Dynamic Populations in Genetic Algorithms

Zhanshan (Sam) Ma
University of Idaho
Computer Science Dept.
Moscow, ID 83843, USA
sam@cs.uidaho.edu

Axel W. Krings
University of Idaho
Computer Science Dept.
Moscow, ID 83843, USA
krings@cs.uidaho.edu

ABSTRACT

Biological populations are dynamic in both space and time, that is, the population size of a species fluctuates across their habitats over time. There are rarely any *static* or *fixed size* populations in nature. In evolutionary computation (EC), population size is one of the most important parameters and it received attention from EC pioneers from the very beginning. Despite many attempts to optimize the population sizing, the prevailing scheme in EC is still possibly the simplest — the fixed size population. This is in strong contrast with population entities in nature. In this paper, we explore the effects of dynamic (fluctuating) populations on the performance of genetic algorithms (GA). In particular, we test five dynamic population-sizing patterns: *random fluctuating population*, *increasing population*, *decreasing population*, *bell-shaped population*, and *inverse bell-shaped population* and compare them against the fixed size population. Our experiment shows very promising results that the dynamic populations perform more efficiently than the traditional fixed size populations, in terms of the number of fitness function evaluations and memory space requirements. We also analyze why the dynamic populations should perform superior to the fixed size populations from the biological perspective.

Categories and Subject Descriptors
Evolutionary Computation, Genetic Algorithms.

General Terms Algorithms.

Keywords: Dynamic Population, Fluctuating Population, Genetic Algorithms, Evolutionary Computation, Ecological Computation.

1. INTRODUCTION

One of the trademark features of evolutionary computation is

the use of a group of operators, rather than a single operator, to search the solution space. These operators are termed individuals, and they form a population. Both the concepts of individual and population are inspired by biology. Obviously, population sizing is a hardly escapable issue for any evolutionary computation paradigms. Therefore, it comes as no surprise that the pioneers of evolutionary computing, such as De Jung (1975) [1], Holland (1975) [6], Goldberg (1989) [3], already studied the population sizing issues from the very beginning of EC. What is perhaps surprising is that more than three decades later, we still largely depend on the experience or *ad hoc* trial-and-error approach to set the population size. In their recent monograph for evolutionary computing, Eiben and Smith (2003) indicated: "*In almost all EA applications, the population size is constant and does not change during the evolutionary search*" [2]. Multiple factors may have contributed to the establishment of the *status quo ante*. We believe two of those factors are particularly worthy of mentioning. The first is the dominant influence of population genetics. Darwin founded the evolutionary theory, but it was the population geneticists, such as Fisher and Wright, who first developed the mathematical theory of evolution and united the evolution theory with genetics. Two of the earliest population genetics models, the Wright-Fisher population model and the Hardy-Weinberg law, assumed population size is constant or infinite. The reality is that the fixed size in early population genetics models is nothing more than a mathematical convenience. We believe the next reason for the *status quo* is that the population sizing is deceivingly simple. In reality, it is an extremely complex issue.

It seems to us that to address the first factor, evolutionary computation should be more active in embracing ecological aspects of evolutionary theory. Genetics is only one of the foundations of evolutionary theory, and Darwin actually formed his evolution theory unbeknownst to the genetic materials. The mathematical theory of evolution can be derived from ecological perspectives without explicitly invoking genetics. For the second factor, we believe that new approaches and thinking might be necessary to achieve breakthroughs in the field.

In this paper, we report our preliminary experimental exploration of population sizing. In section 2, we argue that dynamic (fluctuating) populations should be more appropriate for evolutionary computation than fixed size populations, based on the ecological principles of natural population dynamics. We then present test results of our hypothesis with the schemes that mimic the population dynamics in nature.

2. ECOLOGICAL PRINCIPLES OF POPULATION DYNAMICS AND EXPERIMENT DESIGN.

2.1. Ecological Principle

Biological populations are dynamic in both space and time. In other words, the population size of a species fluctuates across their habitats over time. There are hardly any *static* or *fixed size* populations in nature. The closest to static or fixed size populations in the biological world are the populations that reach the so-called steady equilibrium states, a concept many biologists consider only exists in simplified mathematical models. On the contrary, biological populations frequently demonstrate stochastic fluctuations, and catastrophic or chaotic jumps and falls in many field animal populations are the norm rather than the exception.

It is believed that population dynamics is regulated by extremely complex mechanisms. Many mathematical models have been proposed to study population dynamics, from Sigmoid curve, which originated in population study and later found numerous applications in other fields, Power Law and Chaos theory models, to more realistic large-scale stochastic simulation models. If we can conceive the different time scales between "ecological time" (tens or hundreds of years) and "evolutionary time" (in thousands or millions of years), population dynamics (or size changes) is probably the most appropriate parameter to measure the "fitness" of a species. This is because population dynamics demonstrate the population's capability in exploring and exploiting their potential habitats. Furthermore, the interaction between populations and their environment are frequently instantaneous.

It is recognized that evolutionary computing is a precise subject dealing with algorithms, and it does not have to map or mirror the biological realities exactly, to be an effective algorithm. However, the fact that both nature and evolutionary computing follow Darwin's evolution principles, and that population dynamics is the most fundamental property of a species (especially in ecological time), makes one wonder what implications may be generated when the dynamics mechanisms exhibited by natural populations are introduced into evolutionary computing. Based on this consideration, we try to seek inspirations from insect population dynamics in nature.

We adopt the following five simple schemes to mimic the natural insect population dynamics. These are: the random fluctuating population, increasing population, decreasing population, "bell-shaped" population (also named "mountain-shaped) and inverse "bell-shaped" population (also named "valley-shaped" population). Their meanings are largely self-explanatory, and our motivation for choosing them will become clear when we introduce the experiment results in the next section.

A comprehensive and comparative study of the implications of biological population dynamics in evolutionary computing is beyond the scope of this paper. Therefore, in the paper, we directly focus on one central issue of the population sizing, that is, do fluctuating populations perform better than conventional fixed size populations in evolutionary computing? We generally use the terms "fluctuating populations", "dynamic populations", "variable populations", and population dynamics interchangeably in this article. Although these terms are not exactly the same in biology and each of them has a precise definition (in biology), we found that it is still premature to define them accurately in the context of evolutionary computing. We believe that this interchangeable usage will not cause undue confusion.

2.2. Test Problem and Experiment Design

Assume a problem is represented as a string of bits of size L=32. This string is broken up into blocks of length B. The problem is to find blocks in which all bits are set to one. The fitness function is the number of blocks satisfying this requirement. If B does not divide L evenly, the leftmost block will be of different size. The following are four examples of bit streams and their corresponding fitness values (F). The first and third have optimum solutions.

L = 32, B = 4, F=8: 1111 1111 1111 1111 1111 1111 1111 1111
L = 32, B = 4, F=2: 0000 1111 0110 0100 0111 1010 1111 1010
L = 32, B = 7, F=5: 1111 1111111 1111111 1111111 1111111
L = 32, B = 7, F=2: 1111 0000110 0001000 1111111 0011010

The hypothesis we try to test in the experiment with the standard Genetic Algorithm (GA) is whether the fluctuating populations (dynamic population sizes) outperform the conventional fixed-size populations. Although complex mathematical models can be used to simulate the population dynamics between generations, for simplicity, we choose to implement a set of simple schemes that characterize population dynamics as three *stages*: *beginning*, *intermediate*, and *ending* periods. We also characterize a population *size-type (*or simply *type)* as *small, medium* or *large,* in terms of the population size. We assign various population *types* to different *stages*. Within each combination of the *stage* and *type*, the population fluctuates randomly within controlled *ranges*. The controlled *range* is an interval between 2 and the *range-ceiling*, [2, *range-ceiling*]. The floor of the interval 2 represents the minimum population size, that is, two individuals. The *range-ceiling* is the maximum size the population can reach in the interval. The population fluctuation is designed to follow the *uniform distribution* with parameters, *a* = 2 and *b* = *range-ceiling*. Therefore, the mathematical expectation or mean equals (*b*+*a*)/2, or (*range-ceiling*+2)/2, according to the statistics of the uniform distribution.

The *small*, *medium* and *large* size types are quantified as follows: [2, *psize/3*] *for small* population, (*psize*/3, 2**psize*/3] for *medium* population, and (2**psize*/3, *psize*] for *large* population, respectively. That is, the whole population size interval [2, *psize*] is divided equally into three segments or sub-intervals. Obviously, each of the three sub-intervals has its own ceiling; they are *psize/3,* 2**psize*/3, *psize* for *small*, *medium*, and *large*, respectively.

With the *stage* and *size-type* defined, we devise five dynamic population schemes as well as the fixed population size, which acts as the default *control* treatment.

(1) ***Fixed Size Population***. This is the default in standard GA. Let *p-size* be the size of population, which is fixed throughout generational iterations.

(2) ***Random Fluctuating Population***. This is the basic mode for our fluctuating population schemes. The population fluctuates randomly in the interval of [2, *psize*], following the *uniform* random distribution. Therefore, the mean population size in this scheme is equal to (*psize*+2)/2. It is noted that the interval ceiling

psize here is set to equal the *psize* in the fixed population size in our experiment. This does not have to be the case, as we will explain later.

(3) ***Increasing Population*** is characterized by the initial *small* population in the *beginning*, *medium* sized population in the *intermediate* stage, and *large* sized population in the *ending* stage. As pointed out previously, within each *stage*, population randomly fluctuates in a *range* determined by the *range-ceiling* of specific *size-type*. The maximum population size of this scheme is *psize*, which takes the same value of *psize* as in the standard GA in our experiment, but this does not have to be the case either.

(4) ***Decreasing Population*** is characterized with the initial *large* population in the beginning, *medium* sized population in the *intermediate* stage, and *small* population in the *ending* period. This is in contrast with the "increasing population" scheme; all the other characteristics are the same as in the *increasing populations*.

(5) **Bell-shaped Population** (also named as "**Mountain-shaped**" population) is characterized by the *medium* population sizes in the *beginning* and *ending* periods, and *large* population size in the *intermediate* stage. The other characteristics can be inferred similarly to the previous ones.

(6) **Inverse Bell-shaped Population** (also named as "**Valley-shaped**" population) is characterized by the *large* population sizes in the *beginning* and *ending* periods, and *medium* population size in the *intermediate* stage. The other characteristics can be inferred similarly to the *bell-shaped population* as mentioned above.

In summary, with the above schemes, the scheme (1) acts as control treatment, with the fixed size population, *psize*. In the dynamic population schemes, (2)-(6), the population size fluctuates randomly, but the maximum size will never exceed the *psize*, the size set for the fixed size population in (1). The mean population sizes can be computed from the above definitions of *stage* and *size-type* and should vary from scheme to scheme, ranging from approximately 33% to 50% of the *psize*. In addition, each of the fluctuating population schemes has different patterns of fluctuation, as characterized by the unique combination of the *stage* (*beginning, intermediate,* or *ending*) and the *size-type* (*small*, *medium*, or *large*).

2.3. Why adopt such a complex experiment design?

One may wonder why we use such a complex experiment design. Besides mimicking the natural populations, the other major objective is to devise valid comparisons between the fixed size populations and the dynamic populations. The design we take might be somewhat counter-intuitive. Many of the previous studies on population-sizing focused on searching for the optimum population size, or a simple procedure or formula. We consider that this might be one of the important reasons for the *status quo* in the population sizing, since we doubt whether there is a simple answer. Even if there is an optimum population size, it may be a "moving" target and may not be easily captured by a straightforward formula. Most of the existing studies often test multiple population sizes, say, 16, 32, 64, 128, ..., and then try to find the *optimum* one. The problem with this methodology is obvious. For example, there may not be a fixed optimum size and, even if there is one, it may not be in the range being tested. Furthermore, there are potentially infinite numbers of candidates of optimum values to test. Given these recognized difficulties, our strategy is to set a *fixed task*, and then test which scheme is more efficient. The smaller the population size is, the more efficient the scheme is, since a small population requires fewer fitness evaluations and memory storage space.

For example, to compare the fixed size population with the *random fluctuating population*, let the population size for the standard fixed-size population GA be, *psize* = 100. Accordingly, let the population size for the *random fluctuating population* be controlled by a random variable with uniform distribution in the *range* [2, *psize*]. That is, the maximum population size the random fluctuating scheme may ever reach is equal to the fixed population size, *psize* = 100, in this case. The mean size of the fluctuating population is therefore, (*psize*+2)/2 = 51 ≈ 50 = *psize**50%. Similarly, we can compute the mean population sizes for the other schemes, which range from 33% to 50% of the *psize*. We hope, by controlling the *interval ceiling*, which equals *psize* in this case, that the random fluctuating populations can still be comparable with the fixed size populations.

It is obvious that the maximum fitness evaluations for the randomly fluctuating population are only 1/2 of those in the fixed size population. One may immediately notice that the fluctuating populations are artificially disadvantaged in the comparisons by our schemes. That is, the mean size of the randomly fluctuating population is at most 50% of the fixed size population. An alternative, which would be fair to both the fluctuating and fixed size populations, is to set the *ceiling* in random fluctuating population to double the size of the fixed-size population (2**psize*), since then the mean population size for the fluctuating population would be equal to the *psize*. We were concerned that the artificial disadvantage against the fluctuating population may turn out to be an advantage for it. We found that the counter-intuitive scenario is unlikely for two reasons: (1) Although the scheme limits the average population size in fluctuating population to only 1/2 of the fixed size population, our comparison is based on the *actual count of the number of fitness evaluations*, *not* on the pre-assigned ceilings. In other words, we *counted and compared the accumulated population size before the all 1-bits string is found*, or the actual number of individuals "mobilized" in the search for optimum solutions. (2) In pre-testing, we tried to estimate the approximate *threshold* of *psize*, the size at which fixed population can find the solution in a reasonably short time. In other words, if we set *psize* smaller than the *threshold* psize, with the fixed-size population scheme we may fail to obtain the solution at all within a reasonable period. This also explains our choices of test values reported in Table 2. On the other hand, the fluctuating population rarely needs more than 1/2 of the *psize* to find solutions. This pre-testing is similar to what many existing studies have adopted, such as setting psize to 32, 64, 128, ..., from which we get the approximate *threshold* values for further testing.

3. RESULTS AND DISCUSSION

3.1 Results

The experimental results are summarized in Tables 1 and 2. In Table 1, *max trials* refers to the maximum number of tests run. Each test is an independent experiment; the multiple trials are run to calculate the statistics of performance parameters. The other parameters are standard GA parameters. *Max Generations* refer to the total number of generations iterated per trial.

Table 1. The GA Parameters Except for the Block and Population Sizes.

Max Trials	100
Max Generations	1000
Crossover Prob.	0.9 (2-pt crossover)
Mutation	1/32 (bit flip)
Initial Population	32-bits random strings
Selection	best 2 elitism
Halting	find optimum or exceed the maximum (10^5)

Each experiment starts with a population of randomly generated 32-bits strings of 0 and 1, or the chromosomes. The program stops whenever the optimum solution, all 1-bits string (111...111), is found, or reaches the maximum evaluations set by the product of maximum trials and the generations per trial. In the latter case, it may fail to find the optimum solution.

We define the parameter ***PopuEvalIndex*** as the product of (*psizeReal*)*(*meanGenerations*), that is,

PopuEvalIndex = [(***psizeReal)*(meanGenerations***)]

As explained previously, due to the additional complexity with the fluctuating populations, we actually count the population sizes in the program and compute the mean population size per generation, ***psizeReal***. Similarly, the parameter ***meanGenerations*** is the mean of the generations iterated per trial, before it finds the optimum all 1-bits string. It is computed as the total generations divided by the total number of *trials*.

We therefore use the parameter ***PopuEvalIndex,*** defined above, as a measure to compare dynamic population schemes with the standard fixed-size populations. The small ***PopuEvalIndex*** indicates that fewer evaluations of fitness functions and other associated computations (mutations, crossovers) are needed. In addition, it also implies that less memory space is needed for storing the individual information. Although the exact quantification of the savings may be complex, the difference in the magnitude of *PopuEvalIndex* should be a good indicator for the performance of each scheme. Table (2) is a summary of the values of the ***PopuEvalIndex*** from our experiments. .

Table 2. Comparisons of Fitness Evaluations (*PopuEvalIndex*) for Various Population Sizing Schemes.

Treatment (B, psize)	***Fixed Population***	***Random Population***	***Increasing Population***	***Decreasing Population***	***"Mountain"-Shaped P.***	***"Valley"-Shaped P.***
(1, 64)	59171	29779 [50%]	N/A*	22560 [38%]	26470 [45%]	27072 [46%]
(4, 64)	N/A*	32035 [N/A]	N/A*	24269	26734	29123
(7, 128)	124474	61503 [49%]	N/A*	47088 [38%]	53468 [43%]	56698 [46%]
(8, 256)	247424	125537 [51%]	93185 [38%]	94152 [38%]	101826 [41%]	114748 [46%]
(14, 512)	504960	252567 [50%]	191000 [38%]	190168 [38%]	213000 [42%]	230777 [46%]
(17, 1024)	982497	N/A*	377254 [38%]	N/A*	N/A*	N/A*

* N/A, Failed to find the optimum fitness with the fixed population. Consequently, the comparisons (%) cannot be computed.

In Table 2, the first column displays the various treatments we tested. The first element in the pair is the block size (B), and the second element is the *psize*. As explained in detail in section 2, *psize* is the population size for the fixed populations, but for the fluctuating populations, it serves as the ceilings for the population-sizing interval [2, *psize*]. The percentage, e. g, [38%], following each ***PopuEvalIndex,*** is computed as the ***PopuEvalIndex*** for the specific scheme divided by the ***PopuEvalIndex*** for the corresponding fixed size population in the same treatment. It indicates the relative performance difference.

From Table 2, we draw the following preliminary conclusions:

(1) The results reveal that the fluctuating populations consistently outperform the fixed size populations, except for some cases of the *increasing population* scheme. The improvement percentages range from 38% to 50% approximately. The *increasing population* scheme failed to find the optimum solution in the three smaller block sizes (B = 1, 4, 7), but performed equally well in the three larger block sizes (B = 8, 14, 17), compared with the other fluctuating populations. The fixed size population also failed in the case of block size B=4. Whenever there is a failure to find the optimum solution, we skip the comparisons. In every case that the comparison is made, the fluctuating population prevails.

(2) The failure to find the optimum solution with the "*increasing population*" scheme is interesting to note. Although this is in contrast with the natural populations, it may indeed make sense in EC, at least with some problems such as the one we experimented with. This implies that at the beginning, more individuals (a large population size) are needed when the search space is huge.

What is also noteworthy is that, in the case of block size = 17, the *increasing population* indeed found the optimum solution, but the other fluctuating schemes failed. This is in strong contrast to the phenomenon that the *increasing population* failed in the three smaller block sizes. It also suggests that, for the large block size (B), the large population size has advantage. We re-ran the failed cases with the increased *range-ceiling,* but still bounded by the constraint, *range-ceiling* ≤ 2**psize*. The constraint guarantees that the mean population size in fluctuation populations will still not exceed the *psize*, the value set for the fixed size population to be compared. With the increased size, but still bounded by the constraint, we were able to find the optimum solutions. This means that under a fair condition — both fixed and fluctuating populations are of the same size — fluctuating populations can perform just as well as fixed populations. In other words, in these cases our experiment design disadvantages the fluctuating population too severely by restricting their average size to only 1/2 of the fixed population size (*psize*).

(3) At this stage, we do not draw definite conclusions about the differences among the various fluctuating population schemes. The focus of this preliminary experiment is to test whether the fluctuating or dynamic populations generally outperform the traditional fixed size populations. We are continuing the study to further characterize the fluctuating populations.

3.2 Discussion

Despite the consistent and promising test results, we are not overly optimistic about the potential of devising a universally applicable procedure or formula for sizing the populations in EC. Given the establishment of the *no-free-lunch theorem* (NFL) [7], we tend to think that the fluctuating populations may simply push the "bounds" tighter. In other words, the traditional fixed size populations simply waste resources and new schemes are more efficient because they adjust population sizes dynamically. What happens here, and perhaps in large part of the evolutionary computation, may be that we are simply trying to push the obstacles as far away as possible from the boundary. Nevertheless, the improvement in tightening the bounds is still of significant practical values.

We suggest that one of the most convenient approaches to taking advantages from the fluctuating populations can be using it as a more natural *survival selection* mechanism. This is also consistent with the ecological principles in nature in the *ecological times*. Nature never assigns an exact quota to a population or a species. A species' global population size on earth, in space and time, is determined by its fitness. The fitness itself is dynamic, depending on its adaptability to the environment. Some species become extinct, but others prosper. The most obvious demonstration of the prosperity of a species is the size of its populations. Nature has both positive and negative feedback mechanisms to regulate population size, which leads to constant fluctuations. If we accept the notion that, in nature, the selection that occurred in ecological times, can simply be expressed as the fluctuation of population numbers, then the principle that the fittest survives means that a species' populations *accumulate* more individuals. This may be a more natural selection mechanism than what is currently adopted in the *survivor selection* in evolutionary computing. The problem with some of the current survivor selection mechanisms, which keep the fixed size population from generation to generation, is arguably analogous to using the same number of soldiers in different stages or battles of a war. The ideal situation is to adjust the numbers based on strategies, tactics, logistics, etc. We hope that this study will stimulate the efforts to use the fluctuating population as a more natural *survival selection* mechanism.

Finally, it should be pointed out that there exist some excellent studies on population sizing, both experimentally and theoretically, such as [4][5], besides the early research conducted by EC pioneers such as [1][3][6]. Due to the space constraints, we can only focus on reporting our preliminary exploration on this issue. Regrettably, we have to delay the literature review to a follow-up research on the same topic.

4. ACKNOWLEDGMENTS

We wish to thank Dr. Robert Heckendorn for reviewing the early draft of this paper and his comments. We also wish to thank Dr. Robert Hiromoto for engaging in stimulating discussions on this topic. We appreciate three anonymous reviewers for their insightful comments and suggestions.

5. REFERENCES

[1] DeJung, K. A. 1975. An analysis of the behaviors of genetic adaptive systems. Ph.D. Thesis, University of Michigan, MI.

[2] Eiben, A. E. & J. E. Smith 2003. *Introduction to Evolutionary Computing*. Springer.

[3] Goldberg, D. E. 1989. Sizing populations for serial and parallel genetic algorithms. in "*Proceedings of the Third International Conference on Genetic Algorithms*", ed. by David Schaffer. Morgan Kaufman Publishers, pp70-79.

[4] Goldberg, D. E. et al. 1992. Genetic algorithms, Noise, and the Sizing of Populations. *Complex Systems*, 6:333-362.

[5] Harik, G., E. Cant˘u-Paz, D. E. Goldberg, and B. L. Miller. 1999. The Gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evol. Comput.*, vol. 7(3):231–253.

[6] Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[7] Wolpert, D. H. and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. Evol. Comp*. 1(1):67-82