Dynamic Control of Genetic Algorithms in a Noisy Environment

Akiko N. Aizawa
National Center for
Science Information Systems
3-29-1 Otsuka, Bunkyo-ku
Tokyo 112, Japan
akiko@nacsis.ac.jp

Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main Street
Urbana, IL 61801, U.S.A.
wah@manip.crhc.uiuc.edu

Abstract

In this paper, we present efficient algorithms for adjusting configuration parameters of genetic algorithms that operate in a noisy environment. Assuming that the population size is given, we address two problems specifically important in a noisy environment. First, we study the *duration-sizing problem* that determines dynamically the duration of each generation. Next, we study the *sample-allocation (sizing) problem* that determines adaptively the number of evaluations taken from each population in a generation. For these two problems, we model the search process as a statistical selection process and derive equations useful for controlling the duration and the sample sizes. Our result shows that these adaptive procedures improve the performance of genetic algorithms over those of commonly used static ones.

1 INTRODUCTION

Genetic algorithms provide robust yet efficient procedures for guiding searches even in the absence of domain knowledge. In these knowledge-lean domains, it is difficult to express an application in a well-defined model and analyze its behavior. A feasible way for finding a solution is to measure the performance of candidate solutions through actual experimentation or simulation. This is particularly difficult when the application environment is *noisy*.

In a noisy environment, at least a few evaluations are needed to estimate the performance of a candidate solution; better accuracy on the solution value can be obtained when more tests are performed. On the other hand, because the search is probabilistic, the more candidates we examine, the greater the probability of encountering better ones. Therefore, there is a trade-off between the accuracy of estimation and the number of candidate solutions examined in a search. This trade-off, although important, is difficult in knowledge-lean application domains, where domain knowledge relating quality and cost is missing (Wah 1992).

In this paper, we focus on the problems of selecting appropriate parameter values for genetic algorithms operating in a noisy environment. The parameters specifically important in a noisy environment include:

- M: population size in each generation;
- T: duration of each generation; and
- N: sample size of each candidate.

In the rest of this paper, we use the term *candidate* for a candidate solution and *sample* for the observed performance of a candidate in each simulation or experimental run (sometimes called an episode). In common genetic algorithm terminology, these are often referred to as *individual* and *fitness evaluation*, respectively. [1] In our terminology, M is the number of candidates maintained in one generation; T is the total number of samples assigned to each generation; and N is the number of samples taken from each candidate. The problems we have addressed are on the appropriate adjustment of these parameters for individual applications. We call these problems *parameter sizing problems* in genetic algorithms.

Assuming that unit time is needed to evaluate a candidate once, the relationship among these three parameters is:

$$T = M \times N \ . \tag{1}$$

In practice, it is implicitly assumed that these parameters are constant and do not change between/within generations. The most common approach to adjust these parameters is to assume that duration T is given, and that the goal is to determine the best population size M based on schema analysis (Fitzpatrick & Grefenstette 1988, Goldberg, Deb & Clark 1991).

[1] We use these terms mainly because we approach our problem from statistical consideration of general heuristic search problems.

In this paper, we assume that (i) M is given, and (ii) T and N can vary between/within generations. Let t_k be the total sample size assigned to the k-th generation, and $n_{i,k}$ be the sample size of the i-th candidate in the k-th generation. The relationship corresponding to Eq.(1) is:

$$t_k = \sum_{i=0}^{M} n_{i,k} \ , \qquad (k = 1, 2, \cdots) \ . \tag{2}$$

We study two important problems related to the adjustment of these parameters. The first problem is the *duration-sizing problem* that focuses on methods for varying T and N in between generations. (These parameters are kept constant in a generation; that is, $n_{1,k} = n_{2,k} = \cdots = n_{M,k}$.) The second problem is the *sample-allocation problem* that focuses on the assignment of N within a generation; that is, N may be different for different candidates in one generation while T is kept constant ($t_1 = t_2 = \cdots$.)

The rest of the paper is organized as follows. Section 2 presents the conventional *static policy* with constant T and N and shows the advantage of using *dynamic policies* where T and N are dynamic. Section 3 presents a statistical model of the generate-and-test process used in genetic algorithms. We also show methods for collecting statistics when a genetic algorithm is executed. In Section 4, we study the duration sizing problem and derive in Section 5 decision equations for sample allocation. Section 6 shows the improvement in performance of our dynamic policies over that of the conventional static ones. Finally, conclusions are drawn in Section 7.

2 PROBLEM DESCRIPTION

2.1 ISSUES IN CONVENTIONAL PARAMETER SIZING PROBLEMS

Fitzpatrick and Grefenstette have studied the best selection of M and N assuming that T is given (Fitzpatrick & Grefenstette 1988). Their analysis is based on the schema (hyperplane) theory. Let r be the number of candidates with hyperplane H, where $r \propto M$. Further, let ${\sigma_S}^2$ be the variance of sample means of r candidates, σ_H be the variance of the true performance of r candidates, and $\sigma(c_i)^2$ be the variance of samples from candidate c_i. In their paper, they show the following equation based on statistical analysis:

$${\sigma_S}^2 \ = \ \frac{1}{r}\,{\sigma_H}^2 \ + \ \frac{1}{rN} < \sigma(c_i)^2 >_H \ , \tag{3}$$

where $< \sigma(c_i)^2 >_H$ is the average sample variance of r candidates with hyperplane H. Since duration time T is given, rN in Eq. (3) is constant according to Eq. (1). Eq. (3), therefore, shows that when T is given (and the cost of generating new candidates is negligible), a genetic algorithm will perform better when sample size N is 1. [2]

There have also been other studies on genetic algorithms operating in noisy environments (Grefenstette, Ramsey & Schultz 1990, Goldberg & Rudnick 1991, Goldberg, Deb & Clark 1991). Most analytical studies deal with static policies where T and N are chosen beforehand and remain unchanged throughout the process.

2.2 PROBLEMS WITH STATIC POLICIES

In this paper, we study the case in which the population size M is given. This is often true in practice where M is determined by the available computational resources, the requirement on convergence, and the characteristics of the search space. When only M is given, Eq.(3) is not useful as it only indicates that taking more samples within one generation is better.

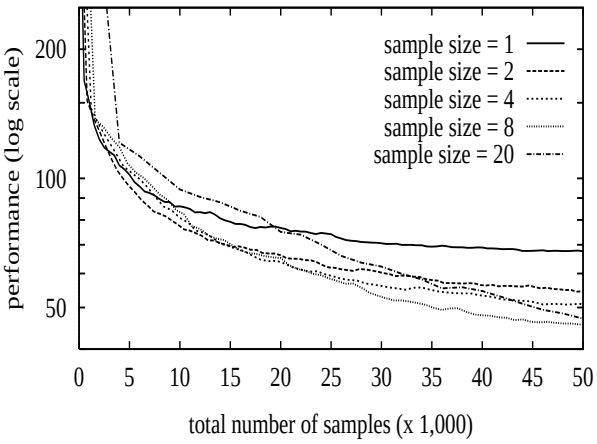


Figure 1: Behavior of static policies.

Fig.1 shows the effect of changing N (and accordingly T) for $M = 100$ in a minimization problem. The conditions for our experiments are the same as those in Fitzpatrick and Grefenstette's study (Fitzpatrick & Grefenstette 1988), except that we use larger sample variance $\sigma = 17$ instead of $\sigma = 2$ in their study. Fig.1 shows (i) that there is no optimal static policy that performs the best consistently even for the same problem, and (ii) that the best value of N depends on the total execution time. In general, smaller sample sizes are better in early generations, and larger ones are better in later generations. For example, Fig.1 shows that $N = 1$ performs the best in the beginning, but performs the worst at the end.

[2] In order to make our analysis consistent, we have modified the symbols used in Fitzpatrick and Grefenstette's analysis. The original equation is ${\sigma_S}^2 = \frac{1}{r}\sigma^2 + \frac{1}{rn} < \sigma^2(x_i) >_H$

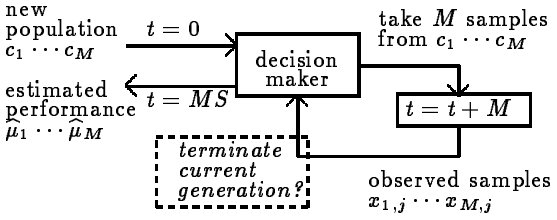
2.3 DEFINITION OF THE SIZING PROBLEMS

Based on the above observation, we focus in this paper on methods for determining N and/or T dynamically during the execution of a genetic algorithm. As is described in Section 1, we classify these problems into two types as is shown in Fig.2.

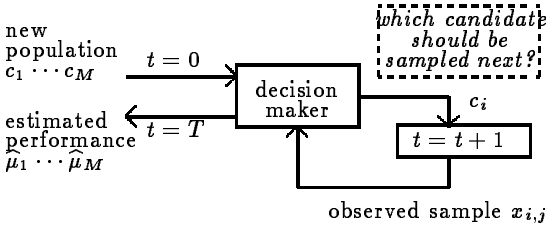
- *Duration-sizing problem.* This entails methods for determining when to stop the current generation. Candidates in the same generation are assumed to be sampled equally.

- *Sample-allocation (sizing) problem.* This involves methods for allocating t_0 samples among M candidates, where t_0 is given.

Duration-sizing is applicable when there are plenty of computation resources. It mainly concerns the convergence of genetic algorithms when time is large. In contrast, sample-allocation is applicable when each evaluation (or sample drawn) is costly. Consequently, it is essential to select tests carefully in order to have the maximum benefit on the sample-test results, and to keep the test results in order to adjust the sample size dynamically. Note that these two problems do not occur concurrently; hence, we do not attempt to evaluate the effects of both of them in the same genetic algorithm.



(a) duration-sizing problem



(b) sample-allocation problem

Figure 2: Two types of sizing problems discussed in this paper.

3 STATISTICAL MODEL AND ASSUMPTIONS

3.1 MODEL OF GENERATE-AND-TEST

A statistical model for the generate-and-test process in genetic algorithms is shown in Fig.3. It consists of two parts.

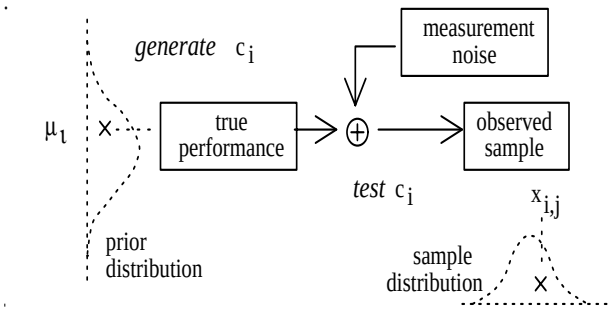


Figure 3: Statistical model.

- *Test process.* We assume that the evaluation noise is Gaussian as is in other studies. The variance of the noise is denoted as σ^2. Also, we assume that the noise distribution is common for all candidates and is invariant in time. Let $\mu_{i(k)}$ be the 'true' performance of candidate c_i in the k-th generation. Denote the j-th sample from c_i as $x_{i,j(k)}$, where $x_{i,j(k)}$ is interpreted as a sample from *sample distribution* $N(\mu_{i(k)}, \sigma^2)$. In the rest of this paper, we omit the suffix (k) when it is obvious.

- *Generation process.* Candidates in the same generation are created by applying randomized genetic operators and should, therefore, have identical statistical properties with respect to their 'true' performance μ_i. The *prior distribution* is the distribution of the 'true' performance of candidates in the k-th generation (namely, the distribution of $\mu_1, \cdots, \mu_M$). It expresses the distribution of expected performance of an arbitrarily chosen candidate in generation k. We assume that the prior distribution is a normal distribution $N(\mu_{0(k)}, {\sigma_{0(k)}}^2)$.

In our model, we assume normal distributions for both the sample and prior distributions. Such assumptions are widely accepted, and transformation methods are available when the distribution is not normal.

Using Bayes Theorem, we can calculate the *posterior distribution* of μ_i, a conditional distribution of μ_i given observed samples $x_{i1}, \cdots, x_{in_i}$, as follows. Assume that n_i samples are drawn from candidate c_i, and that the sample mean is $\overline{x}_i$. Let $h(\mu)$ be the prior distribution and $h_i^*(\mu|\overline{x}_i, n_i)$ be the posterior distribution of c_i. Knowing that the distribution of the sample mean $\overline{x}_i$

is $f(x|\mu_i, n_i) \sim N(\mu_i, \frac{\sigma^2}{n_i})$, the posterior distribution is, therefore,

$$h_i^*(\mu \,|\, \overline{x}_i, n_i) \;\sim\; N\left(\frac{n_i\,\overline{x}_i + \alpha\,\mu_0}{n_i \;+\; \alpha},\; \frac{\sigma^2}{n_i + \alpha}\right), \quad (4)$$

where $\alpha = \frac{\sigma^2}{{\sigma_0}^2}$. We denote the mean and variance of the above normal distribution as μ_i^* and ${\sigma_i^*}^2$, respectively. That is,

$$\mu_i^* \;=\; \frac{n_i\,\overline{x}_i \;+\; \alpha\,\mu_0}{n_i \;+\; \alpha}, \quad {\sigma_i^*}^2 \;=\; \frac{\sigma^2}{n_i + \alpha}\;. \qquad (5)$$

μ_i^* gives the best (Bayes) estimation with respect to μ_i, and ${\sigma_i^*}^2$ gives the expected estimation error (assuming squared-error loss).

3.2 ESTIMATION OF STATISTICAL PARAMETERS

Our model in **3.1** uses three parameters that can only be estimated statistically:

- σ^2 : the variance of the noise;
- $\mu_{0(k)}$: the mean of the μ_i's in the k-th generation;
- ${\sigma_{0(k)}}^2$: the variance of the μ_i's in the k-th generation.

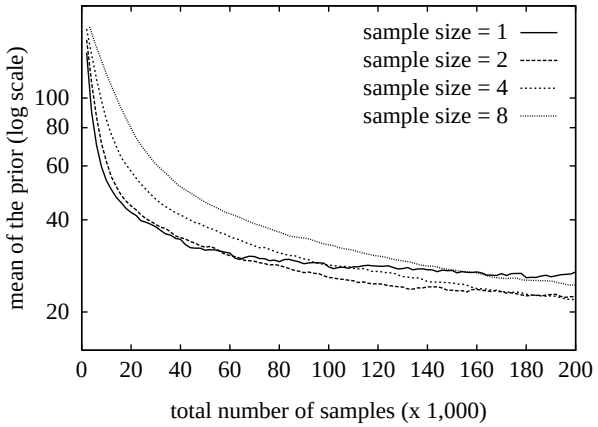
First, σ^2 is estimated using the common (or pooled) variance (Davis, Crow & Maxfield 1960) as follows:

$$\widehat{\sigma}^2 \;=\; \frac{\displaystyle\sum_{i=1}^{M}\left((n_i - 1){s_i}^2\right)}{\left(\displaystyle\sum_{i=1}^{M} n_i\right) \;-\; M},$$

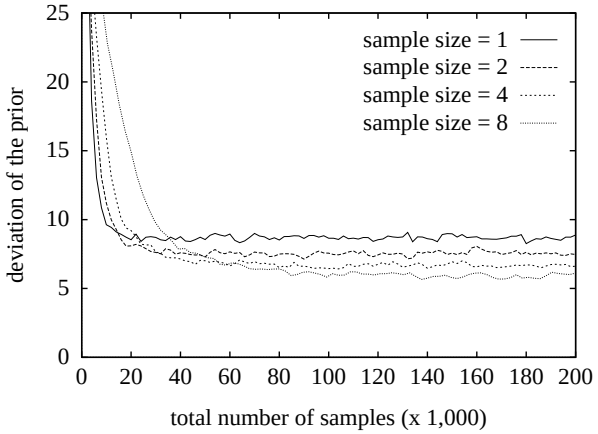
$$where\ (n_i - 1){s_i}^2 = \sum_{j=1}^{n_i}(x_{i,j} - \overline{x}_i)^2. \quad (6)$$

The estimation of μ_0 and ${\sigma_0}^2$ is the same as is in the case of two-factor analysis (Davis, Crow & Maxfield 1960). It is known that, when $n_1 = \cdots = n_M = n$, the distribution of the sample means $\overline{x}_i$'s is normal with mean μ_0 and variance $\left({\sigma_0}^2 + \frac{\sigma^2}{n}\right)$. Therefore, $\mu_{0(k)}$ and ${\sigma_{0(k)}}^2$ are estimated as:

$$\widehat{\mu}_{0(k)} \;=\; \frac{\displaystyle\sum_{i=1}^{M}\overline{x}_i}{M}\;,$$



(a) Transition of $\mu_{0(k)}$



(b) Transition of $\sigma_{0(k)}$

Figure 4: Transition of $\mu_{0(k)}$ and $\sigma_{0(k)}$ between generations. Averaged over 100 simulation runs.

$$\widehat{\sigma}_{0(k)}^2 \;=\; \frac{M\displaystyle\sum_{i=0}^{M}{\overline{x}_i}^2 - \left(\displaystyle\sum_{i=0}^{M}\overline{x}_i\right)^2}{M(M-1)} \;-\; \frac{\widehat{\sigma}^2}{N}\;. \quad (7)$$

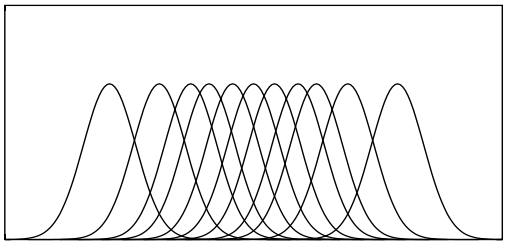
The initial values of these three parameters are obtained through pre-sampling, where a negligible number of samples are taken (say 4 samples from each candidate in the first generation). These estimates are then updated in each generation using Eq.(7). As existing samples are used for these estimations, no additional pre-sampling is needed after the first generation.

The values of $\mu_{0(k)}$ and ${\sigma_{0(k)}}^2$ will change as the genetic algorithm is executed, as is illustrated in Fig.4. In general, $\mu_{0(k)}$ gradually improves as $\sigma_{0(k)}$ becomes smaller.

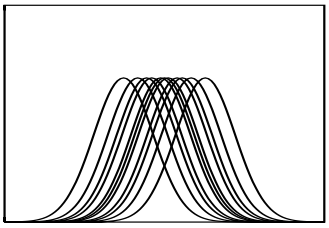
4 DURATION-SIZING PROBLEM

4.1 OBJECTIVE

In executing a genetic algorithm, useful schemata are identified implicitly by selecting candidates with better sample means. Statistically, the difficulty of selection in the k-th generation can be characterized by the *variance ratio* (denoted as $\beta(k)$) defined as the ratio of the sample variance and the variance of μ_i's; that is, $\beta(k) = \frac{\sigma^2}{{\sigma_{0(k)}}^2} (= \frac{1}{\alpha})$. When $\beta(k)$ is small, the μ_i's are distributed 'sparsely,' and the selection of better candidates is easy. On the other hand, when $\beta(k)$ is large, then the μ_i's are distributed close to each other, and the selection is difficult. Fig.5 illustrates the cases with small and large $\beta(k)$.



(a) Small $\beta(k)$



(a) Large $\beta(k)$

Figure 5: Selection problem with different variance ratios.

The above discussion explains why static policies do not work well. Note that in Fig.4-(b) the distribution of the μ_i's is 'sparse' at first and gradually becomes 'dense as time progresses. Since the variance of the μ_i's varies from one generation to another, no sample size N can be the best throughout all the generations. Hence, we can expect the overall performance to improve if we use smaller sample sizes in early generations and larger ones in later generations.

4.2 EQUATIONS

The above observation leads us to use the following strategy for determining the duration time.

$$t_k \;=\; M \;\times\; \left(n_0 \;+\; \lceil\, \gamma \sum_{l=1}^{k-1} t_l \,\rceil \right) . \qquad (8)$$

In this case, $\sum_{l=1}^{k-1} t_l$ is the total number of samples observed so far. Eq.(8) simply means that we increase the duration time at fixed intervals of time. The initial sample size n_0 is determined so that the following condition is satisfied:

$$\frac{\sigma^2/n_0}{\sigma^2_{0(k)}} \;=\; \frac{\beta(0)}{n_0} \;\leq\; \delta\ . \qquad (9)$$

Presently, γ is heuristically chosen as 5×10^{-3}, and δ as 3.0. Under practical conditions, $n_0 = 1$ for most of the cases.

4.3 DURATION-SIZING PROCEDURE

To summarize, our adaptive procedure for determining the duration of a generation consists of four steps.

(1) Determine the initial sample size n_0 using Eq.(9).

(2) Take one sample from each of the candidates in the current generation.

(3) If the current time t is less than t_k as is defined in Eq.(8), then go to Step (2)

(4) Else terminate the current generation; generate new candidates for the next generation; and go to Step (1)

5 SAMPLE-ALLOCATION PROBLEM

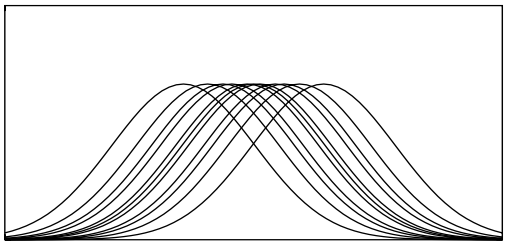
5.1 OBJECTIVE

In this problem, we consider only one generation. Our approach is based on decision theoretic methods. The objective is to determine an efficient allocation $(n_1, \cdots, n_M)$ so that the expected value of a predefined loss function (or *risk*) is minimized. Here, $\sum_{i=1}^{M} n_i = T$, where T is the total number of samples assigned to each generation. As a loss function, we use the estimation error $({\sigma_i^*}^2)$ of the candidate selected.

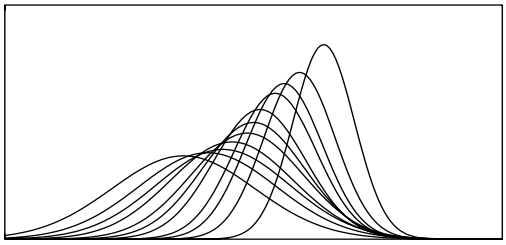
Our choice is motivated by the fact that candidates with better performance have larger probability of being selected for reproduction in the next generation and should, therefore, have greater influence. Hence, we can improve the performance of genetic algorithms by sampling more from better candidates (*i.e.*, by decreasing the estimation error of these candidates) and by spending less time on inferior ones. (See Fig.6.)

5.2 EQUATIONS

Our approach tries to minimize the expected risk by determining adaptively the candidate to be sampled next. The expected risk is expressed as follows:



(a) Equal n_i



(b) Unequal n_i

Figure 6: Equal and unequal allocation of n_i's.

$$\overline{R} \;=\; \sum_{i=1}^{M} P_i^* \; {\sigma_i^*}^2 \;, \qquad (10)$$

where P_i^* is the probability that candidate c_i is the best. We calculate P_i^* as:

$$P_i^* = \int_{-\infty}^{\infty} \prod_{j \neq i} H_j^*(\mu|\overline{x}_j, n_j)\, h_i^*(\mu|\overline{x}_i, n_i)\, d\mu \;, \qquad (11)$$

where $H_j^*(\mu|\overline{x}_j, n_j) = \int_{-\infty}^{\mu} h_j^*(\mu|\overline{x}_j, n_j) d\mu$ is the cumulative density function of h_i^*. Note that P_i^* could have been defined as the actual selection probability $\frac{f_i}{\sum_{i=1}^{M} f_i}$, where f_i is the fitness of candidate i; however, we use Eq.(11) because it is less dependent on the specific fitness function (especially the scaling factor) used in individual genetic algorithms. [3] For ease of calculation, we make the simplifying assumption that P_i^* is independent of ${\sigma_i^*}^2$; therefore, $\frac{\partial P_i^*}{\partial n_i} = 0$ for all i.

Now let $(n_1^+, \cdots, n_M^+)$ be the optimal (desired) allocation for Eq.(11). By applying Lagrange multiplier λ, we obtain the following M equations for each $i = 1 \cdots M$:

$$\frac{\partial}{\partial n_i} \left[\sum_{i=1}^{M} P_i^* {\sigma_i^*}^2 \;-\; \lambda \sum_{i=1}^{M} n_i \right] \;=\; 0 \;. \qquad (12)$$

Since $\frac{\partial P_i^*}{\partial n_i} = 0$, it immediately follows that:

$$P_i^* \; \frac{\partial {\sigma_i^*}^2}{\partial n_i} \;-\; \lambda \;=\; 0 \;, \qquad i = 1 \cdots M \;. \qquad (13)$$

Therefore, using σ_i^* as is defined in Eq.(5), $(n_1^+, \cdots, n_M^+)$ should be chosen in such a way that

$$-P_1^* \frac{\sigma^2}{\left(n_1^+ + \alpha\right)^2} \;=\; \cdots \;=\; -P_M^* \frac{\sigma^2}{\left(n_M^+ + \alpha\right)^2}. \qquad (14)$$

Eq.(14) means that when the samples are optimally allocated, each term of the equation should be equal. Comparing the term $-P_i^* \frac{\sigma^2}{(n_i+\alpha)^2}$ for the current and the desired allocations, we obtain the feedback value for each i. Note that we can only increase n_i for samples drawn in the future; hence, the best sampling strategy is to select candidate i with the largest feedback value:

$$\max_{1 \le j \le M} \left[P_j^* \frac{\sigma^2}{\left(n_j + \alpha\right)^2} \;-\; P_j^* \frac{\sigma^2}{\left(n_j^+ + \alpha\right)^2} \right] \;. \qquad (15)$$

Because the second term of Eq.(15) is common for all candidates, we can only use the first term of the equation in comparing the current and the desired allocations.

5.3 SAMPLE-ALLOCATION PROCEDURE

To summarize, our adaptive procedure for determining sample allocation is as follows.

(1) Sample once each of the M candidates.
(2) Calculate σ_i^* and P_i^* for each candidate.
(3) Select the candidate with the largest feedback value using Eq.(15).
(4) Sample the candidate selected in Step (3).
(5) Repeat Steps (2) through (4) until T samples have been drawn.

6 EXPERIMENTAL RESULTS

6.1 EXPERIMENTAL CONDITIONS

To illustrate the effectiveness of our dynamic parameter-sizing procedures, we compare the performance of static policies with that of dynamic ones. Assuming that candidate c_i is represented by a binary code z_i $(= z_{i,1} z_{i,2} \cdots)$, the 'true' evaluation value μ_i is determined by z_i using a test function f, where $\mu_i = f(z_i)$. In this example, we use two test functions. The first function F_1 is DeJong's uni-modal function (DeJong 1975):

$$f_1(z_i) \;=\; \sum_{m=1}^{30} \; i \; {z_{i,m}}^4 \;, \qquad (16)$$

[3] In our experiments described in Section **6**, instead of defining P_i^* as is in Eq.(11), we use an approximate P_i^*, where only the probability that candidate c_i is better than the current best candidate (or second best, in case that c_i is the best) is considered.

where $z_{i,m}$ has 8 bits, and $-1.28 < z_{i,m} \leq 1.28$. The second function F_2 (Mühlenbein, Schomisch & J.Born 1991) is a highly multi-modal function.

$$f_2(z_i) \;=\; 20 + \sum_{m=1}^{20} \left({z_{i,m}}^2 - cos(2\pi z_{i,m}) \right) \;, \quad (17)$$

where $z_{i,m}$ has 10 bits, and $-5.12 < z_{i,m} \leq 5.12$.

The size of the solution space is 2^{240} for F_1 and 2^{200} for F_2. Both function have the global minima when all $z_{i,m} = 0$. Note that F_2 has 7^{20} local minima. When the z_i's are created randomly, the distribution of the μ_i's is almost normal: (i) for F_1, $\mu_{0(0)} = -225$ and $\sigma_{0(0)} = 69$; and (ii) for F_2, $\mu_{0(0)} = -194$ and $\sigma_{0(0)} = 35$. The evaluation noise is set to be $\sigma = 1.0 \times \sigma_{0(0)}$. We use standard genetic algorithm parameters with population size of 100, crossover rate of 0.6, mutation rate of 0.001, and scaling-window size of 7.

6.2 PERFORMANCE COMPARISON: DURATION-SIZING PROBLEMS

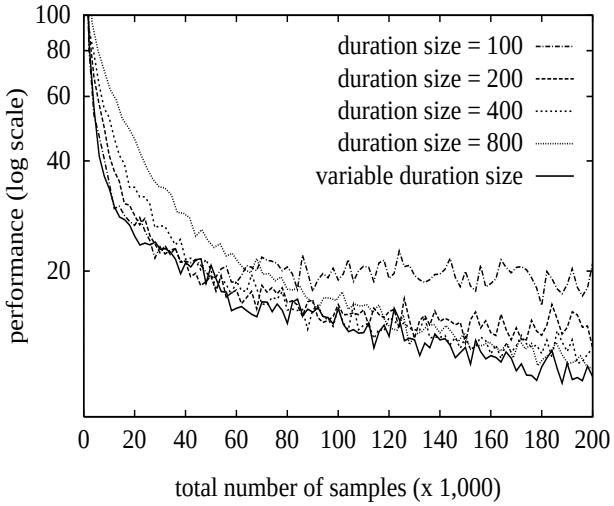
Fig.7 shows the effect of using static versus dynamic duration sizes. The x-axis is the total test time, and the y-axis is the performance of the current top candidate ('on-line' performance). The generation gap is set to be 1.0. Since we assume relatively small sampling cost, no samples drawn in previous generations are carried over to future ones. The result shown in Fig.7 is the average over 50 runs.

For both F_1 and F_2, the performance of our policy using dynamic duration sizes is at least as good as that of the best static policy at any point in time.

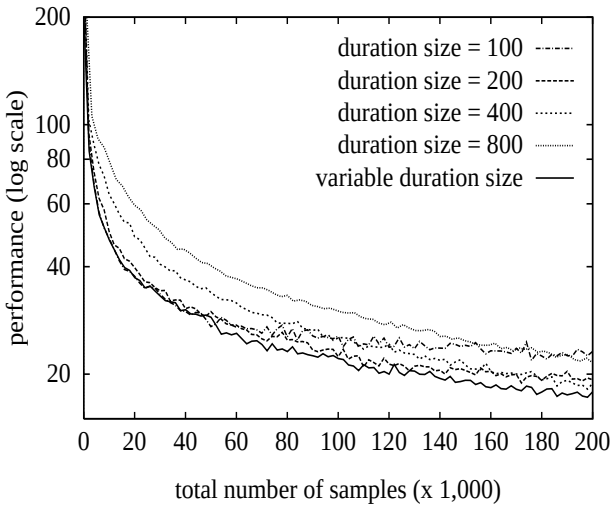
6.3 PERFORMANCE COMPARISON: SAMPLE-ALLOCATION PROBLEMS

Fig.8 shows the performance results of policies based on static and dynamic sample allocation. Again, the x-axis is the total test time, and the y-axis is the performance of the current top candidate. The generation gap is set to be 0.6 in this case, and the best 40% candidates survive into the next generation. As we assume that the sampling cost is high, samples drawn in previous generations are carried over to future ones. The results plotted are the average over 200 runs. For static allocation, we use a sample size of 2, which is the best within the time range.

For both F_1 and F_2, our dynamic policy consistently overperforms the static ones. Though only on-line performance is shown here, similar improvements have been found for off-line performance.

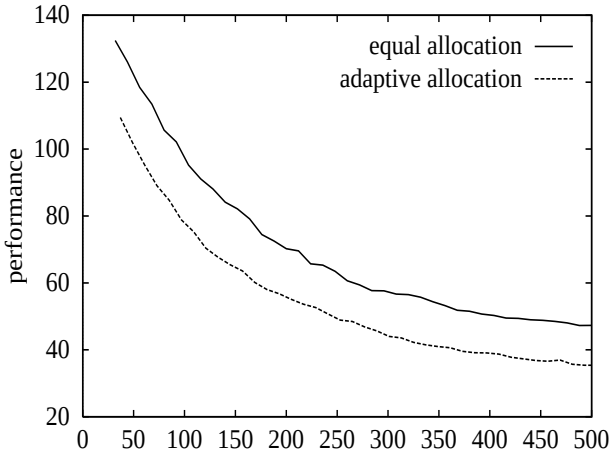


(a) Test function F_1

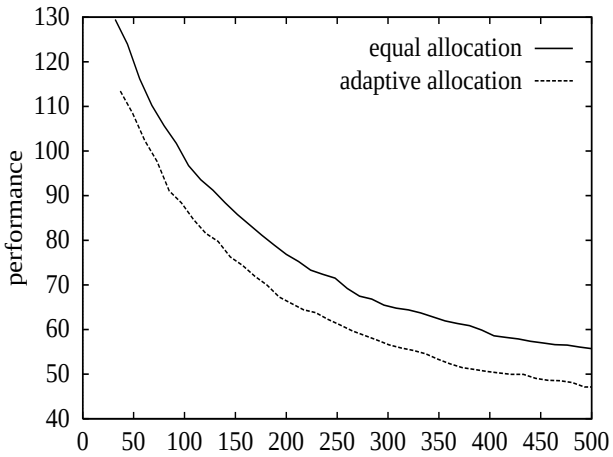


(b) Test function F_2

Figure 7: Effect of variation in duration sizes.



(a) Test function F_1



(b) Test function F_2

Figure 8: Effect of sample allocation.

7 CONCLUSIONS

In this paper, we have studied two related parameter-sizing problems for genetic algorithms operating in a noisy environment: a) determining a suitable duration of a generation, and b) finding an appropriate number of samples to be drawn from each population. We assume a constant number of populations in each generation. Our results show that dynamic policies perform better than existing static ones.

Although we have experimented with simple test functions, the advantage of using dynamic policies is universal and is independent of the reproduction mechanism of individual genetic algorithms. (In the extreme case, dynamic policies can improve the performance of random searches.) We are in the process of incorporating the policies presented here in a heuristics learning system (Wah 1992). Future studies will report results on evaluating these policies for realistic applications.

Acknowledgements

Research supported by National Aeronautics and Space Administration Grant NAG 1-613 and Joint Services Electronics Program Contract N00014-90-J-1270.

References

Wah, B. W. (1992). Population-Based Learning: A New Method for Learning from Examples under Resource Constraints. *IEEE Trans.on Knowledge and Data Engineering*, **4**(5):454-474.

Fitzpatrick, J. M. and J. J. Grefenstette (1988). Genetic Algorithms in Noisy Environments. *Machine Learning*, **3**(2/3):101-120.

Goldberg, D. E., K. Deb and J. H. Clark (1991). Genetic Algorithms, Noise, and the Sizing of Populations. *IlliGAL* Technical Report, 91010, Univ.of Illinois at Urbana-Champaign, Urbana, IL.

Grefenstette, J. J., C. L. Ramsey and A. C. Schultz (1990). Learning Sequential Decision Rules using Simulation Models and Competition. *Machine Learning*, **5**:355-381.

Goldberg, D. E. and M. Rudnick (1991). Genetic Algorithms and the Variance of Fitness. *Complex Systems*, **5**:265-278.

Crow, E. L., F. A. Davis and M. W. Maxfield (1960). *Statistics Manual.* Dover Publications.

DeJong, K. A. (1975). *Analysis of the behavior of a class of genetic adaptive systems.* Ph.D. Thesis, Univ.of Michigan, Ann Arbor, MI.

Mühlenbein, H., M. Schomisch and J. Born (1991). The Parallel Genetic Algorithm as Function Optimizer. In *Proc. of 4'th Int. Conf. on Genetic Algorithms*: 271-278. Morgan Kaufmann, San Mateo, CA.