sizing estimate based on the variance of fitness. Goldberg et al.\textsuperscript{11} derived a population-sizing equation to ensure that the best building blocks required to solve a problem are identified. The model gives a conservative bound on the convergence quality of GAs. The result of that study is a population-sizing equation that—for binary alphabets and ignoring external sources of noise—takes the form

\[ n = 2c\sigma^2 \frac{k}{m} m \]

where \( c \) is the square of the ordinate of a unit-normal distribution where the probability equals \( a, k \) is the order of the building block (BB), \( m \) is one less than the number of BBs in a string (\( m \)), \( \sigma^2 \) is the fitness variance of the partition, and \( d \) is the signal difference between the best and second best BBs. This equation is limited in that it only considers the decision in the first generation of the run. That is, the model assumes that if the decision favors the wrong BBs in the first generation, the GA will be unable to recover from the error. Also, if the decisions are correct in the first generation, the model assumes that the GA will converge to the right BB.\textsuperscript{15} This equation is very difficult to use in real-life, complex problems because acquiring sufficient knowledge of BBs and the relevant sources of noise may not be easy. Furthermore, this model does not apply for micro-GAs because it does not account for the key repetitive converge-and-restart feature of a micro-GA. Even if a similar equation is available for micro-GAs, the limitations noted still apply. Hence, for real-life problems, micro-GA users are still faced with the question of how to choose a suitable population size for their problems.

### 4 BACKGROUND

Applications of GAs and micro-GAs in engineering systems are numerous and diverse. A comprehensive account of these uses is beyond the scope of this article. What follows is a sample of the diverse applications and a brief account of how these studies addressed the question of population sizing and whether or not initial sampling variability was accounted for.

There are several applications of GAs in transportation engineering. Foy et al.\textsuperscript{7} reported on using GAs to determine traffic signal settings. A population size of 50 was used. Hadi and Wallace\textsuperscript{14} used GAs to obtain optimal phasing for signal coordination. They used a population 50. Fwa et al.\textsuperscript{8} and Chan et al.\textsuperscript{3} used GAs in the combinatorial problem of optimizing pavement maintenance. They experimented with different population sizes. For each population size they did three independent test runs. The best population was determined based on the average of the three runs. Memon and Bullen\textsuperscript{21} compared GAs with the quasi-Newton gradient method and chose GAs for real-time optimization of signals. They did not note the population size they used. Sadek et al.\textsuperscript{25} used GAs to solve a dynamic traffic assignment model. They used one population size of 30.

GAs also have been used in other engineering systems. Li et al.\textsuperscript{18} used GAs to handle difficult constraints and hence improve the efficiency of economic-environmental electric power dispatch. A population of 100 was used for 30 generations. Hossain et al.\textsuperscript{16} used a GA to develop an adaptive active control mechanism for vibration suppression. A population of 30 was used for 300 generations. Fisher\textsuperscript{5} used GAs to optimize the design of an engine block for low noise. Different population sizes were tried. In each case the algorithm was run five times, and the choice was based on the mean of fitness of the five runs. Zhao et al.\textsuperscript{26} used GAs to design multipoint connections in a local access network. Only one population size was used. Five runs were performed, each for up to 400 generations. Mansour et al.\textsuperscript{19} used GAs to design active sonar-associated gates. A population size of 100 was used. Powell and Skolnick\textsuperscript{22} used GAs in different engineering design optimization problems with nonlinear constraints. Fixed population numbers were used based on some rules suggested in earlier experiments. Martin and Knight\textsuperscript{20} used GAs for optimization of integrated-circuit synthesis. They tested different population sizes and used 3500 for their problem. It is not noted if variability was accounted for. Rhamani and Ono\textsuperscript{23} used GAs in the channel routing problem. They used one population of 50.

Few implementations of micro-GAs have been reported. Krishnakumar\textsuperscript{17} demonstrated the use of micro-GAs in solving nonstationary function optimization problems; a population of 5 was used. Roy et al.\textsuperscript{24} used an adaptive micro-GA to optimize the design-process variables in multipass wire drawing; a population of 6 was used. Abu-Lebdeh and Benekehah\textsuperscript{1} used micro-GAs to obtain a near-optimal solution for traffic signal coordination and queue management along oversaturated arterials; a population size of 5 was selected based on limited experimentation. In almost all the preceding applications, little or no rationale was given for the particular population size choice. And little or no effort was reported to account for internal variability.

### 5 STUDY APPROACH

The questions of determining the population size and number of generations to ensure convergence can be approached theoretically or experimentally. In this article we use an experi-
mental approach primarily because of the complexity of the problem under consideration, hence the difficulty in mathematically identifying the characteristics of the best solution (i.e., the appropriate building blocks). We experimented with different population sizes and observed the best fitness reached with each population size for a known number of function evaluations (FEs)—the number of FEs is equal to the product of population size and number of generations. Sufficient numbers of generations were run such that convergence to near-optimal values was apparent. The values to which the GA converged, as well as the time needed (in terms of FEs), were observed.

Variability in fitness values due to internal variability is also addressed. Multiple independent runs were executed with each population size such that our determination of the best population size is statistically sound. That is, we made a large enough number of independent runs to determine the best population size with a high confidence level. In each case, the coefficient of variation was calculated to see which population size would provide more consistent results over multiple runs.

5.1 Description of the problem

The problem we investigated in our experiments is that of optimizing a time-varying, discrete-event dynamic system. The system is a signalized arterial operating in oversaturated conditions. A detailed description of the system is available elsewhere.1 In this article we just present the mathematical formulation of the problem:

Maximize \[ \sum_{k=1}^{n} \sum_{j=1}^{n} g_{c(k)j}d_{i} \]

\[ + \left( j \times \sum_{k=1}^{n} \min \left\{ \left( q_{kij} + \frac{AV_{kij}}{g_{kij}-1} \times \left( g_{kij} - \frac{q_{kij}}{s_{j}} \right), AV_{kij} + q_{kij} \right) \right\} + \sigma + \omega \right) \]

where \( n \) is the number of cycles, \( m \) is the number of cross-street approaches, and \( j \) is the number of arterial links.

The first term of the objective functional measures the throughput of the crossing streets in link vehicles. The second term is the throughput of the arterial street in link vehicles. The preceding functional is maximized subject to the following constraints on control and state variables:

1. \( DV_{(k)i} \leq q_{(k)i} + DV_{(k)ij-1} \)
   \[ i = 1, \ldots, n; j = 1, \ldots, n \]
2. \( \text{off}_{(k)i.i+1} = q_{(k)i.i+1} + t_{i.i+1} \)
   \[ i = 1, \ldots, j = 1; k = 1, \ldots, n \]
3. \( \text{exoff}_{(k)i.i+1} = \text{Cyc}_{(k)i+1} - \text{off}_{(k+1)i.i+1} \)
   \[ i = 1, \ldots, j = 1; k = 1, \ldots, n \]
4. \( g_{(k)i.i+1} + \text{off}_{(k)i.i+1} + t_{i.i+1} \geq g_{(k)ij} \)
   \[ i = 1, \ldots, j = 1; k = 1, \ldots, n \]
5. \( g_{(k)i.i+1} = q_{(k)i} - DV_{(k)i} + DV_{(k)ij-1} \)
   \[ i = 1, \ldots, j = 1; k = 1, \ldots, n \]
6. \( \sigma = \sum_{i=1}^{n} q_{(k)i} \times (i-1) \)
   \[ i = 1, \ldots, ; j = 1, \ldots, n \]
7. \( \omega = \sum_{i=1}^{n} q_{(n+1)i} \times (i-1) \)
   \[ i = 1, \ldots, j = 1, \ldots, n \]
8. \( g_{(k)i} \leq \text{maximum value} \)
   \[ i = 1, \ldots, ; j = 1, \ldots, n \]
9. \( g_{(k)i} \leq \text{maximum value (key cross-approach)} \)
   \[ i = m - 1, m; k = 1, \ldots, n \]
10. \( q_{(k)i} \) are given
    \[ i = 1, \ldots, j \]

The definitions of the symbols used in the formulation are as follows:

- \( s_{i} \) adjusted saturation flow rate per lane of approach \( i \), vehicle/second
- \( LV \) average effective vehicle length when stopped, ft
- \( L_{i} \) length of link \( i \), ft
- \( L_{c(k)i} \) storage on receiving link \( i \) at the beginning of cycle \( k \), ft
- \( t_{i.k+1} \) time for acceleration wave to reach tail of queue when signal turns green for approach \( i+1 \) at the start of cycle \( k \), seconds
- \( t_{i.i+1} \) time for stopping wave to reach tail of queue when signal turns red for approach \( i+1 \) when link \( i+1 \) is full, seconds
- \( t_{q(k)i.i+1} \) time for front of queue \( i \) to join rear of queue \( i+1 \), seconds
- \( v_{i} \) speed of an acceleration wave (a shock wave) on link \( i \), ft/s
- \( \lambda_{i} \) speed of a stopping wave (a shock wave) on link \( i \), ft/s
- \( \text{Cyc}_{(k)i} \) length of cycle \( k \) at intersection \( i \), seconds
- \( g_{(k)i} \) effective green time for arterial approach \( i \) during cycle \( k \), seconds
- \( g_{c(k)i} \) effective green time for cross-approach \( i \) during cycle \( k \), seconds
- \( DV_{(k)i} \) discharged vehicles from approach \( i \) during cycle \( k \), vehicles
- \( AV_{(k)i} \) arriving vehicles into approach \( i \) during cycle \( k \), vehicles
- \( q_{(k)i} \) length of queue on approach \( i \) at the beginning of cycle \( k \), vehicles
Convergence variability and population sizing in micro-genetic algorithms

5.2 Experimental setup

We investigated the preceding problem for a five-intersection arterial system and a study period of 10 cycles. A micro-GA was used with the following genetic operators and parameter values: uniform crossover with a rate of 0.5, no mutation, and a binary tournament selection scheme. Based on the values and number of decision variables, the string length was 260 bits. Based on the constraints used on the decision variables, there were over $10^{56}$ possible solutions for this problem. In conditions such as this, GAs were found to provide very acceptable solutions within a practical period of time.\(^\text{10}\)

First, six different population sizes (3, 6, 9, 12, 15, and 25) were used. With each population size, the algorithm was run until convergence was observed. Then the convergence behaviors for these populations were examined. The results are discussed in the following section.

6 DISCUSSION OF RESULTS

In the following subsections, first, we look at convergence behavior versus population size. Second, we show how to properly account for internal variability. Third, we examine the influence of computational resource levels on the choice of population size. Fourth, we look at the degree of dispersion of fitness values around the mean versus different population sizes. Fifth, we address issues related to fitness improvement versus computation costs. As a point of clarification, fitness value means the value of the objective function. In the context of this article, it is the traffic throughput measured in link vehicles.

6.1 Micro-GA convergence versus population size

Figure 1a shows the convergence of the micro-GA in a typical run. The graph shows that convergence has been attained for all population sizes but not necessarily to the same fitness value. It is clear that determination of the optimal population size is influenced by the number of generations. For example, the ranking of populations (at the end of 10,000 generations) from best to worst is 25, 15, 9, 6, 12, and 3. However, this ranking changes several times before it stabilizes around generation 1000. This change in ranking is, in part, attributed to the fact that the algorithm has not converged yet. This point becomes clearer in Figure 1b. Here, we can visually determine when the algorithm has converged (it is the point where the fitness line starts becoming relatively flat). In all cases the algorithm seems to be converging in the range of 2000 to 3000 generations. Therefore, comparison of algorithm performance with different population sizes—and for that matter, any other parameter—must take this fact into account. That is, for a given population size, there is a minimum number of generations below which characteristics of algorithm performance may be unstable. Another important point Figure 1b demonstrates is that convergence may occur without reaching the optimal or a near-optimal solution (note convergence with population size 3). Therefore, users must not rely on convergence as the sole criterion for good GA performance.

Another important point to note is that the preceding ranking of population sizes and their corresponding fitness values change from run to run. This is shown in Figure 2. The figure shows the fitness obtained at 10,000 FEs for three independent runs. The change in fitness values from run to run—and hence the inability to determine which is the best population size—is very clear. This is due to the internal variability of GAs.

The results just presented point out three important issues. First, determination of the best population size cannot be made based on one or few runs. Second, deciding on which is the best population size is a function of the computational
resources used. Third, convergence may be attained without reaching an optimal solution.

Needless to say, then, in order to properly address the issue of population sizing, we have to (1) account for internal variability, (2) consider the effect of computational resources available, and (3) look beyond just convergence as a sign of “good GA behavior.”

6.2 Population size versus internal variability

For the purposes of this article, internal variability is defined as the variation in fitness value caused by changes in the initial population. In different simulation runs, we change the initial population to account for any biases that may be introduced by the initial population. That is, we want to make sure that the final solution is not influenced by the specific initial population. In other words, the starting point in the search space should not determine the final solution. In effect, changing the initial population changes the starting point and “route” that the micro-GA takes to get to the optimal solution. To be reasonably certain of the “best” population size, one must account for this internal variability. Enough number of runs must be made so that we are confident of the fitness associated with each population size and, hence, which population size is best.

For a given level of confidence and tolerable error, one may use the following equation to determine the required sample size (i.e., number of runs) assuming the fitness values are normally distributed:

\[
    n = \left( \frac{ts}{\epsilon} \right)^2
\]

where

- \( n \) = required sample size
- \( s \) = standard deviation
- \( \epsilon \) = user-specified allowable error
- \( t \) = coefficient of the standard error of the mean that represents user-specified probability level

We used a confidence level of 95% and an allowable error of 2% of the mean fitness observed. We started with few runs and, based on the observed fitness variance, increased the number of runs until a sufficient number of runs was made to establish the confidence level desired. It was determined that 18 runs would be sufficient, but we decided to use 20 runs to be conservative.

Based on multiple runs, we can estimate the mean fitness value of a given population size and hence decide which population size is likely to lead to the best performance. However, as noted earlier, this determination depends on how long the algorithm is allowed to run. This point is addressed next.

6.3 Optimal population size and computational resources

In Section 6.1 we showed that the choice of population size is related to the available computation resources. In this section we try to determine the optimal population size at three computational resource levels: limited, midlevel, and generous. These levels correspond to 5000, 10,000, and 30,000 FEs, respectively. Figure 3 shows the mean fitness value of the 20 runs for each population size at the three resource levels.

The figure shows that with limited computational resources, a population of 25 gives the best fitness. This superior performance is explained in part by the initial larger genetic diversity this larger population has. For generous computational resources, on the other hand, a population of 12 is the preferred choice. The superior performance of population size 12 is a clear example of how smaller population sizes, if given enough running time, can outperform larger population sizes. The reason for this is that it is quicker for smaller populations to converge, hence enabling the algorithm to explore larger numbers of points in the search space. For midlevel computational resources, both populations of 12 and 25 are about equally effective.

The preceding observations are particularly relevant if we are running an on-line application. For an off-line application, a large number of function evaluations can be carried out. On
the other hand, for on-line applications, our concern is to get to acceptable performance quickly. Thus, for an off-line application of our problem, we would use a population of 12. For on-line applications, we would use a population of 25, because the solution has to be located quickly.

Figure 3 also demonstrates that the common notion that larger populations are better is not always true. For example, at 10,000 FEs, a population size of 12 is better than a population size 15, and at 30,000 FEs, a population of 12 performed better than a population of 25. In other words, the optimal, or best, population size can vary depending on how long the micro-GA is allowed to run. That is, the optimal population size is a computation time-dependent variable. On the other hand, for all ranges of computation time, performance of the smaller population sizes of 3, 6, and 9 is consistently lower than that of the other population sizes. The performance of the smallest population size (3 in this example) is, by a wide margin, worse than that of all other population sizes regardless of computation time.

These results are consistent with what we know of how micro-GAs work. The larger population introduces larger initial “genetic” diversity, hence more of the necessary building blocks. This initial advantage makes larger populations a better choice when computation time is limited. Smaller (but not very small) populations, if allowed to run for a sufficient number of generations, can narrow this lead through the repeated converge-and-reinitialize process typical of micro-GAs. This is why we note that after sufficient computation time, 30,000 FEs, for example, the performance of all but the populations of 3 and 6 are comparable.

6.3.1 Ranking of performance. To further examine the relative performance of each of the 6 population sizes tested, we looked at the performances in each of the 20 runs. The results are presented in Table 1, which shows the ranking of the 6 population sizes based on the mean fitness values of the 20 runs. The rankings are based on how each population size performed in each of the 20 runs. For example, for a given run, the population that performs best (i.e., gives the best fitness) is given a rank of 1; the one that performs worst is given a rank of 6. These ranks are then summed for the 20 runs and shown in the “Sum of Ranks” column of Table 1. A low sum of ranks means the population performed well overall, and vice versa.

In general, the results shown in Figure 3 and Table 1 indicate that (1) the common notion that larger populations are better is not necessarily true, (2) the choice of a very small population (3 in our case) is not a good choice irrespective of the computational resources spent, and (3) the performance of midsized populations seems to become comparable with that of larger populations with increased computational resources.

For practical applications, this means that larger populations should be used when computation time is limited. For generous computational resources, smaller populations can be a better choice depending on exactly how much computational resources are available. Also, for a given computational resource level, the extra payoff due to increasing the population size is not always positive. For example, for limited computational resources, increasing the population size consistently leads to improvement in the quality of the solution. However, if the computational resources are not bound, the quality of the solution improves as the population increases up to a certain population size, beyond which either the quality of the solution does not improve (with increased population) or it may even be lower.

6.3.2 Best population size and number of generations. Figure 4 shows the fitness values versus number of generations. The broken lines join points of equal computational resources, and the solid lines connect points of similar population size. The superior performance of larger populations with smaller generation numbers is very clear. For example, moving from left to right on the 5000-FE line (from larger populations to smaller ones) we note a sharp rate of decline in fitness value with the increase in number of generations. This is due to the advantage of higher genetic diversity larger populations have at low computational resource levels. With larger numbers of generations, we note that the rate of decline of fitness value as we move from larger to smaller populations is becoming “gentler.” This is so because as available computational resources increase, smaller populations can have an opportunity (through quicker and repeated convergence

Table 1

<table>
<thead>
<tr>
<th>Computational resources</th>
<th>Population size</th>
<th>Sum of ranks (20 runs)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited (5000 FEs)</td>
<td>25</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>58</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>71</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>82</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>119</td>
<td>6</td>
</tr>
<tr>
<td>Midlevel (10,000 FEs)</td>
<td>12</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>52</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>72</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>118</td>
<td>6</td>
</tr>
<tr>
<td>Generous (30,000 FEs)</td>
<td>12</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>54</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>63</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>68</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>78</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>119</td>
<td>6</td>
</tr>
</tbody>
</table>
and reinitialization) to narrow the lead that larger populations had at smaller numbers of generations. This explains why smaller populations (but not 3 and 6 in our example) can perform almost as well as larger populations if they are run for a sufficient number of generations.

6.4 Dispersion of fitness values around the mean

So far we have been basing all judgments and observations on the mean fitness value for each population. However, it is not sufficient to just observe the mean fitness and then decide which population size is best. At least from the results of Section 6.3, there is enough reason to be concerned about how the different observations (fitness values from the different runs) are distributed around the mean fitness. In other words, should we be concerned about the extent of variability of the fitness values associated with each population size? And what are the practical implications of this?

Before answering these questions, we need to note that in real life when we apply GAs to solve an optimization problem, we use the outcome of one run, or one observation, which can be any distance (measured in fitness units) from the mean. Therefore, it is important to select a population size such that the observations are as clustered around the mean as possible, that is, a population with low variability outcome. A population with such a property is more desirable because the results of different runs will be more consistent (i.e., less diverse), and hence we are less likely to end up with a solution that is much different from the mean. The following material addresses this issue in some detail.

6.4.1 Frequency and distribution of fitness values. One way of assessing variability and dispersion of observations is through frequency plots (Figure 5). First, we note that for all levels of computational resources, fitness values obtained with population size 9 are more concentrated around the mean than other population sizes (also, lowest standard deviation). This means that if obtaining consistent results is critical, then our best bet is using population size 9. Second, for midsized populations of 9, 12, and 15 (12 not shown), the dispersion of results does not change much with the level of computational resources. Also, these populations had the lowest standard deviations for all computational resource levels. This implies that using these populations is more likely to produce consistent results. That is, for a given run using these populations, we are more likely to obtain a solution that is closer to the mean, and over a number of independent runs, we are more likely to get similar solutions. Furthermore, this is so regardless of the level of computational resources used.

6.4.2 Coefficient of variation of fitness values. Another way to assess the degree of variability is to look at the coefficient of variation (CV), which is the ratio of the standard deviation to the mean. A CV closer to zero is desirable because it indicates little or no variation between different samples. Figure 6 shows the CV as a function of population size and computation time. We note that for the three levels of computational resources, the midsized populations of 9, 12, and 15 consistently exhibit lower variation than very high and very low population sizes. It is interesting to note that population size 9 in our experiment consistently (i.e., regardless of computation time) had the lowest CV.

As the reader can see, some rules of thumb are emerging. If we combine the results on variability with the findings on
Convergence variability and population sizing in micro-genetic algorithms

329

the relation between population size and expected fitness, we note that some population sizes provide a good compromise between fitness and variability, whereas others are good in one but not the other area. For example, in our problem, each of the three populations 9, 12, and 15 is a good compromise between fitness and variability for the case of generous computational resources. Among the three populations, population size 12 gives the best mean fitness and a relatively low variability and hence would be our choice. With limited computational resources, the populations 9, 12, 15, and 25 exhibit a similar level of variability. However, population size 25 gives the best mean fitness and hence would be our choice in this case.

6.5 Population size, fitness improvement, and computation cost

Computation cost is another important issue that is related to fitness value. If we go back to Figures 3 and 4, a not-so-obvious point in these two figures is that for the same population size, the increased fitness value may not be worth the added computation time. It is clear that for a given population size, the more generations we run, the more improvements we will see in the fitness value. However, other than the variability issue noted in the preceding section, the relative cost of these improvements becomes progressively higher with the increase in computation time, as clearly demonstrated in Table 2. For each population–computational resource level combination, the table shows the proportion of the best fitness that can be obtained with that combination. For example, for population size 12, there was only a 6.8% increase in the fitness value for a 100% increase in computation time (5000 FEs compared with 10,000 FEs) and a 14% increase in fitness for a 500% increase in computation time. For population size 25, only a 9.3% fitness improvement was obtained for a 500% increase in computation time. It is unlikely that these marginal improvements in fitness values are worth the increase in computation cost. Therefore, the choice of how many generations to run the algorithm should be based on the practical value of the improvement in the objective function and the cost of the associated computation time. This is a particularly important consideration for real-time applications or applications where function evaluation happens to be very costly.

The preceding point also can be demonstrated across populations, as shown in Figure 7. This figure is generated from one run for each population size by mapping the fitness values at 10,000 generations. The scale used varies from 0 to 100, with the best fitness (in this run for population size 25) mapped to 100 and all others as percentages of it. Similarly, the number of FEs (completed at 10,000 generations) is also mapped on the same scale. The largest number of FEs, that of population size 25, mapped to 100 and all others as percentages of it. Figure 7 shows that while the improvement in fitness changes modestly from populations 6 to 25, the associated computation time increases at a significantly higher rate. For example, with a population size of 9, one can get 92% of the fitness value obtainable with a population of 25 but only at 27% of the computation cost (measured in FE units). Therefore, we may substitute smaller populations for larger ones and obtain most of the worth of the solutions ob-

---

**Table 2**

Percentages of best fitness obtained for each population size

<table>
<thead>
<tr>
<th>Population size</th>
<th>5000 FEs</th>
<th>10,000 FEs</th>
<th>30,000 FEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>83.6%</td>
<td>90.6</td>
<td>100.0</td>
</tr>
<tr>
<td>6</td>
<td>86.1</td>
<td>92.8</td>
<td>100.0</td>
</tr>
<tr>
<td>9</td>
<td>86.5</td>
<td>92.1</td>
<td>100.0</td>
</tr>
<tr>
<td>12</td>
<td>85.7</td>
<td>92.5</td>
<td>100.0</td>
</tr>
<tr>
<td>15</td>
<td>89.8</td>
<td>94.2</td>
<td>100.0</td>
</tr>
<tr>
<td>25</td>
<td>90.7</td>
<td>94.5</td>
<td>100.0</td>
</tr>
</tbody>
</table>
tained with larger populations, but at a substantially lower cost. This, of course, will depend on the application at hand. For example, for an on-line application, the time savings between populations 9 and 25 are very significant and may be more than enough to offset the 8% loss in fitness value.

7 PROCEDURES FOR DETERMINING POPULATION SIZE

The results presented to this point show that the interrelated issues of population size and number of generations, fitness variability, and computation cost can have significant practical implications to the users of micro-GAs. Some of the results do apply for regular GAs as well. Given the numerous possible combinations of different operators and parameter values (including population size), and given the complex stochastic nature of GAs, it may not be possible, or even necessary, to provide a "cook book" analytical approach to using GAs. However, there should be at least some guidelines. The following subsection discusses some possible approaches to determining the optimal population size for micro-GAs.

7.1 Population sizing

Depending on how critical the quality of the final solution is, the user may approach the question of population size in different ways. Here we propose three approaches; we label them random, detailed, and simplified.

7.1.1 Random choice of population size. A quick and simple way is to pick a small population size, say, between 3 and 7, and run the GA until convergence is attained. Convergence can be ascertained either visually by plotting the resulting fitness against the number of generation or when the ratio of fitness improvement to number of function evaluations (FEs) falls below a certain cutoff value. This value will have to be decided by the user. Using the random choice approach to decide the population sizing, one runs the risk of not getting as high a quality solution as would be the case if a more rigorous approach is followed. A clear example of this is the use of population size 3 in our experiment (see Figure 1, for example).

7.1.2 The detailed approach. The detailed approach is the one we used in this article. The idea in this approach is to experiment with several population sizes at the same time to take a large enough sample (i.e., number of independent runs) to estimate the mean fitness of each population size with enough confidence. This approach is as follows:

1. Decide on a starting population size. Pick a population size around $\sqrt{L}$, where $L$ is the string length (for binary coding), and make at least 5 runs. In each run start with a different, independently generated initial population. Use a large enough number of generations until the micro-GA converges (i.e., until little or no fitness improvement is observed). In this case, one might want to start with a very large number of generations and then cut the number of generations where convergence is apparent.

2. Decide on a range to locate the best population size. Use a range between 3 and $1.5\sqrt{L}$; then pick several populations from this range. For each population size, do at least 5 independent runs and in each ensure convergence. For each population size, start with the same random seeds as in step 1.

3. Assess internal variability, and determine the need for additional runs. Based on fitness variability of each of the populations identified in steps 1 and 2, and based on the desired confidence level and tolerable error, determine the number of additional runs needed. If no additional runs are needed, go to step 5.

4. Do the additional runs; then go to step 3.

5. Calculate the mean fitness values associated with each population size, and examine the variability of observations (i.e., runs) by calculating the coefficient of variation (CV).

6. Decide on the best population size. The population with the highest mean fitness and the lowest CV is the best population size for the problem at hand. If the population size with the highest mean fitness is different from the population with the lowest CV, then more than one population size should be considered. The final choice in this case will depend on which is more valued to the user: the fitness value or the consistency of results between different runs.

The detailed approach suggested here ensures that the neighborhood where the best population lies is identified, and hence the population size identified thereby, is a near-optimal one. It does not, however, guarantee that the exact best population size is identified. It should be noted, though, that it is difficult to recommend a population size for all regular or micro-GA problems. The optimal size may vary depending on a number of factors such as the GA coding, genetic operators, values of other parameters, and nature of the problem at hand. In GA terminology, the optimal population size depends on the number of building blocks in the chromosome, order of the building blocks, cardinality, and fitness variance of the initial population.

7.1.3 The simplified approach. This approach is motivated by and builds on the outcome of the detailed approach. It goes as follows:

1. Determine a range of populations approximately between $0.4\sqrt{L}$ and $1.4\sqrt{L}$, where $L$ is the string length.
2. Use a population that is roughly equal to $\sqrt{L}$. This should be a good start with different levels of computational resources. However, if you have low computational resources, use a population in the upper part of the preceding range. Use the lower part of the range if you have generous computational resources.

The random and detailed approaches are not problem-dependent. However, the simplified approach might be problem-dependent. In particular, it is still important to examine the impact of the type of scaling (i.e., the manner by which the value of any bit is factored, e.g., linear, exponential, etc.) on the choice of populations size. This point is not covered in this article.

8 COMPARISON OF MICRO-GA AND REGULAR GA PERFORMANCE

So far we have shown the ability of a micro-GA to find near-optimal solutions for the combinatorial traffic control problem. In this section we compare the micro-GA performance with that of a regular GA for the same problem.

For the micro-GA we used a population of 12. For the regular GA we sized the population according to the relation derived in Goldberg et al.\textsuperscript{11}:

$$n = O\left(\frac{L}{k}\left(\frac{2^k}{k}\right)^2\right)$$

where $n$ is the population size, $L$ is the string length, and $k$ is the average size of the schema of interest, effectively, the average number of bits per parameter. For our problem, the string length is 260, and the number of bits per parameter is either 2 or 4. We used a population of 400 with a simplified version of the traffic problem (narrower search space) and ran both GAs for 48,000 FEs. We mapped the best fitness of the two GAs to 100 and all others as a percentage of it. Figure 8 shows a plot of the results. The real difference between the two GAs is in the early stage of the run (shown in part a), where the micro-GA had a chance to converge and restart the small population several times before the regular GA even completed evaluating the first generation. By the time the regular GA gave its first evaluation result, the micro-GA had reached 98% of the best fitness. The value of the initial "edge" of the micro-GA would be at a premium for on-line applications. But once both GAs have run long enough, their performance is comparable.

For the traffic control problem presented in this article, the micro-GA performed better than the regular GA. Others have reported up to a fourfold improvement in computation time with micro-GA's.\textsuperscript{2,17} It is still not clear what types of problems would benefit from using a micro-GA and what types would be more suited to regular GAs. To answer this question, micro-GAs and regular GAs have to be tested on a wide range of problems. This is the subject of ongoing work and will be reported at a later time.

9 MICRO-GA PERFORMANCE WITH EASY AND DECEPTIVE PROBLEMS

The apparent success of the micro-GA with the combinatorial traffic control problems is clear. For that problem we do not know the exact shape of the functions and do not know the optimal solution. So we had to visually assess the performance and convergence of the micro-GA. In this section we set out to see how micro-GAs would perform in difficult and easy problems. We also try to see if the population-sizing relationships derived earlier are applicable.

9.1 Micro-GA in difficult problems

To test the performance of the micro-GA with a difficult problem, we used the deceptive function described in Goldberg et al.\textsuperscript{12} It is a 30-bit, order-six bipolar function that is concatenated from the order-five, 6-bit bipolar function shown in Figure 9. This test function, with over 5 million local optima and 32 global optima, was shown to present a difficult challenge to GAs. Goldberg et al.\textsuperscript{12} showed that a regular GA can find one of the 32 global optima if a large enough population is used.
In this section we use a micro-GA to find at least one of the global optimum. We test the micro-GA performance with different population sizes with an eye on determining the best population size. As a criterion to compare performance of the different populations, we use time (in FEs) needed to reach a global optimum. We used uniform crossover with a rate of 0.5, no mutation, and binary tournament selection.

For each of the populations used, the micro-GA successfully identified one global optimum. The time it took to reach the global optima differed with each population tested. Figure 10 shows the mean number of FEs to reach the first global optimum for each of the four populations tested. The number of FEs shown is the mean of 33 independent runs.

Two points can be made based on Figure 10. First, the micro-GA does not do well with too small a population. Although with population 3 it reached a global optimum, it did so after a significantly longer search compared with the other populations. The second point is that too large a population is not a much better choice.

There appears to be a population size with which the micro-GA is likely to do better than others. The results shown in Figure 10 indicate that this population size is 5. Note that this population size is around the value of the square root of the string length (\(\sqrt{30} \approx 5.5\)). This concurs with our earlier findings for the traffic problem. In that case, the micro-GA also performed best with a population size around the square root of the string length. These results lead us to believe that a micro-GA performs best when the population is sized to be around the square root of the string length. To further test this proposition, we test the micro-GA on a very simple function.

### 9.2 Micro-GA in easy problems

Here, we test the micro-GA on the following one-max, 30-bit string problem:

\[
\text{Maximize } \sum_{i=1}^{J} x_i \quad x_i \in \{0, 1\}
\]

The one-max function counts the number of bits set to 1 in the string and uses that value as the fitness of the individual. This is a simple function with one global optimum. It occurs when all string digits are 1’s. For string length 30, the maximum value of this function is 30. We compare the performance of the micro-GA with different population sizes. We measure the time (in FEs) to reach the global optimum. Figure 11 shows the mean number of FEs to reach the global optimum. The number of FEs shown is the mean of 33 independent runs.

Figure 11 illustrates two points. First, as was the case with the difficult problem, too small a population did not perform very well compared with other populations. The second point is that there appears to be a population size below which the micro-GA does not perform well. The micro-GA performed equally well with populations greater than or equal to this “lower bound.” Although the micro-GA performed best with
population size 5, the differences between population sizes 5, 8, and 12 are negligible. Thus, for the easy problem, we can reasonably state that a population size around the square root of the string length is a lower bound on the population size that should be used with micro-GAs. This finding is consistent with what we observed with the preceding problems.

10 CONCLUSIONS AND RECOMMENDATIONS

This article addresses the question of how to determine the best population size when using micro-GAs. As part of answering this question, several related issues are addressed. Specifically, the article addresses the issues of micro-GA convergence behavior versus population size, criteria for determining best population size, micro-GA internal variability, and micro-GA performance versus computational resources. The article also proposes some empirical approaches to determine the best population size.

For a given population size, the article shows that there is a minimum number of generations below which characteristics of algorithm performance may be unstable, and hence the determination of the best population size at such a stage may not be appropriate. It is also shown that convergence may occur without reaching the optimal or a near-optimal solution. Hence users must not rely solely on convergence as a criterion for good GA performance. Results show that to determine the best population size, users may need to examine more than a single criterion. In addition to the fitness value at convergence that is used normally, and as a second criterion, the article proposes the use of consistency of the final fitness values obtained by independent runs with the same population size. As well, results show that because of internal variability, determination of the best population size cannot be made reliably based on one or a few runs. It is critical to account for internal variability when determining the best population size. In the experimental runs for this study, the mid-sized populations consistently exhibited lower variation than very high and very low population sizes.

The article shows that answering the question of optimal population size is a function of the computational resources used and that at most one can determine only a near-optimal population size. For a given computational resource level, it is shown that the common notion that larger populations are better is not always true. Smaller populations with generous computational resources were found to perform as well as, and sometimes better than, larger populations. But even when larger populations perform better, we found that we can still use smaller populations and obtain most of the worth of the solutions obtained with larger populations, but at a substantially lower cost. In addition, for a given population size, more computational resources will lead to better fitness values. However, the increased fitness value may not be worth the added computation cost. Therefore, the choice of how many generations to run the algorithm should be based on the practical value of improvement in the objective function and the cost of the associated computation time.

The article proposes three approaches to determine the best population size: (1) a random selection approach, where a small population size is picked at random, (2) a detailed one, where several population sizes are tested with enough rigor so as to establish statistical confidence, and (3) a simplified, “middle-ground” approach that combines simplicity with some depth. The detailed and random approaches are not problem-dependent. The simplified approach applies to problems with uniformly scaled bits and hence may still need to be tested on problems with different scaling schemes. In general, with sufficient computational resources, the micro-GA performs best when the population is sized around the square root of the string length (for binary coding). This proposition was tested on real-world as well as standard simple and difficult problems. It held true in all cases.

It is recommended that micro-GA users use the best population size in order to ensure optimal solutions to their problem, regardless of which criterion is used to determine the best population size. Because of internal variability, it is recommended that better estimates of the mean fitness associated with each population size be obtained using larger samples (i.e., number of independent GA runs).

REFERENCES

4. De Jong, K. A., An analysis of the behavior of a class of genetic adaptive systems, Ph.D. dissertation, University of Michigan, Ann Arbor, 1975; noted in Goldberg.10