

---

# Fixed Budget Allocation Strategies for Noisy Fitness Landscapes

---

**Adrian Grajdeanu**  
Computer Science Dept.  
George Mason University  
Fairfax, VA 22030

**Kenneth De Jong**  
Computer Science Dept.  
George Mason University  
Fairfax, VA 22030

## Abstract

A well-known problem in dealing with noisy fitness landscapes is the difficulty in deciding *a priori* how many samples per individual to take in order to get a useful estimate of an individual's fitness. This is particularly important when an EA is given *a priori* a fixed total budget of samples and must trade off increased accuracy of fitness estimates over fewer generations vs. less accurate fitness estimates over more generations. This paper investigates a technique for dynamically adapting the fitness estimation accuracy as evolution proceeds and compares the results of such adaptive runs with others in which the accuracy is chosen *a priori* and kept constant.

## 1 INTRODUCTION

The study of this paper concerns itself with improving the performance of evolutionary algorithms running on problems where the fitness of an individual can only be estimated by sampling. This is not a new question (see, for example, [Fitzpatrick and Grefenstette, 1988], [Hartley, 2002], or [Goldberg et al., 1992]). In such studies the focus has been on determining *a priori* an optimal fixed sampling rate or an optimal population size. The aim of our research is to evaluate techniques for dynamically adapting the accuracy of this sampling, striking a balance between tolerance to noise and budgetary resources (equivalent to available running time).

In the remainder of this paper we will describe the test problem that we used to evaluate our ideas. Then we will formally introduce two methods of adapting the accuracy parameter. The comparative results will be

presented and finally results of sensitivity analysis are included.

## 2 COMPETITION FOR RESOURCES PROBLEM

A classic example of problems requiring multiple evaluations to estimate an individual's fitness are domains in which fitness evaluation consists of running a stochastic simulation. Since the simulation's initial conditions change from run to run, the same individual's fitness assessment can vary widely from run to run. Since running a simulation can be quite expensive, we have the classic tradeoff of running more simulations per individual for a fewer number of generations vs. running fewer simulations per individual for more generations.

The simulation problem we're going to use in this study was introduced by [Spears and Gordon, 2000]. Because of space limitations, in this paper only the main details of their study are summarized, emphasizing where our work differs from theirs.

### 2.1 THE SIMULATED GAME

The game involves two players on a toroidal board of squares. Each square corresponds to a resource, and the two players compete for them. If the board is  $N \times N$  then one player starts at square  $(0, 0)$  and the other player at  $(1, 1)$ . The remaining squares are initially unoccupied. In this paper we assume  $N = 10$ .

Each player can only perceive limited information, namely, the status of the north, south, east and west squares neighboring the current position of the player. The diagonal squares cannot be seen. The status of each square will be one of the following: unoccupied, occupied by that player or occupied by the opponent. Due to the toroidal nature of the board the players are

close together at the beginning of the game. However, because they cannot see along diagonals, they can't see one another initially.

Each time step the players alternate taking an action, which consists of moving to a neighboring square. A player can move to an unoccupied square (and occupy it) or back to a square that it has previously occupied, but not to a square occupied by the opponent. The player isn't allowed to stand still and make no move. Once a player occupies a square it will hold that resource until the end of the game. A game ends when all squares are occupied or when time runs out. At the end, the player that holds most resources is the winner.

Throughout this paper, one of the players will have a fixed stochastic strategy: if it detects around itself any unoccupied squares it uniformly randomly moves to one of them. If there are no unoccupied squares, it uniformly randomly backtracks to one of the neighboring squares that it has previously occupied.

Given the game and this stochastic player, we focus on developing effective strategies for the other player.

## 2.2 THE PLAYER

The player that is evolved is a finite state machine with up to 25 states (not all of them necessarily active). The first state is the initial state and the table of transitions is indexed by input and current state. The current input is a string of codes for the neighboring squares: 0 for empty, 1 for occupied by the player and 2 for occupied by the opponent. For example if the square at north is empty, while the east is occupied by opponent and south and west are occupied by the player, the input will be '0211'. The transition table encodes the new state and an action of moving the player to one of the 4 immediate neighbors. If as a result of following the encoded action the player would perform an illegal move (attempting to move to a square already occupied) then the player stays still and loses the turn. Because the goal of the game is to secure as many squares as possible, the mobility of the player should be essential. It is hoped that this strategy acts as enough penalty and alleles indicating illegal moves will be weeded out by the evolution alone.

To estimate the fitness of a player, a number of games are played against the standard opponent. The fraction of victories achieved is the estimation of the fitness. The more games are played, the more accurate such estimation will be.

## 3 ADAPTING THE ACCURACY OF FITNESS ESTIMATE

The EAs rely on qualitative differences in individual fitnesses to guide the evolution process. However, they seem to be quite robust to noise in the estimation of fitness. For example, "common wisdom" in evolutionary robots is that, to evaluate the fitness of a robot control program, only a low number (in the single digits) of simulations are usually required. However, what was noticed in this particular problem domain is that, as evolution progresses, a gradual increase in the accuracy of the estimate is beneficial. A run with low accuracy will progress and plateau fairly quickly, while a run with high accuracy will register more consistent progress, but at a very slow rate.

Faced with this problem, [Spears and Gordon, 2000] decided to use a two-stage evaluation process. First a low accuracy measure is taken. If it suggests that the individual has high potential, then a high accuracy estimate is then taken, replacing the one less accurate. The knobs for high and low accuracy had fixed values determined via initial exploratory experiments. The goal of this paper is to see if this can be done dynamically during an evolutionary run.

Two ways of adapting fitness accuracy are investigated: the first method uses an island model, and the second method uses a meta EA. The common yardstick by which all methods are compared is a fixed budget of 30,000,000 games. Each of the methods may allocate this budget in different ways.

### 3.1 THE ISLAND MODEL

In this model, there are actually 3 instances of the same EA running in parallel. They all have the same parameters except the accuracy of the fitness evaluations. Each of the population sizes is 100 individuals, crossover and mutation run at probability .7 and .4. The algorithm is generational, uses fitness proportional selection and finishes when the budget of games is exhausted. When comparing the islands to determine the best performing one, the estimated fitness of the best so far individual in each of the islands is inadequate because the lack of accuracy of the fitness function. Instead the average estimated fitness of all individuals in the generation is employed as a more robust alternative in the face of noisy evaluations.

The fitness accuracy parameters on the 3 islands are always set to 3 consecutive values. These values are mapped to actual number of games per evaluations according to a monotonous increasing function that

controls the speed and magnitude of adjustment. In the initial experiments this function is the exponential of base 2. The 3 tracks evolve independently - drawing from the same limited budget of games - and every N generations they all stop for synchronization. At such point the best performing island is identified. Its accuracy parameter becomes the median and the value of this parameter on the other two islands may be adjusted. For example, suppose that islands A, B and C arrive at the synchronization point with the following results for fitness accuracy and average estimated fitness of last generation: (5 0.21) (7 0.23) and (6 0.14). In this case, island B is the best performing so far due to its highest average estimated fitness. 7 becomes the median value for fitness accuracy and therefore islands A and C adjust their accuracy parameter value one to 6 and one to 8, randomly. While there is a lower limit for the fitness accuracy parameter at 0, there is no upper limit imposed. In practice however, when the values of the parameter gravitate toward the high end of the spectrum, they tend to exhaust the budget of games rather quickly. Therefore a practical high end limitation imposes itself.

### 3.2 THE META EA MODEL

The Meta EA model is easily described as a variation of the island model. When the 3 parallel tracks meet for synchronization, not only the value of the fitness accuracy is altered according to the best performing island. In addition to that the whole population of the best performing island is cloned and entirely replaces both the populations on the two other islands. Looking at this scenario from a meta scale one can see an entire population as an individual and observe a (1,3) Evolutionary Strategy type of algorithm. The aim of this ES algorithm is to co-evolve the value of the accuracy fitness estimation with an ever-changing landscape. The tweaking operator performs a spread on the value of the accuracy parameter followed by a full blown N generations evolution on the current set of individuals.

## 4 EXPERIMENTAL RESULTS

The initial results present the fitness estimate resulting from averaging 6 independent runs for each method and each set of parameters mentioned.

### 4.1 FIXED ACCURACY BASELINE

For our baseline studies we used a standard EA with a population size of 100 using a fixed number of games per fitness estimate. Using guidance from the litera-

ture we experimented with a variety of accuracy parameter values. Figure 1 and Figure 2 show the best results obtained, namely, with an evaluation with the accuracy set at 4 and 8 games per evaluation. The Y axis represents the fraction of games won, and the curves are depicting the average fitness estimate  $\pm$  one standard deviation.

Although there is considerable variance from run to run, 4 games per evaluation seemed about optimal. This is consistent with earlier work [Fitzpatrick and Grefenstette, 1988].

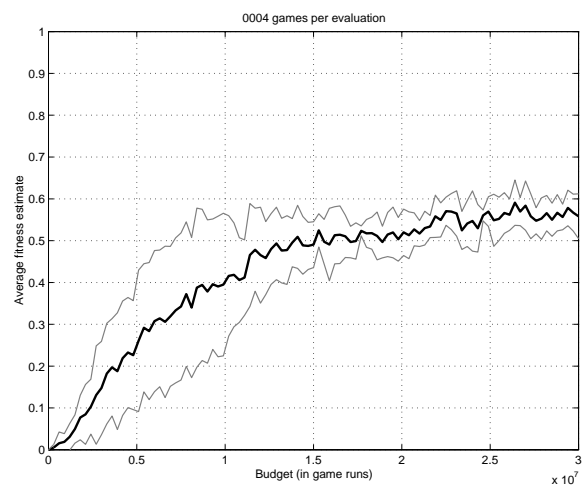


Figure 1: Fixed fitness estimation accuracy: 4

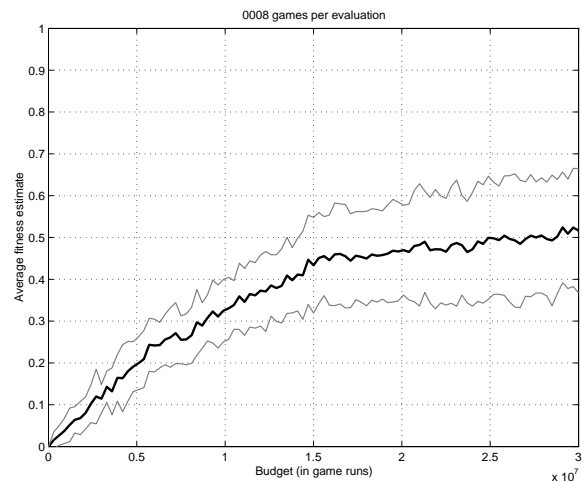


Figure 2: Fixed fitness estimation accuracy: 8

## 4.2 THE ISLAND MODEL

In our initial experiments we held the population size fixed at 100 in each of the 3 island EAs. The experiments were designed to test the effects of the frequency  $N$  of the synchronization points, ranging from every generation to every 8192 generations in exponential increments. Typical results show that for  $N$  below 64 at the end of the budget there was high variability between the independent runs. When the synchronization was performed every 64 to 1024 generations ( $64 \leq N \leq 1024$ ), the independent results came within .15 and .35 on the fitness scale. For  $N > 1024$  the results slowly inched up to between .25 and .45. Figure 3 and Figure 4 depict the above observations.

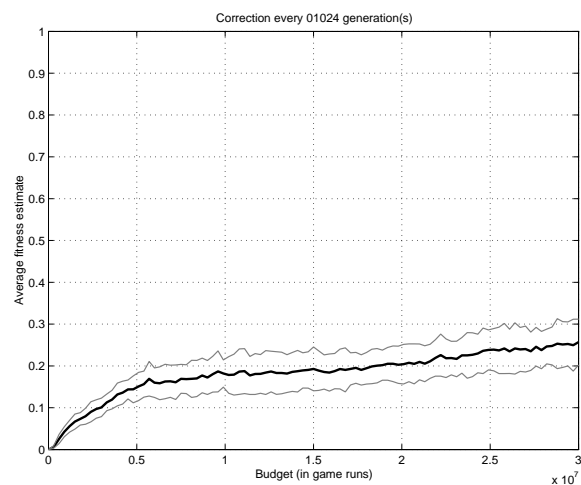


Figure 3: Island Model

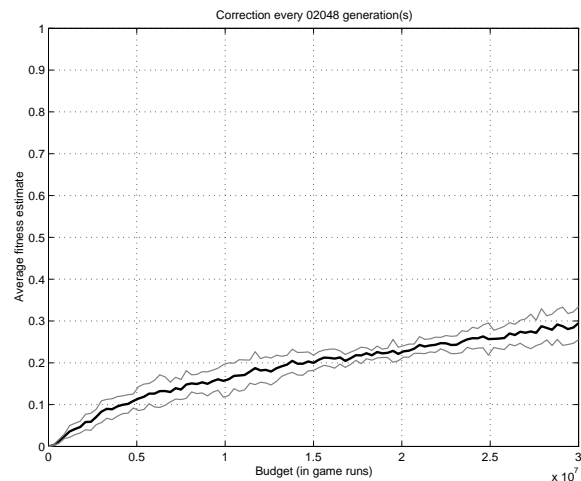


Figure 4: Island Model

As far as adapting the number of games per evaluation, Figure 5 shows that for  $N = 2048$  the accuracy parameter evolves to a value that maps to between 8 and 64 games per evaluation. In general for values of  $N < 64$  the accuracy parameter wanders towards either the high or the low ends and takes a significant time to come back around ideal values. This explains the variability observed when  $N < 64$ .

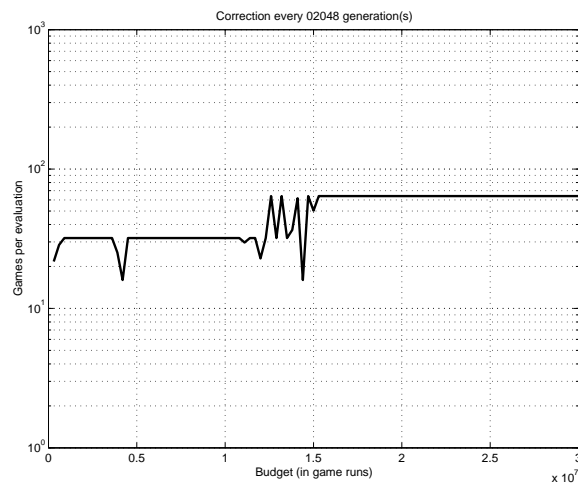


Figure 5: Island Model

## 4.3 THE META EA MODEL

Similar experiments were performed on an identically configured Meta EA model to study the effects of the frequency  $N$  of the synchronization points. Like in the previous model, for values of  $N < 64$  variability between independent runs was too large. However, for values above 64, as  $N$  increases, the results are increasingly better. Included are the results for  $N = 1024$  (Figure 6) and for  $N = 2048$  (Figure 7). They show fitness between .25 and .4 in the first case, and .3 to .45 in the second case. In general the fitness results of the Meta EA Model are superior to those of the Island Model. This was the expected result because the more opportunistic nature of the Meta EA Model.

Figure 8 shows a typical evolution of the accuracy parameter. It shows a steady and paced increase from between 8 and 32 to between 32 and 128.

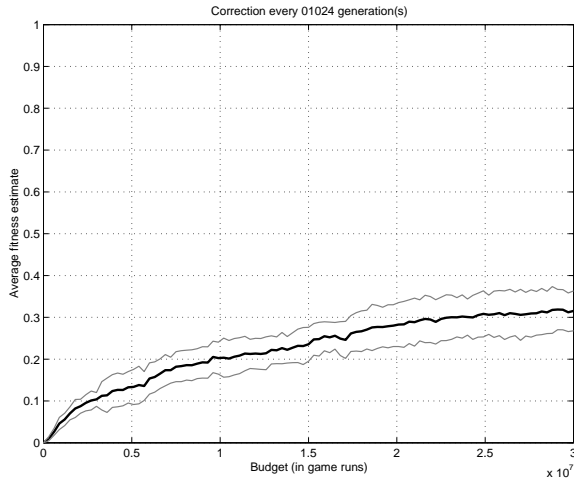


Figure 6: Meta EA Model

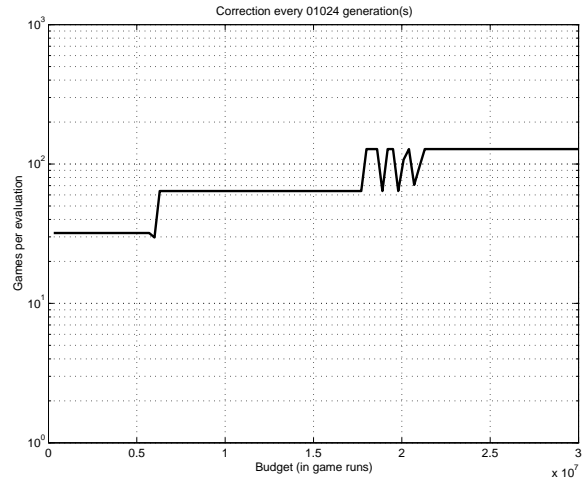


Figure 8: Meta EA Model

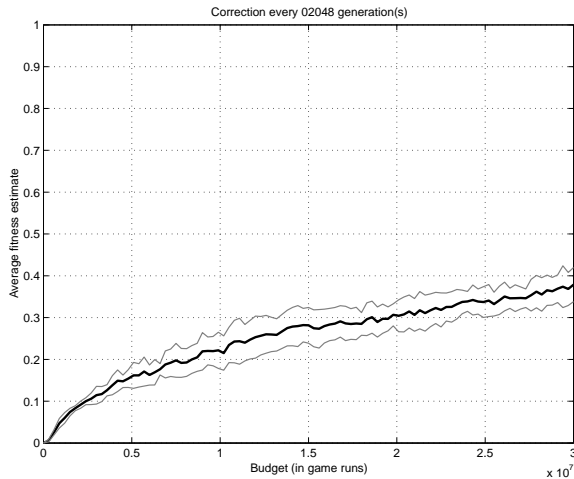


Figure 7: Meta EA Model

#### 4.4 SENSITIVITY ANALYSIS

To date a number of sensitivity studies have been performed. Since, in the earlier experiments, the Island and Meta EA models each had cumulative population sizes of 300 while the standard EA had a population size of 100, direct performance comparisons seemed somewhat unfair since population size is known to affect fixed budget performance. So, for both the Island and the Meta EA models, the population size was reduced from 100 to 34 individuals. The results showed that, in order to avoid excessive variability in the case of Island Model, the synchronization must be performed at intervals of 256 or more generations. This was somewhat to be expected due to the fact that an average of 100 gives a more accurate estimate than the average of 34 individual fitnesses in each genera-

tion. Due to the same considerations the accuracy parameter starts showing a steady paced increase from  $N \geq 512$ . Unfortunately, the average estimated fitness, although progressing at a promising pace, fails to equal the results achieved when the population is 100. As far as the Meta EA Model, variability between runs is reduced if  $N > 64$ , but average fitness results in excess of .15 at the end of the run are only obtained for  $N > 256$ . These effects are likely due to the fact that the underlying EA is a generational GA for which small population sizes are known to introduce significant sampling errors.

A second important design decision for a sensitivity analysis is the mechanism for adapting the accuracy parameter. Recall that this was implemented as a mapping function that transforms the accuracy parameter into a number of games per evaluation. This function controls the speed and magnitude of adjustment. Initially it was set to be the 2 based exponential. A linear regime, a logarithmic one and the double exponential were tried out. They all were scaled as to map the same  $[0, 10]$  range to  $[1, 1024]$  - just like the exponential. The results indicate that if the accuracy parameter is allowed to grow too fast at the beginning, it will do so and such behavior will be detrimental. The logarithmic schedule performed worse then linear schedule which in turn performed worse then exponential. The double exponential was barely distinguishable from the exponential schedule because for small values of the argument, the exponential itself gives very small variations. The double exponential, constrained to be lower then exponential, yet of integer values (number of games per evaluations) had no choice but to keep constant or vary insignificantly.

## 5 CONCLUSIONS

Our preliminary experiments suggest that the Meta EA Model performs better than the Island Model. Yet both of them lag behind running an EA with *a priori* known good parameters. However, out of fairness, a critical set of experiments needs to be performed, namely, evaluating the performance of a standard EA with a population size of 300. Preliminary results suggest that in this case they are comparable to the Meta EA model.

In fact, even the results presented here are somewhat encouraging in that, in spite of its much larger population size, the Meta EA Model synchronizing every 2048 generations, produce results comparable to the best results obtained by running with constant accuracy.

## 6 FUTURE WORK

Clearly the results presented here are preliminary in nature. As indicated earlier, more work needs to be done in studying the effects of population size. In general, longer runs are needed to better ascertain the long term behavior of the adaptation. Rather than a multi-population adaptation mechanism, there are a number of possibilities for sensitivity parameter update rules in single population EAs

### Acknowledgments

This research was supported by a variety of grants to the Evolutionary Computation Laboratory at George Mason University ([www.cs.gmu.edu/eclab](http://www.cs.gmu.edu/eclab)).

### References

- [Fitzpatrick and Grefenstette, 1988] Fitzpatrick, J. and Grefenstette, J. (1988). Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120.
- [Goldberg et al., 1992] Goldberg, D., Deb, K., and Clark, J. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362.
- [Hartley, 2002] Hartley, R. (2002). The long term behavior of genetic algorithms with stochastic evaluation. In *To appear in Foundations of Genetic Algorithms 7*. Morgan Kaufmann.
- [Spears and Gordon, 2000] Spears, W. M. and Gordon, D. F. (2000). Evolving finite-state machine strategies for protecting resources. In *International Symposium on Methodologies for Intelligent Systems (ISMIS-2000)*, pages 166–175.