Complex Systems **9** (1995) 47–72

Adaptively Resizing Populations: Algorithm, Analysis, and First Results

Robert E. Smith*
Department of Engineering Science and Mechanics,
The University of Alabama,
Box 870278, Tuscaloosa, Al 35487 USA.

Ellen Smuda
Department of Aerospace Engineering,
The University of Alabama,
Tuscaloosa, Al 35487 USA

Abstract. Deciding on an appropriate population size for a given genetic algorithm (GA) application can often be critical to the success of the algorithm. Too small, and the GA can fall victim to sampling error, affecting the efficacy of its search. Too large, and the GA wastes computational resources. Although advice exists for sizing GA populations, much of this advice involves theoretical aspects that are not accessible to the novice. This paper suggests an algorithm for adaptively resizing GA populations. The algorithm is suggested based on recent theoretical developments that relate population size to schema fitness variance. The algorithm is developed theoretically, simulated with expected value equations, and tested on a problem where population sizing can mislead the GA. The positive results presented suggest that adaptively sizing GA populations may be a practical extension to the typical GA. Such an extension frees the user from a critical parameter decision, and expands the usefulness of GA search. Moreover, this extension creates a new, interesting class of genetic search systems, where adaptive changes in population size reflect problem complexity.

1. Introduction

A novice genetic algorithm (GA) user that sets out to apply a GA to an optimization problem faces a series of decisions. First, it is discovered that the problem must be encoded into a GA amenable representation. Next, the

*Electronic mail address: rob@comec4.mh.ua.edu

user must choose selection, recombination, and mutation operators. Finally, the user must choose a set of GA parameters, typically including a crossover probability (p_c), mutation probability (p_m), and population size (N).

For the purpose of this discussion, one can make the assumption that, for a given set of parameters, a GA with a given set of operators is sufficiently robust to cope with a variety of problem encodings. This, of course, ignores the possible dilemma of deception in certain encodings, the relative merits of various encoding alphabets, and the different effects of various operators. However, GAs are generally robust, and this assumption allows us to focus on tunable aspects of a given GA, rather than the wide variations of GA implementations.

Given these assumptions, the key decisions of the user are parameter settings. For setting p_c and p_m, some heuristic guidance exists. Crossover probabilities between 0.6 and 0.9 are recommended. Mutation rates should be between $1/N$ and $1/\ell$ (where ℓ is the encoding length). In practice, GAs are typically robust enough to cope with some variation in the settings of these two parameters. Experience shows that, within the bounds of these heuristics, variations in these parameters result in relatively small changes in GA performance. However, variations in population size can have substantial effects on GA performance on many problems. Too small a population, and the GA can fall prey to sampling error. Too large a population, and computational effort is wasted on extra fitness evaluations.

Some advice exists for sizing populations. In early papers on this topic, Goldberg develops formulae and indicative plots for population sizing [1, 2]. These results are based on some limiting assumptions about the necessary levels of schemata sampling in the GA. A more recent paper [3] presents a formula for sizing populations that is directly based on schemata fitness variance. This formula directly indicates the necessary schema sampling for given schema statistics.

The formulae in [3] are useful, but they do present a difficulty for the novice. Without an understanding of the GA theoretic concepts of schemata and their statistics, the developments are inaccessible. Therefore, from the viewpoint of a novice user, the GA may not seem robust at all. They may comment, "I want to use the GA, but I don't want to have to understand it." Thus, what appears to be a robust algorithm to an experienced GA user may not be robust to the world at large.

One might comment that all optimization schemes suffer from this difficulty. To use them adequately for a broad range of problems, there must be some understanding of the interactions of the parameters, and thus, some understanding of the algorithm. However, parameters used in traditional search algorithms are often easy for a novice user to access and understand. These parameters are typically a desired accuracy for the search process or its end result, and a constraint on the computational resources (time and memory) that the user is willing to expend. In other words, the user need only answer: "How good do I want my answer to be, how long do I want to spend, and how much memory do I have in which to work?" The GA suffers

from the difficulty of having parameters (like population size) that are much more difficult to relate to user needs.

This paper is the first outcome of a project to develop a GA whose parameters are more user accessible. The project's current efforts are concentrated on the population sizing decision. This seems a logical first step, since population sizing is a decision the user must make in any GA, and it is a decision on which the novice has little accessible advice. The desired end result of the project is the development of a GA where the user need only specify a desired accuracy parameter, a computational time constraint, and a memory constraint. This new GA system will adaptively resize the population in an effort to meet user requirements. The remainder of this paper will present a suggested technique for dynamically resizing populations given these parameters. The technique is first developed theoretically, simulated in expectation, and then tested on a problem where population sizing is critical to GA performance.

2. Algorithm development

To design an algorithm for adaptively resizing populations, one must form a basis for resizing and design a procedure that exploits that basis. The following sections undertake that task.

2.1 Population sizing theory

The technique presented here is based on recent suggestions on population sizing from [3]. Thus, a brief summary of these developments is provided.

Theory suggests that GAs search by implicitly evaluating the mean fitness of various schemata based on a series of population samples, and then recombining highly fit schemata. Since the schemata average fitness values are based on samples, they typically have a nonzero variance. Consider the competing schemata:

$$H_1 = * \ * \ * \ * \ 1 \ 1 \ 0 \ * \ * \ 0$$

and

$$H_2 = * \ * \ * \ * \ 0 \ 1 \ 0 \ * \ * \ 0$$

Assuming a deterministic fitness function, variance of average fitness values of these schemata exist due to the various combinations of bits that can be placed in the "don't-care" (*) positions. This variance has been called *collateral noise* [4]. Let $f(H_1)$ and $f(H_2)$ represent the average fitness values for schemata H_1 and H_2, respectively, taken over all possible strings in each schema. Also let σ_1^2 and σ_2^2 represent the variances taken over all corresponding schemata members.

The GA does not make its selection decisions based on $f(H_1)$ and $f(H_2)$. Instead, it makes these decisions based on a sample of a given size for each schema. We call these observed fitness values $f_o(H_1)$ and $f_o(H_2)$. Observed

fitness values are a function of $n(H_1)$ and $n(H_2)$, the number of copies of schemata H_1 and H_2 in the population, respectively. Given moderate sample sizes, the central limit theorem[1] tells us that f_o values will be distributed normally, with mean $f(H)$ and variance $\sigma^2/n(H)$.

Due to the sampling process and the related variance, it is possible for the GA to err in its selection decisions on schema H_1 versus H_2. In other words, if one assumes $f(H_1) > f(H_2)$, there is a probability that $f_o(H_1) < f_o(H_2)$. If such mean fitness values are observed the GA will incorrectly select H_2 over H_1. Given the $f(H)$ and σ^2 values, one can calculate the probability of such an error based on the convolution of the two normals. This convolution is itself normal with mean $f(H_1)-f(H_2)$ and variance $(\sigma_1^2/n(H_1))+(\sigma_2^2/n(H_2))$. Thus, the probability that $f_o(H_1) < f_o(H_2)$ is α, where

$$z^2(\alpha) = \frac{(f(H_1) - f(H_2))^2}{(\sigma_1^2/n(H_1)) + (\sigma_2^2/n(H_2))},$$

and $z(\alpha)$ is the ordinate of the unit, one-sided, normal deviate. Note that $z(\alpha)$ is, in effect, a signal-to-noise ratio, where the signal in question is a selective advantage, and the noise is the collateral noise for the given schemata competition.

For a given z, α can be found in standard tables, or approximated. For values of $|z| > 2$ (two standard deviations from the mean), the gaussian tail approximation can be used:

$$\alpha = \frac{\exp(-z^2/2)}{\left(z\sqrt{2\pi}\right)}.$$

For values of $|z| \leq 2$, one can use the sigmoidal approximation suggested in [5]:

$$\alpha = \frac{1}{1 + \exp(-1.6z)}.$$

Given this calculation, one can match a desired maximum level of error in selection to a desired population size. This is accomplished by setting $n(H_1)$ and $n(H_2)$ such that the error probability lower than the desired level. In effect, raising either of the $n(H)$ values "sharpens" (lowers the variance of) the associated normal distribution, thus reducing the convolution of the two distributions.

In [3] it is suggested that if the largest value of $2^{o(H)}\sigma^2/|f(H_1) - f(H_2)|$ is known for competitive schemata of order $o(H)$, one can conservatively size

[1]Technically, the central limit theorem only applies to a random sample. Therefore, the assumption that the mean of observed, average fitness values are the same as average fitness values over all strings is only valid in the initial, random population, and perhaps in other populations early in the GA run. However, GA theory makes the assumption that selection is sufficiently slow to allow for good schemata sampling. This common assumption is made in this work as well.

the population by assuming the $n(H)$ values are the expected values for a random population of size N. This gives the sizing formula

$$N = z^2(\alpha)2^{o(H)}\frac{\sigma_1^2 + \sigma_2^2}{(f(H_1) - f(H_2))^2}.$$

Particular versions of this formula for various problem configurations and levels of deception are considered in [3]. These developments are not considered here, but may have implications for later developments of the suggested adaptive resizing algorithm. Important heuristics are also suggested in [3].

The formula presented above is a thorough compilation of the concepts of schemata variance and its relationship to population sizing. However, it does present some difficulties from the viewpoint of robustness for the novice. The values and ranges of $f(H)$ are not known beforehand for any schemata, although these values are implicitly estimated in the GA process. Moreover, the values of σ^2 are neither known nor estimated in the usual GA process.

In addition, the formula does not consider the relative importance of schemata competitions. If two competing schemata have fitness values that are nearly equal, the overlap in the distributions will be great, thus suggesting a large population. However, if the fitness values of these schemata are nearly equal, their importance to the overall search may be minimal, thus precluding the need for a large population on their account. To compensate for this effect, one could consider the absolute expected selection loss due to an error in selection $L(H_1, H_2)$, as opposed to the probability of such an error:

$$L(H_1, H_2) = |f(H_1) - f(H_2)|\alpha(H_1, H_2).$$

The technique suggested here will attempt to dynamically size a population based on matching the selection loss L to a desired target for this loss L_t provided by the user. To do so, one must estimate schemata variance, a topic taken up in section 2.2.

2.2 Estimating variance and selection loss

The standard GA estimates schema mean fitness values through repeated, probabilistic selection and recombination of strings, based on the fitness value of each single string. The fitness of a single string that belongs to a schema is the smallest sample possible for evaluating a meaningful average fitness value for that schema. Clearly, this minimal sample cannot be relied upon to deliver an adequate average fitness. However, the GA uses a population of strings to implicitly deliver the schema average fitness for a larger sample, while never explicitly evaluating the average of the sample. Also, under many selection schemes, the GA makes selection decisions over competing schemata in small increments (small changes in population proportions), thus distributing the evaluation of schemata averages fitness values over time. This spatially and temporally distributed evaluation is key to a GA's *implicit parallelism* [7], as well as its easy *explicit* use on parallel computers. The distributed approach

Table 1: A mated pair of sample strings and their fitness values.

	string	fitness
S_1	0 1 0 1 0 0 1 1 1 0 1 0	100
S_2	1 0 0 1 0 1 1 0 0 1 0 1	50

also maintains the GA's naturally inspired character. In developing a method for estimating variance, it would be desirable to maintain a similar approach.

Clearly, one cannot evaluate an estimate of schema variance based on a single member of that schema. To evaluate a variance, one must consider at least two points. Consider the fitness variance of a pair of strings, in particular, the variance of mates. What schemata variances does one learn about by observing the variance in fitness between two strings? In a deterministic problem, any variance is caused by the bits where the two strings differ. Thus, the variance in fitness of two strings gives an estimate of the fitness variance of the schemata that the strings have in common. For example, consider the strings and fitness values in Table 1. The strings S_1 and S_2 share schema

$$H_{1,2} = **010*1*****$$

and indicate that an estimate of its average fitness is $f(H_{1,2}) = (f(S_1) + f(S_2))/2 = 75$. The indicated variance is given by

$$\sigma^2_{1,2} = \frac{(f(S_1) - f(H_{1,2}))^2 + (f(S_2) - f(H_{1,2}))^2}{2} = \frac{(f(S_1) - f(S_2))^2}{4} = 625.$$

Although this is a very crude estimate of schema fitness variance, it is no more crude than the estimate of schema average fitness implicit in the usual GA.

Note at this point that almost all children of a pair of strings will share the same common schemata as their parents. Only children who mutate in the defined bits of the shared schema will fail to share this schema with their parents. This fact will prove useful in the algorithm in section 2.3.

To size populations based on the developments in [3] competing schemata must be considered as a basis for calculating α. To do so, we consider a competition of pairs of schemata. Recall that each pair can be used to gain variance information on their common schemata. When two such pairs are compared, information is obtained about fitness differences that result from any bits that are common within each pair, and are in common bit positions in the competing pairs. These are the competitive schemata indicated by a competition of pairs. For instance, consider the strings S_1 and S_2 in Table 1, competing as a mating pair against the strings in Table 2, arranged as another mating pair. Strings S_3 and S_4 share schema

$$H_{3,4} = 1**00*1*****$$

Table 2: Another mated pair of sample strings and their fitness values.

	string	fitness
S_3	1 1 1 0 0 0 1 1 1 1 1 0	152
S_4	1 0 0 0 0 1 1 0 0 0 0 1	68

and indicate that $f(H_{3,4}) = 110$ and $\sigma^2_{3,4} = 1764$. The schemata $H_{1,2}$ and $H_{3,4}$ share defined positions 4, 5, and 7. Thus, the competition of string pair S_1 and S_2 against string pair S_3 and S_4 gives information about the following two competitive schemata

$$H'_{1,2} = ***10*1*****$$

$$H'_{3,4} = ***00*1*****$$

both of which are in partition [8]

$$J_{1,2,3,4} = ***\mathtt{ff}*\mathtt{f}*****.$$

Thus, from this information, one can begin to obtain an estimate of α, and thus L, for the schemata indicated by this competition of pairs. To complete this calculation, one must determine the number of copies in the current population for each of the two competing schemata under consideration, $n(H'_{1,2})$ and $n(H'_{3,4})$. Discussion of methods for obtaining these counts and their computational expense will be deferred to the conclusion of this paper. For now, assume that such a count can be obtained at reasonable computational expense.

Given these factors, a competition of two pairs can be used to calculate an estimated selection loss:

$$\hat{L}(H_{1,2}, H_{3,4}) = |f(H_{1,2}) - f(H_{3,4})|\alpha(H_{1,2}, H_{3,4}),$$

where $\alpha(H_{1,2}, H_{3,4})$ is calculated using either the sigmoidal or gaussian tail approximations, and an estimated z value, given by

$$z = \frac{(f(H_{1,2}) - f(H_{3,4}))}{\sqrt{(\sigma^2_{1,2}/n(H'_{1,2})) + (\sigma^2_{3,4}/n(H'_{3,4}))}}.$$

The following section will show how this estimated selection loss can be used to adaptively resize populations.

2.3 An adaptive population sizing algorithm

Consider the following steps to adaptively resize populations. Assume that the user has supplied a desired target value for acceptable selection loss L_t. Start with a small, randomly initialized population, and repeat the following steps until population size and distribution stabilizes.

1. Randomly put population members together in mating pairs.
2. Determine the shared schemata in each mating pair.
3. Put together competitions of mating pairs.
4. Determine the competitive schemata for each mating pair competition.
5. Obtain a count of each competitive schema.
6. Calculate $\hat{L}$ for each mating pair competition.
7. Use $\hat{L}/L_t$ as a basis for assigning more (or fewer) population members.
8. Construct new population members by crossover and mutation of selected mating pairs.

As previously noted, almost all of the children of a given mating pair will have the common schema from that pair. Thus, one can expect that the schema survival relationships of the typical GA will extend to the common schemata in the suggested algorithm.

It is desired that the suggested process converge to a population where expected selection losses match the target value L_t. Consider the following use of $\hat{L}/L_t$ values. Each $\hat{L}/L_t$ value is mapped to an expected one step growth rate for the associated mating pair competition, via a sigmoid function:

$$G = (1 - \gamma) + \frac{2\gamma}{1 + \exp\left(-\beta\left(\frac{\hat{L}}{L_t} - 1\right)\right)}$$

where γ is a desired, maximum, expected percentage increase (or decrease) per generation, and β is a parameter. This growth factor can be mapped to any one of a variety of selection schemes [9].

Of course, this represents the introduction of two new parameters, which may defeat the original purpose of this work: to increase ease of GA use. However, if robust heuristics can be determined for these parameters, ease of use will not be affected. In the simulations presented in section 3, $\beta = 1$ and $\gamma = 0.9$ yielded an effective population size.

The G factor is associated with all four strings in the mating pair competition. One should ask how this factor should be divided between the two competing mating pairs. $\hat{L}$ gives no information on which schemata's numbers should be increased in the given competitive partition. For lack of a better strategy, one can inject a degree of selection by dividing the change in number of the schemata under consideration based on relative fitness. For instance, if $G_{1,2,3,4}$ reflects a competition of mating pair S_1, S_2 against S_3, S_4, then

$$G_{1,2} = \left(1 - \frac{f(H_{1,2})}{f(H_{1,2}) + f(H_{3,4})}\right) + \left(G_{1,2,3,4}\frac{f(H_{1,2})}{f(H_{1,2}) + f(H_{3,4})}\right)$$

and

$$G_{3,4} = \left(1 - \frac{f(H_{3,4})}{f(H_{1,2}) + f(H_{3,4})}\right) + \left(G_{1,2,3,4}\frac{f(H_{3,4})}{f(H_{1,2}) + f(H_{3,4})}\right)$$

where $G_{1,2}$ and $G_{3,4}$ are respective growth factors for the two mating pairs.

Given a one step growth rate for each mating pair, new copies can be assigned. The $\hat{L}/L_t$ values for each mating pair competition will effect population growth through this selective process. When $\hat{L}/L_t$ is greater than one (for a given mating pair competition), population size will increase (with respect to that competition). When $\hat{L}/L_t$ is less than one, population size will decrease. When $\hat{L}/L_t$ values are equal to one, population growth will cease. When population growth ceases, or population size becomes relatively stable, the regular GA can begin to select purely based on fitness. This divides the GA selective process into two distinct phases. First is a selection based on variance, with variable population size. Next is selection based on fitness, with a fixed population size. This is a clear division of exploration and exploitation phases. Later revisions of the algorithm will require a gradual transition between these phases. Viewed in this light, the suggested algorithm bears some resemblance to the exploration-exploitation strategies for reinforcement learning problems suggested in [10].

When population size adjustment ceases in the suggested scheme, the proportions of various schemata will not be evenly distributed in any given partition. However, the fitness biasing used to divide growth factors should help to insure that high fitness schemata in any given partition will be likely to have a higher number of copies.

3. Computational simulation

The suggested algorithm's behavior can be simulated by considering a single partition of competitive schemata. Before doing so, it is useful to illustrate the sampling error effects that can occur due to finite population size. This can be accomplished in simulation by considering eight competing schemata in an order three partition. Consider $f(H)$ and σ values for the eight schemata in Table 3. The values in Table 3 were selected to show a variety of combinations of fitness and variance for simulating the population resizing scheme.

Iterating proportion equations [11] will show the expected performance of fitness proportionate selection on these schemata:

$$P_i^{t+1} = \frac{f_i P_i^t}{\sum_j P_j^t f_j}$$

for all i and j in the competitive partition. However, this expected value model does not consider variance effects. As an abstraction of actual fitness

Table 3: Mean fitness values and variances for eight schemata.

schema	$f(H)$	σ
1	1.00	6.4
2	0.90	13.2
3	0.80	3.2
4	0.70	10.0
5	0.65	16.4
6	0.95	2.0
7	0.80	20.0
8	0.72	0.4

proportionate selection under sampling noise, consider iteration of a noisy proportion equation:

$$P_i^{t+1} = \frac{f_i' P_i^t}{\sum_j P_j^t f_j'}$$

where the f' values are given by the f values with zero-mean gaussian noise added. The variance of this noise is given by the σ values listed in Table 3, divided by appropriate counts of the corresponding schema. To simulate a count of each schema, assume a population size N, and let $n(H_i) = P_i N$. The proportion values will also be rounded to reflect the simulated finite population size.

Figure 1 shows typical results of iterating the noisy proportion equations with $N = 1024$. The highest fitness schema quickly takes over the population. This result occurs consistently with this population size. However, with $N = 64$, the results are quite different. Figure 2 shows a typical run. As shown in this example, the population often fails to converge to the highest fitness schema. The propensity of the proportion equations to miss the highest fitness schema increases with decreasing population size. Figure 3 shows a convergence histogram for 25 runs for $N = 64$. This histogram shows the proportion of each schema at the end of 100 generations, averaged over the 25 runs. In all 25 runs, some schema overtakes nearly the entire population (well over 98%). The histogram indicates the schema that overtakes the population is often not the most fit schema. Figures 4 and 5 show similar histograms for $N = 128$ and $N = 32$, respectively. Clearly, sampling error can cause selection error, and associated failure to converge to the highest fitness schemata in a partition. Since selection error is a function of population size, the population resizing algorithm should reduce selection error to a point where correct convergence occurs.

The f and σ values listed above are used in an expected value model to simulate the action of the suggested population resizing algorithm. Note that this simulation considers the probability of any two schemata i and j coming into competition as $P_i P_j$.

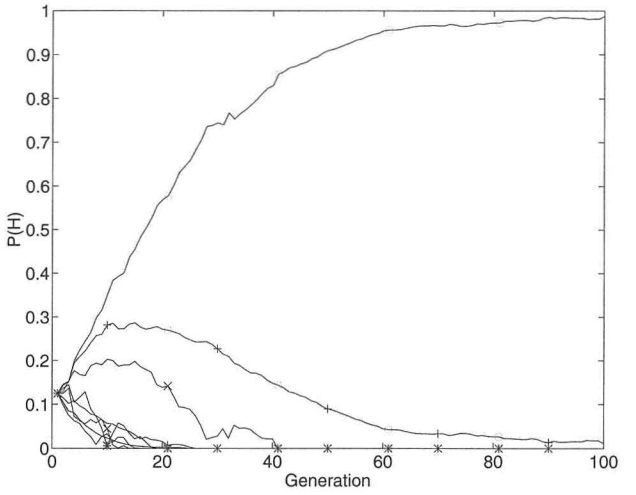

Figure 1: Simulated evolution of schema proportions under selection with noisy fitness values ($N = 1024$). Symbols are as follows: –o=H_1, –x=H_2, – +=H_3, –*=H_4, o–x=H_5, o–+=H_6, o–*=H_7, and —=H_8.

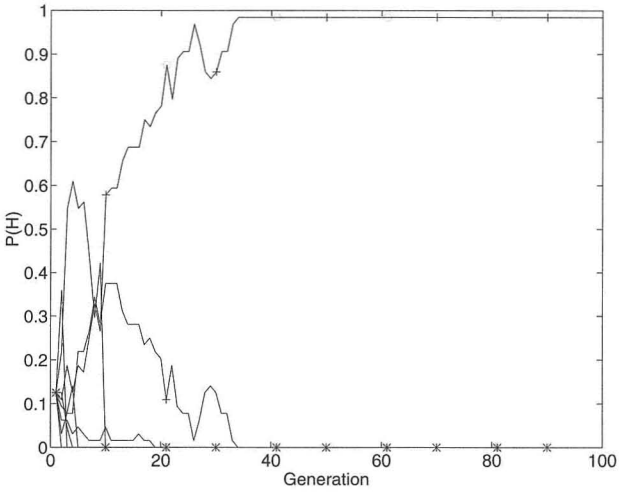

Figure 2: Simulated evolution of schema proportions under selection with noisy fitness values ($N = 64$). Symbols are as stated in Figure 1.

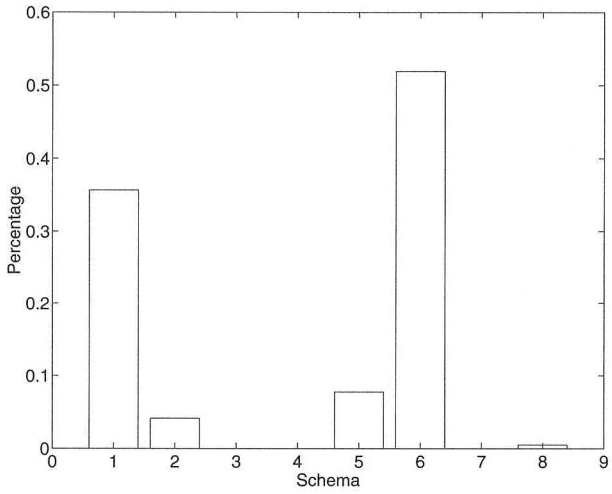

Figure 3: Histogram indicating the converged schemata distribution with each percentage averaged over 25 simulations under selection with noisy fitness values ($N = 64$).

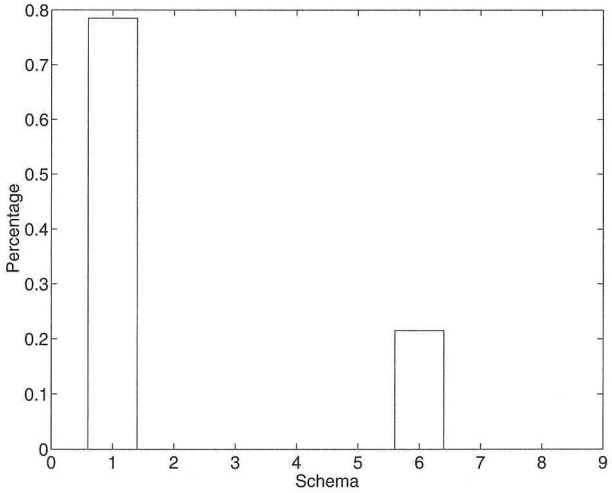

Figure 4: Histogram indicating the converged schemata distribution with each percentage averaged over 25 simulations under selection with noisy fitness values ($N = 128$).

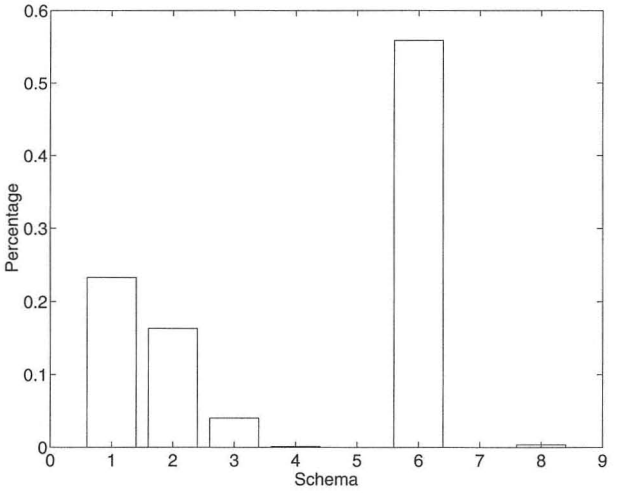

Figure 5: Histogram indicating the converged schemata distribution with each percentage averaged over 25 simulations under selection with noisy fitness values ($N = 32$).

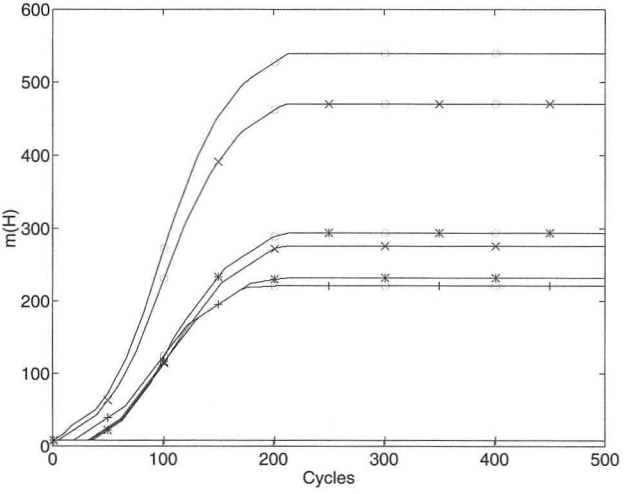

Figure 6: Evolution of numbers of schemata in simulation of the population resizing scheme. Symbols are as stated in Figure 1.

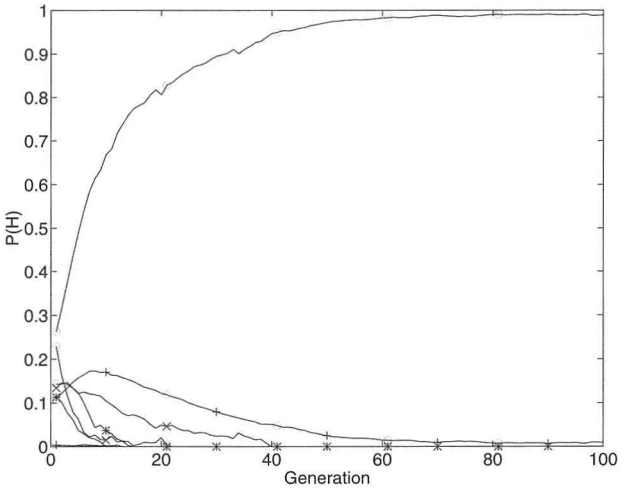

Figure 7: Evolution of population proportions starting from the resulting population size and proportions from the population resizing scheme, under selection with noisy fitness values. Symbols are as stated in Figure 1.

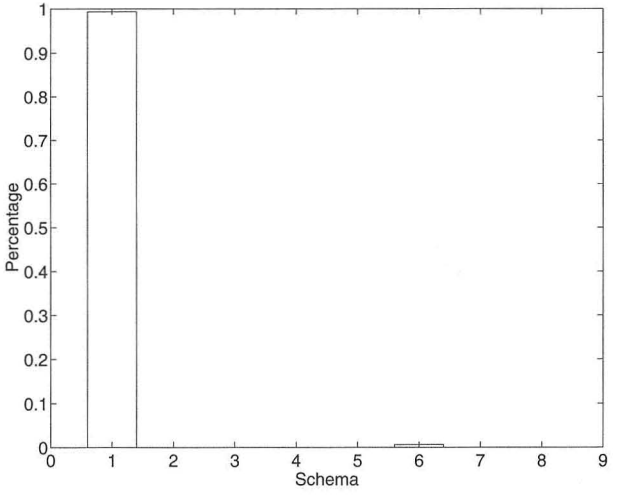

Figure 8: Histogram of converged schemata distribution with each percentage averaged over 25 simulations starting from the resulting population size and proportions from the population resizing scheme, under selection with noisy fitness values.

Given a starting population size of $N = 64$, with equal initial numbers of each schema in the competitive partition, Figure 6 shows the results of simulating the population resizing scheme. The final population size from this simulation is 2048.

Given the resulting proportions and population size at cycle 500 in Figure 6, one can iterate the noisy proportion equations to simulate selection after population resizing. Figure 7 shows the results of these iterations. Note that the simulation quickly converges to the highest fitness schemata. Figure 8 shows a histogram of the convergence of 25 similar runs. In all of these runs, the most fit schema overtakes the population. Small proportions of other schemata shown in the histogram are only due to the persistence of these schemata at near zero proportions at the end of some of the runs.

4. Implementation

The results presented in section 3 are only those of simulations. They certainly do not reflect the complexities of a live GA run with the suggested population sizing scheme. In particular, the schema fitness and variance values used in the simulation are assumed to be explicitly given, as opposed to estimated from the stochastic process inherent to a real GA. However, such expected value simulations have been useful in past GA studies [11, 12]. The simulations only consider the competitive schemata in a single partition. In a real GA, there will be an interplay of many partitions of schemata, some of which indicate that the population should be larger, some of which indicate that the population should be smaller. The results of the simulations used here indicate that the suggested scheme may be an effective method for automatically resizing populations. This section presents the next phase of testing the method: its implementation in a real GA.

The interplay of multiple schemata competitions in the real GA led to somewhat different dynamics than those of the simulations. In particular, the variations in population size were broader than desired. Often the population would expand to a much larger value than was necessary for a given test problem (see below). This expansion seems to be due to the conservative nature of the loss measure. Recall that expansion is based on the estimated expected loss,

$$\hat{L}(H_{1,2}, H_{3,4}) = |f(H_{1,2}) - f(H_{3,4})|\alpha(H_{1,2}, H_{3,4}).$$

The system must obtain the given level of target loss for *every* schemata competition, which is difficult in the real GA, and introduces broad, sustained oscillations in the population size. To reduce these oscillations, the criteria is relaxed:

$$\hat{L}(H_{1,2}, H_{3,4}) = \frac{|f(H_{1,2}) - f(H_{3,4})|}{n(H'_{1,2}) + n(H'_{3,4})}\alpha(H_{1,2}, H_{3,4}).$$

In effect, this weights the importance on any one selection loss less severely for larger population sizes. This seems logical in connection with the real

GA, since it is less critical for the GA to select correctly every time in a given competition if that competition occurs several times in the population.

In implementing the algorithm simulated in previous sections, we must more seriously consider the transition from population sizing to actual selection. In the simulations, the transition was made abruptly, after the population size had obviously stabilized. While evaluating stability is easy in an expected value simulation, it is difficult in an actual GA, where the complex interplay of various schemata make the convergence of population size less obvious. Although the population size may converge in expectation, it will still vary around the mean in the real GA. This effect makes the decision to switch to pure selection difficult. Rather than attempting to use some statistical measure to make this decision (and possibly introducing more tunable parameters), the implementation presented here folds selection and population resizing into one continuous operation.

In the implementation presented here, growth or decay is accomplished by a procedure that is similar to stochastic remainder selection [13]. First, the growth rate is multiplied by two. If the growth rate for a mating pair is $2 \cdot G_{1,2}$, the pair is given the integer part of $2 \cdot G_{1,2}$ children deterministically, and an extra copy with probability equal to the fractional part of $2 \cdot G_{1,2}$. If the growth rate for a particular competition $G_{1,2,3,4}$ is greater than or equal to one, the growth rate is divided between the two mating pairs based on relative fitness, as was suggested earlier. If $f(H_{1,2}) = f(H_{3,4})$ and $G_{1,2,3,4} < 1$, the growth rate is also divided as was indicated in previous sections. However, if the growth rate $G_{1,2,3,4}$ is less than one, *all of the losses are assigned to the less fit mating pair.* In other words, if $f(H_{1,2}) > f(H_{3,4})$ and $G_{1,2,3,4} < 1$, the growth rate is divided as follows:

$$G_{1,2} = 1,$$
$$G_{3,4} = G_{1,2,3,4}.$$

The motivation for this method is based on the population sizing theory discussed previously. A growth rate of less than one indicates that the variance level of the schemata competition at hand is such that selection can be performed with some confidence in the results. If the algorithm is started with a small population size, this condition will not occur frequently until late in the run, when the population size is corrected downward. Thus, the algorithm should have two distinct phases, with population expansion and little selection early on, and population contraction with selection later.

In an actual implementation of the algorithm, another important decision is how to set the mutation rate. Since the mutation rate is usually tied to population size in a GA, there is no clear way to set a fixed value for the mutation rate in a GA where population size varies. In this implementation, the mutation rate is held at $1/N$, where N is population size, throughout the run.

Since the algorithm suggested is based on competitions of four strings at a time, it is helpful for the population size to remain a multiple of four. In the implementation presented here, this is accomplished as follows. If the

new population size is not a multiple of four, a random decision is made as to whether to add or subtract individuals. This prevents a bias towards population expansion or reduction. If subtraction is selected, the last few individuals are deleted from the population. If addition is selected, individuals are randomly created. In either case, the resulting population size is the nearest multiple of four.

In addition to the previously discussed details, we must also consider that small values of L_t are likely to dictate population sizes that are larger than are practical. Thus, user-dictated memory and computational time limitations must be added to the algorithm. In the implementation given here, individuals are randomly deleted when a maximum population size is exceeded. There may be better strategies for such deletion, including deletion based on fitness. Such strategies are an avenue for future study. Another important extension of this implementation is a user-specified limitation based on computational time. Implementing such a limitation will require estimates of GA convergence time for a given population size, and corresponding adjustments of the maximum population size. Previous studies of convergence times for GA selection schemes may be helpful in this extension [9].

5. Testing

This section formulates a test problem for the resizing technique, and demonstrates the technique's effects on this problem. It will be useful to employ a test problem where it can be demonstrated that the results of a fixed-population size GA vary with the selection of the population size. To do this, it is necessary to insure that mutation alone is not sufficient to (eventually) locate the problem's optima. If mutation is sufficient, then the eventual convergence of the GA to the optima is assured. Population size may affect transient response of the GA, but this would not be sufficient for the desired illustration of the population sizing scheme's effects. Because of this, it is necessary to have at least a partial degree of *deception* in the problem [13], so that order one building blocks alone cannot lead to the optima.

Consider the following problem:

$$\text{fitness } = f(x) = \begin{cases} m_1 x + b_1 & \text{if } x < F_2 \\ m_2 x + b_2 & \text{otherwise} \end{cases}$$

where x is the interpretation of the GA bit string as an integer, scaled to a real number between zero and one, and $m_1 = -0.533$, $b_1 = 0.5$, $m_2 = 16$, $b_2 = -15$, and $F_2 = 15/16$. A plot of this function is shown in Figure 9. The deception in this function can be shown by considering the four order one schemata competitions represented by the four most-significant bits in x, listed in Table 4. The function is partially deceptive, since the first two schemata competitions have selective pressure towards the false optima ($x = 0$), while the last two have selective pressure towards the true optima ($x = 1$). Moreover, the order four competitive partition in the most-significant bits is not deceptive, as is shown in Table 4. If disruption of the schema `1111`

Table 4: Approximate order one schema average fitness values for the partially deceptive function.

schema H	approximate average fitness $f(H)$
0*** ...***	0.3672
1*** ...***	0.1679
*0** ...***	0.2987
*1** ...***	0.2359
0* ...*	0.2668
1* ...*	0.2683
0 ...	0.2501
1 ...	0.2683

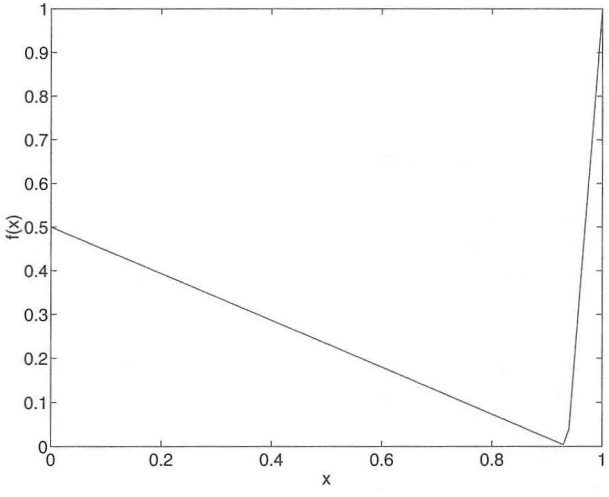

Figure 9: A partially deceptive function that the GA can solve.

...*** is low, it would be expected that the GA would select this schema over others in its partition, and the process would converge to the correct optima.

However, it is easy to imagine the GA being misled by the function plotted in Figure 9. If high order bits are set to zeros (through disruption of the high-order schema or selection error), the remainder of the string will improve overall performance by moving away from the true optima at $x = 1$. However, the linkage in the high order building blocks is tight, and the GA is not usually misled by this function, even with relatively small population sizes, as Figures 10 and 11 illustrate. These figures show GA results for

Table 5: Approximate order four schema average fitness values for the partially deceptive function.

schema H	approximate average fitness $f(H)$
0000 ...***	0.4843
0001 ...***	0.4509
0010 ...***	0.4174
0011 ...***	0.3840
0100 ...***	0.3506
0101 ...***	0.3171
0110 ...***	0.2837
0111 ...***	0.2502
1000 ...***	0.2168
1001 ...***	0.1833
1010 ...***	0.1499
1011 ...***	0.1164
1100 ...***	0.0830
1101 ...***	0.0496
1110 ...***	0.0161
1111 ...***	0.5241

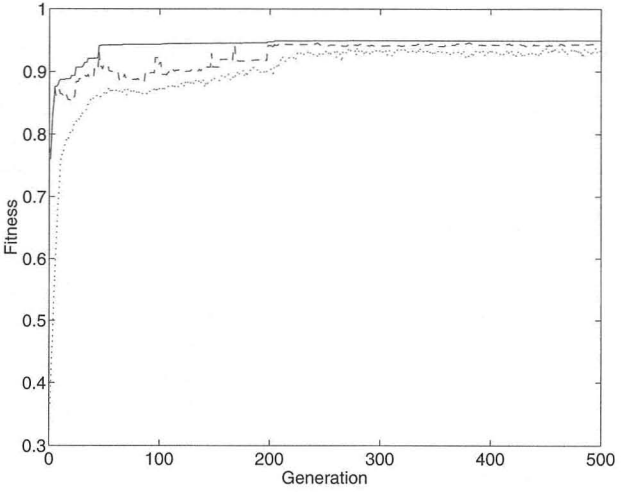

Figure 10: Average results for ten simple GA runs with population size 50 on the partially deceptive problem. The best fitness individual located thus far is shown with a solid line, maximum fitness in the current generation is shown with a dashed line, and average fitness in the current generation is shown with a dotted line.

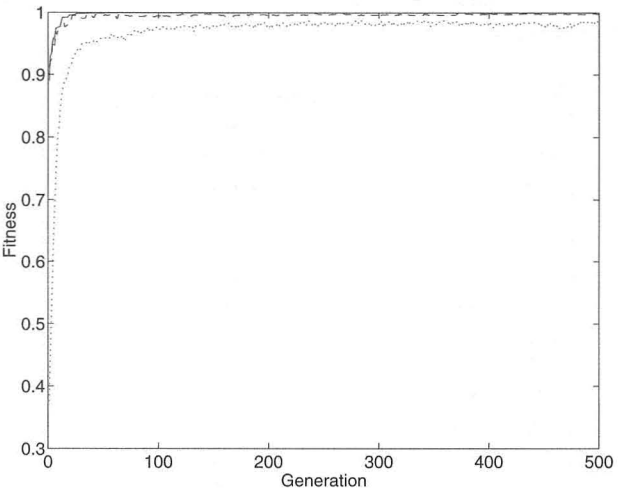

Figure 11: Average results for ten simple GA runs with population size 100 on the partially deceptive problem. The best fitness individual located thus far is shown with a solid line, maximum fitness in the current generation is shown with a dashed line, and average fitness in the current generation is shown with a dotted line.

populations of size 50 and 100 (respectively), with bit strings of length 21, crossover rate $p_c = 0.6$, mutation rate $p_m = 0.001$, and tournament selection with tournaments of size two [9]. The results shown are averaged over ten independent runs. For a population size of 50, the GA converges to the true optima nine out of ten times. For the population size of 100, the GA consistently converges to the true optima. Qualitatively similar results were obtained with other parameter settings.

Given a partially deceptive problem that the GA can solve (with appropriately linked building blocks), we decrease the selective signal-to-noise ratio, using the previously demonstrated problem, concatenated with a 21-bit, bitwise linear problem. In other words, consider interpreting the first 21 bits of the string as x in the previous problem, and each of the remaining bits as a separate variable x_i. The suggested fitness function is

$$\text{fitness} = 0.1 \cdot f(x) + \frac{0.9}{21} \cdot \sum_{i=1}^{21} x_i.$$

In effect, the previously discussed partially deceptive function is added to the simplest of all problems for the GA. However, the results are not what might be expected. Figures 12 and 13 illustrate the effects of GAs with various population sizes on this problem. Figure 12 shows the best individual found up to the current generation and Figure 13 shows the maximum individual

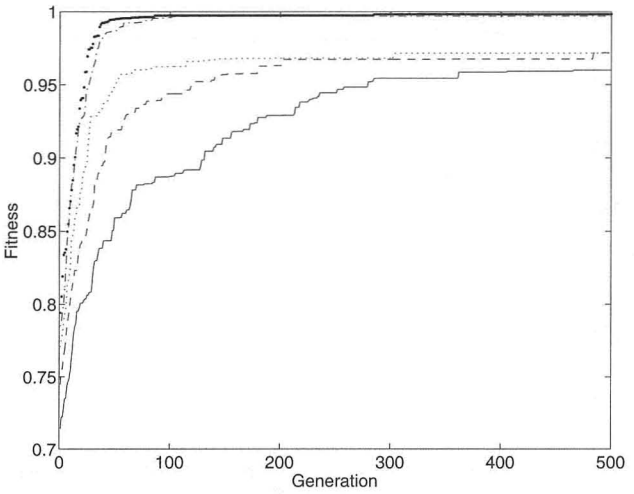

Figure 12: Plots showing the best individual found up to the current generation for various population sizes on the modified partially deceptive problem. Plots represent an average of ten runs. Population size 50 is shown with a solid line, 100 with a dashed line, 200 with a dotted line, 500 with a dashed-dotted line, and 1000 with a large-dotted line.

in the current population. Other parameters are the same as in the previous example. Each line on these plots represents an average of ten GA runs. Note that the function $f(x)$ represents the final 10% of fitness in these experiments, and that if the GA is misled to $x = 0$, the maximum fitness is 0.95. With small population sizes, this happens often. As the population size is increased, the GA more frequently finds the correct solution. Qualitatively similar results were obtained for different parameter settings.

The failures in these runs are caused by the low signal-to-noise ratios for the critical, higher order bits of parameter x. In effect, for small populations the GA initially concentrates its selective decisions on the most important bits. These bits are the 21 x_i. The higher order bits of x are carried along in these selective decisions, often with incorrect values. These bits are what have been called *hitchhikers* in the GA theoretic literature [14]. In effect, hitchhikers are the result of insufficient population size.

Given a problem where population sizing can be seen as critical to overall performance, the effects of adaptive population sizing can be examined, with a foundation for interpreting effects. Figures 14 and 15 show the results of a representative run of the adaptive population sizing code. This run used the parameters shown in Table 6. Results of similar, independent runs were qualitatively similar. The GA consistently sized the population in a similar

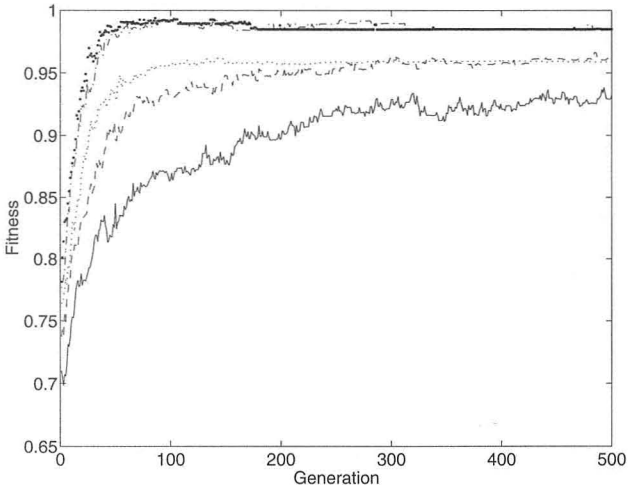

Figure 13: Plots showing the maximum individual in the current generation for various population sizes on the modified partially deceptive problem. Plots represent an average of ten runs. Population size 50 is shown with a solid line, 100 with a dashed line, 200 with a dotted line, 500 with a dashed-dotted line, and 1000 with a large-dotted line.

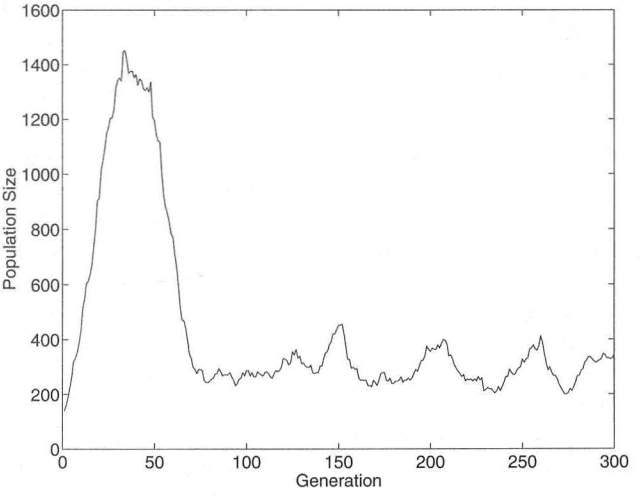

Figure 14: Population size versus generation for the adaptive population sizing GA applied to the modified partially deceptive function.

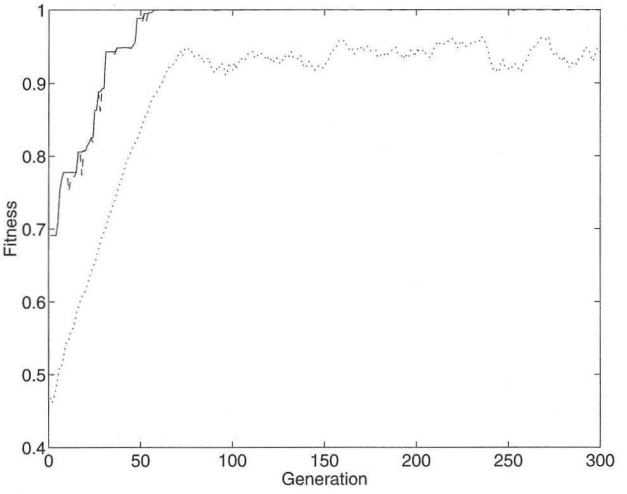

Figure 15: Fitness versus generation for the adaptive population sizing GA applied to the modified partially deceptive function. Best individual found up to the current generation is shown with a solid line, maximum individual in the current population is shown with a dashed line, and the current population average is shown with a dotted line.

range, and found the true optima. Note that the population size first increases, lowering variances and spreading a population over the search space. Then, as the population size decreases, and selection is applied to the individuals deleted, recombination assembles the correct solution. This can be seen in the jump from the false optima to the true optima when overall population size decreases. Near the end of the run, the population size remains steady at a value lower than its maximum. This happens because selection has lowered the diversity of the population, thus lowering the variance of population members and the adequate population size needed to match the target loss.

6. Final comments

The theoretical developments, simulations, and experiments presented in this paper indicate that the suggested technique for adaptive population size adjustment is viable. Further experimentation will be necessary to fully confirm the effectiveness of the algorithm. The fundamental framework of the algorithm is firmly founded. That is, schemata variances affect the accuracy of selection, and population sizes effect variances. Therefore, population size adjustment should be based on variances. The suggested procedure uses

Table 6: Parameters used in the representative run of the adaptive population sizing GA.

Initial Population Size	100
Maximum Population Size	5000
Crossover Rate	0.6
Mutation Rate	0.001
Target Loss L_t	0.0001
Sigmoid Slope β	10
Maximum Growth Rate γ	1.0

mating pairs to estimate schemata fitness variances, in much the same way that the traditional GA uses individuals to estimate schemata average fitness values. This keeps with the notion of implicit parallelism in the GA, and allows for explicit parallelism as well.

There are aspects of the suggested algorithm that could be altered while staying within the same conceptual framework. In particular, it may be useful to explore alternatives to the the sigmoidal mapping used to determine the growth rate $G_{1,2,3,4}$ and the division of $\hat{L}$ by $n(H'_{1,2}) + n(H'_{3,4})$ to control population over-expansion. In effect, these features are a control strategy over population growth. Other stochastic control strategies may also be effective. Exploration of a variety of such methods may be an important area for future research. However, the direct or indirect use of estimated variance and loss as a basis seems appropriate for any adaptive population sizing scheme.

Parts of the suggested algorithm have much in common with fitness sharing [15]. Fitness sharing spreads the population over high-fitness niches in the search space. The population resizing algorithm spreads the population across niches of variance. Like sharing, the population resizing algorithm uses a count of similar strings to control reproduction. A word must be said about the expense of counting the number of competitive schema in the suggested algorithm. Like the comparisons required in fitness sharing, this is an $O(N^2)$ operation. However, in considering this expense, one must compare it to the $O(N)$ operation of calculating string fitness values. In many GA applications, the expense of calculating fitness values far exceeds the cost of $O(N^2)$ masked, binary comparisons. Therefore, this expense may be negligible. Also, it may be possible to estimate the schema counts. As suggested in [16] samples can be used to evaluate approximate counts in fitness sharing. A similar technique may be useful in the population resizing scheme. It may also be possible to abstract a method of estimation from recent work on sharing-like behavior in GA immune system simulations [17]. Another method for estimating counts may be to periodically gather strings into *families* that share common schema. The resulting family sizes could be used as estimated counts. Since the results of a mating usually preserve

common schema between parents, adding children strings to a family would simply result in an increment of the estimated count. As mutations occurs, so would new families. Periodic creation of hybrid families could help to insure accurate count estimates.

Systems like the one suggested in this paper have significant potential. If GAs can control their search by automatically adjusting population sizes, they can effectively expand and contract to accommodate variations in problem complexity, and variations in available computational resources. A class of adaptive algorithms of this sort could extend the applicability of the GA, and extend its usability beyond the GA expert, to the novice user.

7. Acknowledgments

The author gratefully acknowledges support for this work provided by NASA under grant number NAG 9-625. The author also acknowledges support provided by the National Science Foundation under grant number ECS-9212066.

References

[1] D. E. Goldberg, "Optimal Initial Population Size for Binary–coded Genetic Algorithms," TCGA Report No. 85001, University of Alabama, The Clearinghouse for Genetic Algorithms, Tuscaloosa, 1985.

[2] D. E. Goldberg, "Sizing Populations for Serial and Parallel Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, pages 70–79, 1989.

[3] D. E. Goldberg, K. Deb, and J. H. Clark, "Accounting for Noise in the Sizing of Populations," in L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 127–140 (San Mateo, Morgan Kaufmann, 1992).

[4] D. E. Goldberg and M. Rudnick, "Genetic Algorithms and the Variance of Fitness," *Complex Systems*, **5** (1991) 265–278.

[5] M. Valenzuela-Rendon, "Two Analysis Tools to Describe the Operation of Classifier Systems," TCGA Report No. 89005, The University of Alabama, The Clearinghouse for Genetic Algorithms, Tuscaloosa, 1989.

[6] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic Algorithms, Noise, and the Sizing of Populations," IlliGAL Technical Report No. 91010, University of Illinois at Urbana-Champaign, 1991.

[7] J. H. Holland, *Adaptation in Natural and Artificial Systems* (MIT Press, Cambridge, 1992).

[8] D. E. Goldberg, "Genetic Algorithms and Walsh Functions: Part I: A Gentle Introduction," *Complex Systems*, **3** (1989) 129–152.

[9] D. E. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93 (Morgan Kaufmann, San Mateo, 1991).

[10] S. B. Thrun, "The Role of Exploration in Learning Control," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pages 527–560 (Van Nostrand Reinhold, New York, 1992).

[11] D. E. Goldberg, "Simple Genetic Algorithms and the Minimal, Deceptive Problem," in L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 74–88 (Morgan Kaufmann, Los Altos, 1987).

[12] R. E. Smith and D. E. Goldberg, "Diploidy and Dominance in Artificial Genetic Search," *Complex Systems*, **6** (1992) 251–285.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison–Wesley, Reading, 1989).

[14] S. Forrest and M. Mitchell, "Relative Building-block Fitness and the Building-block Hypothesis," in D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 109–126 (Morgan-Kaufmann, San Mateo, 1993).

[15] K. Deb, "Genetic Algorithms in Multimodal Function Optimization," TCGA Report No. 89002, The University of Alabama, The Clearinghouse for Genetic Algorithms, Tuscaloosa, 1989.

[16] C. K. Oei, D. E. Goldberg, and S. Chang, "Tournament Selection, Niching, and the Preservation of Diversity," IlliGal Tech. Rep. 91001, University of Illinois, Urbana, 1991.

[17] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for Diverse, Cooperative Populations with Genetic Algorithms," *Evolutionary Computation*, **1** (1993) 127–149.