A Review of Adaptive Population Sizing Schemes in Genetic Algorithms

Fernando G. Lobo
DEEI-FCT
University of Algarve
Campus de Gambelas
8000-117 Faro, Portugal
flobo@ualg.pt

Cláudio F. Lima
DEEI-FCT
University of Algarve
Campus de Gambelas
8000-117 Faro, Portugal
clima@ualg.pt

ABSTRACT

This paper reviews the topic of population sizing in genetic algorithms. It starts by revisiting theoretical models which rely on a facetwise decomposition of genetic algorithms, and then moves on to various self-adjusting population sizing schemes that have been proposed in the literature. The paper ends with recommendations for those who design and compare adaptive population sizing schemes for genetic algorithms.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search; I.2.6 [Artificial Intelligence]: Learning.

General Terms: Algorithms, Performance.

Keywords: Genetic Algorithms, Parameter Setting, Population Sizing.

1. INTRODUCTION

The population size is a critical parameter in a genetic algorithm (GA). Too small and the GA converges to poor solutions. Too large and the GA spends unnecessary computational resources. The intuition behind population sizing in GAs is that it is related to the problem's size and difficulty; the larger the problem is and the more difficult a problem is, the larger the population should be. Problem difficulty is very hard to estimate on real world problems, and therefore, many users end up either using a so-called "standard setting" (50-100 individuals), guessing a number, or doing some experimentation with a number of different sizes to see which one works best. Guessing right is pure luck, and most likely a user guesses wrong by choosing a population size that is either too small or too large for the problem.

This paper is divided in two parts. In the first part, it reviews existing theoretical work on population sizing. That work is essentially based on Goldberg's facetwise decomposition for designing competent GAs [5] and relies on the notion of a building block (BB). The second part of the paper reviews a number of techniques that have been developed to adjust the population size during the GA run itself. The paper ends with recommendations for those who propose and compare adaptive population sizing schemes for genetic algorithms.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

2. POPULATION SIZING THEORY

In this section we review relevant theoretical studies that help our understanding on the role that the population size has in terms of GA performance and solution quality. Most of these studies use a facetwise approach to get a better insight in population sizing dynamics, and for that some assumptions are made: (1) work with selectorecombinative GAs (no use of mutation), (2) use fixed-length and binary-coded strings, (3) use fixed-size and non-overlapping populations, and (4) solve stationary objective functions. Although these assumptions are made for computational and analytical tractability, most of them can be relaxed with small or no changes.

Despite these assumptions and the operational simplicity of GAs, they are still complex systems to analyze. To have a better understanding on how to design better GAs a decomposition approach [5] has been taken by many. In this way, it is assumed that the problem to be solved is additively decomposable in a number of m subfunctions. Each subfunction maps to k decision variables and corresponds to a building block partition. Under this definition, it is possible to have 2^k BBs in a partition, one of which is superior to the others and belongs to the global optimal solution.

We start by reviewing the work on initial supply of BBs. Next, the issue of deciding well between competing BBs is revisited and major results are presented under the form of *dimensionless models* that relate the population size (n) with certain problem features, such as problem size (l), BB size (k), number of subfunctions (or BB partitions) (m), and others. Finally, the population size requirements for correct identification and mixing of BBs is addressed in the context of estimation of distribution algorithms (EDAs) [12, 17].

2.1 Initial Supply of Building Blocks

When using selectorecombinative GAs (no mutation), the only source of diversity is the supply of raw BBs in the initial generation. Once the randomly initialized population is filled up with enough BBs, it is more likely that the GA will be able to propagate and mix them, converging to good so-

lutions. A simple supply model considers the number of BBs present in the initial random population (generated under a uniform distribution), where the probability to generate a single BB of size k is $1/2^k$, for binary encodings. Therefore, the initial supply of BBs can be estimated as $x_0 = n/2^k$. From this simple relation it is possible to notice that the population required to supply all possible BBs grows exponentially with the BB size k. This suggests that problems with short BBs require smaller populations than the ones with longer BBs.

2.2 Decision Making Between Competing Building Blocks

The second aspect where population size plays an important role is in the decision-making between competing BBs. Holland [11] early recognized this issue as a statistical decision problem. He recasted this problem as a cluster of parallel and interconnected two-armed bandit problems. Although this is an idealization of the GA behavior, his results gave an optimistic bound on the allocation of trials in a GA. De Jong [3] also used this analogy and presented equations for trial allocation that removed some of the assumptions made by Holland [11]. He also recognized the role of signal-to-noise ratio on the population sizing question. Years later, Goldberg and Rudnick [7] presented a method to calculate the variance of schema fitness using Walsh transforms, and also proposed an estimate for the population size based on the variance of fitness. Following this work, Goldberg, Deb, and Clark [6] proposed a decision-based model that gives a conservative bound on the convergence quality of selectorecombinative GAs. The model focus on the correct decision between the best BB and its closer competitor in a given partition. This decision process takes place in the presence of noise that comes from the other partitions.

Let us consider a competition between two individuals that contain the best BB and the second best BB (in the partition of interest). The GA will probably choose the individual with the best BB given its higher fitness contribution to the overall fitness. However, in a single trial, the wrong decision can also be made because the remaining $m-1$ partitions also contribute to the fitness of the individuals in competition, and act as noise in the decision making process in the particular partition. By increasing the population size, it is possible to make this decision error as small as possible.

Based on this observation, Goldberg et al. [6] derived a population sizing equation which gave, for additive decomposable problems with uniformly scaled BBs, the required population size (in order to correctly solve each BB with a specified error probability α) in terms of the BB size k, the total number of BBs m, and the fitness difference d between the best and second best BBs.

Goldberg et al. [6] confirmed that the equation conservatively estimates the convergence quality of the GA. This somewhat pessimistic estimation is due to the fact that the model approximates the behavior of the GA by the outcome of the first generation. If the wrong BBs are selected in the initial generation the model assumes that the GA will be unable to recover from the error.

A few years later, Harik, Cantu-Paz, Goldberg, and Miller [8] refined this model by incorporating the initial BB supply as well as cumulative effects of decision making over an entire GA run. Their model was based on the well-known gambler's ruin problem and resulted in the following population sizing equation

$$n = -2^{k-1} \ln(\alpha) \sqrt{\pi (m-1)} \frac{\sigma_{BB}}{d}. \quad (1)$$

with σ_{BB} is the fitness variance of the BB partition. Experimental results from the original paper show that this equation is a reliable estimate for population sizing on decomposable problems with uniformly-scaled and near-uniformly-scaled building blocks. Additionally, the authors also proposed an expression to population sizing in the presence of exogenous noise as well as for different selection pressures.

2.2.1 Random Genetic Drift in Building Blocks

The gambler's ruin model is accurate for problems with uniformly-scaled and near-uniformly-scaled BBs. If the fitness contribution to the overall fitness function is exponentially-scaled the model can not estimate the population size correctly. For that case, Thierens, Goldberg, and Pereira [25], for BBs with unit size, and later, Lobo, Goldberg, and Pelikan [15], for a general BB size, developed a model based on the concepts of domino convergence and genetic drift.

In this kind of problems, BBs converge in a sequential manner from the most salient BBs to the least salient ones. The least salient BBs only get the attention of the selection operator later in the run, and may be lost from the population due to chance variation alone, a process known as *random genetic drift*. It has been shown that the drift process is the crucial aspect regarding population sizing for this type of problems. That is, the population size has to be large enough to prevent the low salient BBs from getting extinct from the population during the GA run. The theoretical models for exponentially-scaled BBs reveal that the population size should grow as $\mathcal{O}(m)$, instead of $\mathcal{O}(\sqrt{m})$ for the uniformly-scaled case.

2.3 Proper Identification and Mixing of Building Blocks

The issue of correct identification and mixing of BBs can be related to the population size if we consider a recent class of GAs known as estimation of distribution algorithms (EDAs) [12, 17]. In EDAs the underlying problem structure is learned on-the-fly for proper identification and mixing of BBs. EDAs have very strong connections with Data Mining algorithms. Both interpret the population as data and use techniques to infer patterns in that data that are associated with high fitness. Without sufficient data it's not possible to do accurate inference. Fortunately, within a GA framework, having more data translates into working with larger populations.

EDAs use probabilistic models to induce non-linearities between variables in order to detect the proper decomposition of the problem (BB partitions). In order to be able to do that the EDA requires a minimal population size to detect the correlations. Pelikan, Goldberg, and Cantu-Paz [16] were the first to approach this issue, reporting that the minimum population size required for BOA to correctly detect the problem structure in the first generation grows linearly with the number of BBs. Recently, that study has been extended [20], and in order to effectively capture the problem structure, the population size of BOA should be greater than $\mathcal{O}(l^{1.05})$ and smaller than $\mathcal{O}(l^{2.1})$, where l is the problem size. The upper bound has to be respected in order to

avoid that the probabilistic models discover dependencies between independent variables. Typically, the population size requirements for initial supply and decision-making are bounded by the requirements for correct model building.

Before finishing this part of the paper, we would like to stress that although not trivial to put in practice, the theoretical models on population sizing are very important and crucial for understanding the role of the population in a GA. Among other things, an important lesson of those models is that setting the population size to a fixed value regardless of the problem's size and difficulty, is certainly a mistake.

3. ADAPTIVE POPULATION SIZING SCHEMES

This section reviews a number of techniques that have been developed to adjust the population size during the GA run itself. The motivation for developing these schemes are essentially threefold: (1) the recognition that the population sizing requirement is problem dependent and is hard to estimate, (2) the observation that a GA with an adaptive population sizing scheme may yield a better solution quality and better performance than a GA with a fixed (and poorly set) population size, and (3) to make life easier for users by eliminating the population sizing parameter. The exposition follows in chronological order.

3.1 Population sizing through estimates of schema variances (1993)

Based on Goldberg et al.'s. [6] theoretical work, Smith and Smuda [23, 24] suggested an algorithm to autonomously adjust the population size as the search progresses. They argued that the parameters of the GA are hard to relate to the user's need. That is, the user typically doesn't know how the population size affects the solution quality of the problem. The authors suggested that the parameters should have more meaning from the user's point of view, and proposed an algorithm that only required the user to specify a desired accuracy for the overall search, something equivalent to the building block's signal from Goldberg et al.'s equation. In order to do that, Smith and Smuda defined the *expected selection loss* between two competing schemata as the probability that the GA makes a selection error (selection chooses the lower fit schema) weighted by their fitness difference. Then, they suggested an algorithm that does online estimates of schema fitness variances, and sizes the population so that the expected selection loss approximates the user's specified target accuracy.

3.2 Population sizing in GAVaPS (1994)

Arabas, Michalewicz, and Mulawka [1] proposed the *Genetic Algorithm with Varying Population Size* (GAVaPS). As opposed to Smith and Smuda's work, this method does not rely on the population sizing theory [6]. Instead, it relies on the concept of *age* and *lifetime* of an individual. When an individual is created, either during the initial generation or through a variation operator, it has age zero. This step corresponds to the birth of the individual. Then, for each generation that the individual stays alive its age is incremented by 1.

At birth, every individual is assigned a lifetime which corresponds to the number of generations that the individual stays alive in the population. When the age exceeds the individual's lifetime, the individual dies and is eliminated from the population. At every generation, a fraction ρ (called the *reproduction ratio*) of the current population is allowed to reproduce. Every individual of the population has an equal probability of being chosen for reproduction. Thus, GAVaPS does not have an explicit selection operator as traditional GAs do. Instead, selection is achieved indirectly through the lifetime that is assigned to individuals. Those with above-average fitness have higher lifetimes than those with below-average fitness. The idea is that the better an individual is, the more time it should be allowed to stay in the population, and therefore increase the chance to propagate its traits to future individuals.

Arabas et al. said that different lifetime strategies could be implemented based on the idea of reinforcing individuals with high fitness and restricting individuals with low fitness, and suggested three different strategies: proportional, linear, and bi-linear allocation. All those strategies relied on two parameters, $MinLT$ and $MaxLT$, which correspond to the minimum and maximum lifetime value allowable for an individual. The authors argued that this approach seems natural because the aging process is well-known in all natural environments.

The authors tested GAVaPS on four test functions, compared its performance with that of a simple GA using a fixed population size, and observed that GAVaPS seemed to incorporate a self-tuning process of the population size. GAVaPS requires the user to specify the initial population size, but the authors refer in their work that GAVaPS is robust with respect to that, i.e., the initial population size seemed to have no influence on the performance on the test functions chosen. The same thing did not hold with the reproduction ratio ρ. Different values of ρ yielded different performance of the algorithm. For the $MinLT$ and $MaxLT$ parameters, Arabas et al. used 1 and 7 respectively. However, they didn't give recommendations on how those parameters should be set.

3.3 Strategy adaptation by competing subpopulations (1994, 1996)

Schlierkamp-Voosen and Mühlenbein [21] proposed an adaptive scheme which simulates the population changes of species which compete for the same resource (e.g. food). Well adapted species increase and poorly adapted species decrease.

In their scheme, there are a number of S competing populations, each one running a different search strategy (or algorithm). The search algorithms run independently in each population, and at regular intervals the populations compete with each other. The idea is to increase the size of the best performing population, and decrease the size of the others.

In order to do so, the authors define a *quality criterion* which scores the performance of a population, as well as a *gain criterion* which gives the amount of increase or decrease in the size of the competing populations. In their scheme, the sum of the sizes of all populations remains constant through time. After each competition, in addition to adapting the population sizes, the best performing individual migrates to the other populations, giving them a chance to perform better for future competitions.

Later on, the authors [22] extended the model by allowing a *consumption factor* γ to be assigned to each population.

The idea was motivated by the recognition that different search strategies seem to require different population sizes to perform efficiently. For example, mutation is more efficient with small populations and recombination works best with large populations. As opposed to the basic competing model, the extended one allowed the total population size to change during the whole simulation. According to the authors, the extended competition scheme is useful for location good regions of attraction with a breadth search algorithm, and to do a fine adaptation by using a more directed search method.

The scheme suggested by Schlierkamp-Voosen and Mühlenbein is, in some sense, an hybrid algorithm consisting of multiple search strategies, each utilizing different resources (different population sizes and potentially different variation operator). The overall scheme reinforces those strategies that appear to work well, and penalizes those that appear to work poorly.

3.4 Population sizing in SAGA (1996)

Hinterding, Michalewicz, and Peachey [10] proposed the *Self-Adaptive Genetic Algorithm* (SAGA) which included an adaptive population sizing scheme. The basic idea is to run three independent GAs, each with its own population size (call them $P1$, $P2$, $P3$ with $P1 < P2 < P3$) which are initially set to 50, 100, and 200, and have rules that adjust the population sizes in order to "optimize" the performance of the mid-sized population $P2$.

Hinterding et al. allowed the populations to range between 10 and 1000 individuals, and made sure that there should always be a difference of at least 20 individuals in their sizes in order to avoid population sizes which are too similar.

At regular intervals (an *epoch* in the authors' terminology), the best fitness found in each GA is used as a criteria to adjust the population sizes. The adjusting mechanisms are based on a number of rules. Specifically, there is a "move-apart" rule that states that two populations should be moved apart when the fitness of their best individuals are within a threshold $\epsilon = 10^{-9}$. In such cases, moving the populations apart means halving the size of the smallest population, and doubling the size of the largest population. This situation also holds for the case that all three populations are within the ϵ fitness threshold. In such case, the mid-size population is left untouched, while the others grow and shrink.

In case the fitness values of the best individuals of the three populations are different (i.e., not within ϵ distance), the following rules apply (below, f stands for the fitness of the best individual):

- if $f(P1) < f(P2) < f(P3)$: move right
- if $f(P3) < f(P2) < f(P1)$: move left
- if $f(P1) < f(P3) < f(P2)$ or $f(P2) < f(P1) < f(P3)$: compress left
- if $f(P2) < f(P3) < f(P1)$ or $f(P3) < f(P1) < f(P2)$: compress right

where 'move right' and 'move left' means doubling and halving the size of each population respectively, 'compress left' means adjusting the size of $P1$ to be the average of its current size with that of $P2$, and 'compress right' means adjusting the size of $P3$ to be the average of its current size with that of $P2$.

The reasoning behind these rules is to increase (or decrease) all the population sizes if the "ideal" population size appears to be out of the range of the current population sizes. On the other hand, if the "ideal" population size appears to be within the current population sizes, then the rules should adjust the size of the worst performing population so that its size becomes closer to the size of the other two populations.

The time span of an epoch was chosen by Hinterding et al. as 1000 fitness function evaluations for each GA. The authors also described different strategies that either allowed the populations to mix or to continue running separately after each epoch.

3.5 Population sizing in the parameter-less GA (1999)

The parameter-less GA introduced by Harik and Lobo [9] was developed having in mind the assumption that solution quality grows monotonously with the population size. Based on that observation, the authors suggested a scheme that continuously increases the population size as time goes by. Harik and Lobo made the assumption that given two identical GAs whose only difference is in their population sizes, call them *GA-bigPop* and *GA-smallPop*, then the following statement holds,

- if *GA-smallPop* is allowed to spend more fitness function evaluations than *GA-bigPop*, and if the average fitness obtained by *GA-bigPop* is greater than the average fitness of *GA-smallPop*, then there's a strong evidence that *GA-smallPop* is using an undersized population.

The above observation is the crucial aspect of the parameter-less GA and allows it to eliminate undersized populations. Figure 1 illustrates an example.

The paremeter-less GA establishes a competition among multiple populations, each with a size twice as large as the previous one, and starting with a very small sized population. Smaller populations are given an advantage because the algorithm allows them to spend more fitness function evaluations than larger populations. As time goes by, the smaller populations are eliminated and larger populations are created. The deletion of populations is controlled by inspecting the average fitness of the populations and eliminating the undersized ones as suggested in Figure 1. In the absence of mutation, the parameter-less GA also eliminates populations that converge since its not possible to generate new individuals thereafter.

The time allocation that is given to each population is orchestrated in such a way that a population gets the chance to execute m times more generations than the next immediately larger population. Harik and Lobo [9] suggested a value of $m = 4$ which allowed a given population to run 4 times more generations (2 times more function evaluations) than the next larger population. When to stop this growing process is left up to the user. He decides when to stop as soon as he realizes that it is not worth to wait more for an improvement in solution quality.

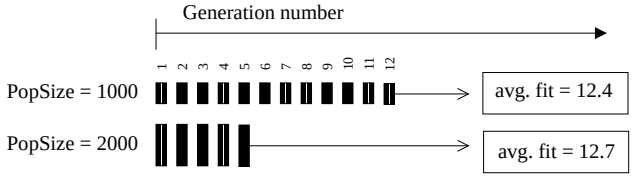

Figure 1: The GA with the population size 1000 is in its 12^{th} generation while the GA with population size 2000 is in its 5^{th} generation. Overall, the GA with the smaller population spent more fitness function evaluations that the other GA (12000 versus 10000). However, its average fitness is lower than the average fitness of the GA with the larger population. According to Harik and Lobo, that's a strong indication that 1000 individuals is not a sufficiently large population size.

Pelikan and Lobo [19] did a theoretical analysis for the worst-case performance of the parameter-less GA as compared to a GA that starts with an optimal population size. The worst-case scenario occurs when populations are never eliminated (either never converge, or are never overtaken by a larger population). That work revealed that a value of $m = 2$ is the better one in terms of the time complexity of the worst-case performance, yielding an increase in the number of fitness function evaluations of only a logarithmic factor when compared with a GA that uses an optimal population size.

This population sizing technique has been used not only with traditional GAs but also with EDAs [14, 18]. Lima and Lobo [13] also suggested a way to integrate local search with the parameter-less GA. They recognized that local search makes small variations on solutions and thus is not likely to benefit from very large populations. On the contrary algorithms that rely essentially on recombination require large population sizes in order to mix the bits and pieces of the different solutions. The approach suggested by the authors was run both search strategies independently and in some sense goes back to the same remarks made previously by Schlierkamp-Voosen and Mühlenbein [22] that different search algorithms require different type of resources.

3.6 Population sizing in APGA (2000)

The *Genetic Algorithm with Adaptive Population Size* (APGA) proposed by Bäck, Eiben, and Van der Vaart [2] is a slight variation of the GAVaPS algorithm. The difference between the two is that APGA is a steady-state GA and the best individual in the population does not get older. As opposed to Arabas et al., the authors of APGA set the values of *MinLT* and *MaxLT* to 1 and 11, because according to them, initial runs with different values indicated that *MaxLT=11* delivers good performance.

APGA needs the specification of an initial population size (Bäck et al. used 60 individuals in their experiments). At every iteration of the steady-state GA, all individuals (except the best one) grow older by 1 unit. Thus, it's quite likely that after *MaxLT* iterations, most of the individuals will have died and the only ones that remain in the population are either (1) the individuals generated during the last *MaxLT* iterations, or (2) top-fit individuals found previously (recall that the best individual doesn't get older). In other words, after *MaxLT* iterations the population size will be of order $\mathcal{O}(MaxLT)$.

By not decreasing the remaining lifetime of the best individual, APGA ends up using a strong kind of elitism. In their paper, the authors don't show the evolution of the population sizes through time, but in all their experiments, the average population size at the end of the runs were in the range between 7.8 and 14.1, which confirms our reasoning that the population size in APGA tends to be of the same order of *MaxLT*.

3.7 Population sizing in PRoFIGA (2004)

Eiben, Marchiori, and Valkó [4] and Valkó [26] proposed the *Population Resizing on Fitness Improvement GA* (PRoFIGA) and did a comparison of it with other adaptive population sizing schemes, including APGA and the parameter-less GA.

PRoFIGA is similar to a traditional GA but at the end of the typical selection, reproduction, and mutation steps, the population size can grow or shrink based on an improvement of the best fitness contained in the population. The population grows either when (1) there is an improvement in best fitness, or (2) there is no improvement in the best fitness for a "long time". If neither of the above occurs, the population size shrinks by a small percentage (1-5%). According to the authors, the motivation behind PRoFIGA is to use large population sizes for exploration and small population sizes for exploitation. In their work, the growth rate X for a population is given by,

$$X = increaseFactor \times (maxEvalNum - currEvalNum) \times \frac{maxFitness_{new} - maxFitness_{old}}{initMaxFitness}$$

where $increaseFactor$ is a parameter in the interval (0,1), $maxEvalNum$ is the maximum number of fitness evaluations allowed for the whole run (or a very large number if one does not know it ahead of time), $currEvalNum$ is the current evaluation number, and $maxFitness_{new}$, $maxFitness_{old}$, and $initMaxFitness$, are the best fitness values in the current, previous, and initial generation.

A user of PRoFIGA also has to specify an initial population size, as well as a minimum and maximum population sizes in which the algorithm must operate. Thus, although PRoFIGA eliminates the traditional fixed population sizing parameter, it introduces 6 new parameters: (1) initial population size, (2) $increaseFactor$, (3) the so-called "long time without improvement" (V), (4) $decreaseFactor$, (5) $minPopSize$, and (6) $maxPopSize$.

In their comparative experiments, the authors used $initialPopSize = 100$, $increaseFactor = 0.1$, $V = 500$, $decreaseFactor = 0.4$, $minPopSize = 15$, $maxPopSize = 1000$.

4. SUMMARY AND CONCLUSIONS

This paper has made a review of population sizing in genetic algorithms. We have addressed both theoretical models which rely on a facetwise decomposition of GAs, as well as automated population sizing schemes that have been proposed by various researchers in the field.

Some adaptive population sizing schemes that have been proposed were developed having in mind the population sizing theory for genetic algorithms as a foundation to engi-

neer the adaptive algorithms, other methods were primarily inspired by what happens in Nature, and included the concepts of *age*, *lifetime*, and competition among species for limited resources.

Several authors have argued that it is beneficial to have a varying population size through a GA run because it seems that the population sizing requirements differ depending on the stage of the search. Typically, these authors argue that the GA needs a larger population size early on in order to do a breadth search to find good regions of the search space, and then the GA needs less population size at the end of the run in order to fine-tune the solution.

We agree with the argument above. However, the critical aspect is to discover a population size which is large enough to find those good regions of attraction. At this point, we would like to stand back and give some recommendations for authors that propose and compare adaptive population sizing schemes.

- **Do not enforce a bound for the maximum population size.** Several adaptive population sizing algorithms have substituted the fixed population size by an interval *MinPopsize* and *MaxPopsize* where the adaptive algorithm is supposed to work. We believe that the upper bound for population sizing should not be present.

  Sizing a population in a GA to solve a particular problem is a difficult task and that's why many users end up choosing it in an ad hoc way. Some choose a value in the range 50-100, others simply guess a number. The end result of those practices is poor GA performance. Both theoretical and empirical work have shown that the population sizing requirement for GAs grow with respect to the problem's size and difficulty. Thus, it is not rational to impose an upper bound on the size of the population. In other words, a *MaxPopsize* parameter should be eliminated from adaptive population sizing schemes. If that doesn't happen, then the user ends up having to guess a value for *MaxPopsize* (more or less in the same way that he has to guess a value for the population size of a traditional GA). In our opinion, there are only two reasons for setting an upper bound on the size of the population:

  1. when the maximum number of fitness function evaluations allowed for the whole simulation, T, is known in advance. In such case, the upper bound for the size of the population could be set to a value which is a function of T (and obviously less than T). As an extreme case, one generation of a population of size T would terminate the GA run, and that would correspond to random search.
  2. when there are memory constraints that don't allow our computers to run populations larger than a certain amount.

- **Test on problems with known population sizing requirements.** The different adaptive algorithms that we have reviewed were tested by their authors on different sets of problems. It is our strong belief that the test suit should include problem instances of varying size and difficulty and whose optimal population sizing requirements are well known. Moreover, the problems should have different population sizing requirements, some requiring small and others requiring large population sizes. A good self-adjusting population sizing algorithm should be able to detect both cases. We suggest that, among other problems, the class of additive decomposable problems, either with uniform and exponentially scaled building blocks, as well as problems with external sources of noise, should be included in the test suite.

- **Do fair comparisons.** There are several ways to make comparative studies on the performance of algorithms. Two of the most used methods are (1) given a fixed computational time T measure the solution quality Q, and (2) given a solution quality Q, measure the time T that the algorithm needs to reach that quality. Both components should be addressed on comparative studies. Otherwise, comparisons may be unfair.

- **Do scalability analysis.** It's important to make time and space complexity analyses for the proposed algorithms, even if they can only be done on a particular class of problems. The analysis should be complemented by computer experiments that verify how the algorithm scales up with problems of different size.

- **Make life easier for users.** All the adaptive population sizing algorithms that we have reviewed in this paper eliminate the need to specify a fixed population size for the GA run. Unfortunately, eliminating one parameter and introducing several other parameters is certainly not making life any easier from a user's perspective. If new parameters are introduced, either there should be recommendations on how to set them, or the adaptive algorithm has to be robust enough so that the performance is not much affected by these new parameters.

Acknowledgments

This work was sponsored by the Portuguese Foundation for Science and Technology (FCT/MCES), under grants POSI/SRI/42065/2001 and SFRH/BD/16890/2004.

5. REFERENCES

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS – a genetic algorithm with varying population size. In *Proc. of the First IEEE Conf. on Evolutionary Computation*, pages 73–78. IEEE Press, 1994.

[2] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An empirical study on GAs "without parameters". In *Parallel Problem Solving from Nature, PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2000.

[3] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor, 1975.

[4] A. E. Eiben, E. Marchiori, and V. A. Valkó. Evolutionary algorithms with on-the-fly population size adjustment. In *Parallel Problem Solving from Nature, PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2004.

[5] D. E. Goldberg. *The Design of Innovation - Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.

[6] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[7] D. E. Goldberg and M. Rudnick. Genetic algorithms and the variance of fitness. *Complex Systems*, 5(3):265–278, 1991.

[8] G. R. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[9] G. R. Harik and F. G. Lobo. A parameter-less genetic algorithm. In *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 258–267. Morgan Kaufmann, 1999.

[10] R. Hinterding, Z. Michalewicz, and T. C. Peachey. Self-adaptive genetic algorithm for numeric functions. In *Parallel Problem Solving from Nature, PPSN IV*, pages 420–429. Springer-Verlag, 1996.

[11] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[12] P. Larrañaga and J. A. Lozano, editors. *Estimation of distribution algorithms: a new tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston, MA, 2002.

[13] C. F. Lima and F. G. Lobo. Parameter-less optimization with the extended compact genetic algorithm and iterated local search. In *GECCO-2004: Proceedings of the Genetic and Evolutionary Computation Conference, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 1328–1339. Springer, 2004.

[14] F. G. Lobo. *The parameter-less genetic algorithm: Rational and automated parameter selection for simplified genetic algorithm operation*. Doctoral dissertation, Universidade Nova de Lisboa, Lisboa, 2000.

[15] F. G. Lobo, D. E. Goldberg, and M. Pelikan. Time complexity of genetic algorithms on exponentially scaled problems. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 151–158. Morgan Kaufmann, 2000.

[16] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Bayesian optimization algorithm, population sizing, and time to convergence. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 275–282. Morgan Kaufmann, 2000.

[17] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[18] M. Pelikan and T.-K. Lin. Parameter-less hierarchical BOA. In *GECCO-2004: Proceedings of the Genetic and Evolutionary Computation Conference, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2004.

[19] M. Pelikan and F. G. Lobo. Parameter-less genetic algorithm: a worst-case time and space complexity analysis. IlliGAL Report No. 99014, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, 1999.

[20] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2003.

[21] D. Schlierkamp-Voosen and H. Mühlenbein. Strategy adaption by competing subpopulations. In *Parallel Problem Solving from Nature, PPSN III*, pages 199–208. Springer-Verlag, 1994.

[22] D. Schlierkamp-Voosen and H. Mühlenbein. Adaptation of population sizes by competing subpopulations. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pages 330–335, 1996.

[23] R. E. Smith. Adaptively resizing populations: An algorithm and analysis. Technical Report No. 93001, The University of Alabama, Tuscaloosa, AL, 1993.

[24] R. E. Smith and E. Smuda. Adaptively resizing populations: Algorithm, analysis, and first results. *Complex Systems*, 9:47–72, 1995.

[25] D. Thierens, D. E. Goldberg, and A. Pereira. Domino convergence, drift, and the temporal-salience structure of problems. In *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pages 535–540. IEEE, 1998.

[26] V. A. Valkó. Self-calibrating Evolutionary Algorithms: Adaptive Population Size. Master's thesis, Free University Amsterdam, 2003.