GAVaPS — a Genetic Algorithm with Varying Population Size

Jarosław Arabas* Zbigniew Michalewicz[†] and Jan Mulawka [‡]

Abstract

The size of the population can be critical in many applications of genetic algorithms. If the population size is too small, the genetic algorithm may converge too quickly; if it is too large, the genetic algorithm may waste computational resources: the waiting time for an improvement might be too long. In this paper we propose an adaptive method for maintaining variable population size, which grows and shrinks together accordingly to some characteristic of the search. The first experimental results indicate some merits of the proposed method.

I. Introduction

The size of the population is one of the most important choices faced by any user of genetic algorithms (GAs) and may be critical in many applications. If the population size is too small, the genetic algorithm may converge too quickly; if it is too large, the genetic algorithm may waste computational resources: the waiting time for an improvement might be too long. Another decision faced by a user is the selection mechanism to be incorporated in the GA, since it influences the convergence of the algorithm. It seems that these two factors are strongly related; we discuss them briefly in turn.

Several researchers have been investigated the size of population for genetic algorithms from different perspectives. Grefenstette [13] applied a meta-GA to control parameters of another GA (including populations size and the selection method). Goldberg [10, 12] provides a theoretical analysis of the optimal population size. A study on influence of the control parameters on the genetic search (online performance for function optimization) is presented in [18]. Additional experiments with population size were reported in [15] and [5]. Recently Smith [20] proposed an algorithm which adjusts the population size with respect to the probability of selection error.

The selection step (selecting fit individuals from the population) of GAs enjoyed also a significant effort of several researchers. The first, and possibly the most recognized work, was due to DeJong [7] in 1975. He considered several variations of the simple selection. Further, in 1981 Brindle [4] considered some further modifications: which were superior over simple selection. In 1987 Baker [2] provided a comprehensive theoretical study of these modifications using some well defined measures, and also presented a new improved version (*stochastic universal sampling*), which uses a single wheel spin. Other methods to sample a population are based on introducing artificial weights: chromosomes are selected proportionally to their rank rather than actual evaluation values (see e.g., [21]). Also, a method where only few members of the population are changed (within each generation) was studied (*steady state* GAs [21]). In [1] the authors discuss different categories of selection procedures and classify them with respect to their properties.

It seems that there are two important issues in the evolution process of the genetic search: population diversity and selective pressure. These factors are strongly related: an increase in the selective pressure decreases the diversity of the population, and vice versa. In other words, strong selective pressure "supports" the premature convergence of the GA search; a weak selective pressure can make the search ineffective. Thus it is important to strike a balance between these two factors. Clearly, both factors are influenced by the size of population.

This paper discusses a Genetic Algorithm with Varying Population Size (GAVaPS). The new algorithm does not use any variation of selection mechanism considered earlier, but rather introduces the concept of "age" of a chromosome, which is equivalent to the number of generations the chromosome stays "alive". Thus the age of the chromosome replaces the concept of selection and, since it depends on the fitness of individual, influences the size of the population

*J. Arabas is with the Institute of Electronics Fundamentals, Warsaw University of Technology, Poland; e-mail: *jarabas@ipe.pw.edu.pl*. His work was supported by the Polish State Committee for Scientific Research under Grant 8S50301905

[†]Z. Michalewicz is with the Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA; e-mail: *zbyszek@mosaic.uncc.edu*.

[‡]J. Mulawka is with the Institute of Electronics Fundamentals, Warsaw University of Technology, Poland; e-mail: *jml@ipe.pw.edu.pl*. His work was supported by the Polish State Committee for Scientific Research under Grant 8S50301905

at every stage of the process. It seems also that such approach is more "natural" than any selection mechanism considered earlier: after all, the aging process is well-known in all natural environments.

Additional motivation for this work was based on the following observation: a few researchers examined a possibility of introducing adaptive probabilities of genetic operators in genetic algorithms [6], [8], [18]; other techniques, like evolutionary strategies [19] already incorporated adaptive probabilities for its operators some time ago. It seems reasonable to assume that at different stages of the evolution process different operators would have different significance and the system should be allowed to self-tune their frequencies and scope. The same should be true for population sizes: at different stages of the evolution process different sizes of the population may be 'optimal', thus it is important to experiment with some heuristic rules to tune the size of the population to the current stage of the search.

The paper is organized as follows. The next section presents the proposed algorithm. Section III discusses results of a few experiments, and some simulation results are provided and compared to the classical GA formulation. The last section concludes and provides hints on further research.

II. The GAVaPS Algorithm

We describe the algorithm to solve maximization problems where the fitness function takes non-negative values. These assumptions are not critical and were made only to simplify the presentation. If fitness function with negative values is considered, it can be easily transformed into non-negative one [11]. Also, a minimization problem can be reformulated as a maximization problem by multiplying the function by -1 and by further transformation to non-negative valued function, if necessary.

The GAVaPS algorithm at time t processes a population $P(t)$ of chromosomes. During the 'recombine $P(t)$' step, a new auxiliary population is created (this is a population of offspring). The size of the auxiliary population is proportional to the size of the original population; the auxiliary population contains $AuxPopSize(t) = \lfloor PopSize(t) * \rho \rfloor$ chromosomes (we refer to parameter ρ as a *reproduction ratio*). Each chromosome from the population can be chosen to reproduce (i.e., to place the offspring in the auxiliary population) with equal probability, *independently* of its fitness value. Offspring are created by applying genetic operators (crossover and mutation) to selected chromosomes. Since the selection of the chromosomes does not depend on their fitness values, i.e., there is no selection step as such, we introduce the concept of *age* of the chromosome and its *lifetime* parameter.

The structure of the GAVaPS is shown in Figure 1.

```
procedure GAVaPS
begin
  $t = 0$
  initialize $P(t)$
  evaluate $P(t)$
  while (not termination-condition) do
  begin
    $t = t + 1$
    increase the age of each individual by 1
    recombine $P(t)$
    evaluate $P(t)$
    remove from $P(t)$ all individuals
      with age greater than their lifetime
  end
end
```

Figure 1: The GAVaPS algorithm

The lifetime parameter is assigned once for each chromosome during the evaluation step (either after the initialization for all chromosomes or after the recombination step for members of auxiliary population) and remains constant (for a given chromosome) through the evolution process, i.e., from the birth of the chromosome to its death. It means that for the 'old' chromosomes their lifetime values *are not* re-calculated. The death of a chromosome occurs when its age, i.e., the number of generations the chromosome stays alive (initially set to zero), exceeds its lifetime value. In other words, chromosome's lifetime determines the number of GAVaPS generations during which the chromosome is kept in the population: after expiring its lifetime, the chromosome dies off. Thus the size of the population after single iteration is

$$PopSize(t+1) = PopSize(t) + AuxPopSize(t) - D(t),$$

where $D(t)$ is the number of chromosomes which die off during generation t.

There are many possible strategies of assigning lifetime values. Clearly, assigning a constant value (greater than one) independently of any statistics of the search would cause an exponential growth of the population size. Moreover, since there is no selection mechanism as such in the GAVaPS, no selective pressure exists, so assigning a constant value for the lifetime parameter would result in a poor performance of the algorithm. In order to introduce a selective pressure, a more sophisticated lifetime calculation should be performed. The lifetime calculation strategies should (1) reinforce the individuals with above-average fitness, (and consequently, restrict the individuals with below-average fitness), and (2) tune the size of the population to the current stage of the search (in particular, prevent the exponential growth of the population and lower simulation costs). Reinforcement of fit individuals should result in above-average allocation of their offspring in the auxiliary populations. Since there is an equal probability for each individual to undergo the genetic recombination, the expected number of the individual's offspring is proportional to its lifetime value (since the lifetime determines number of generations of keeping the individual in the population). So individuals having above-average fitness values should be granted higher lifetime values. While calculating the lifetime, a state of the genetic search should be taken under consideration. Because of that we use a few measures of the state of the search: $AvgFit$, $MaxFit$ and $MinFit$ represent average, maximal and minimal fitness values, respectively, in the current population, and $AbsFitMax$ and $AbsFitMin$ stand for maximal and minimal fitness values found so far. It should be also noted that the lifetime calculation should be computationally easy in order to spare the computational resources.

Having in mind the above remarks, several lifetime calculation strategies have been implemented and used for the experiments. The lifetime parameter for the i-th individual ($lifetime[i]$) can be determined by:

(1) proportional allocation:

$$\min(MinLT + \eta\frac{fitness[i]}{AvgFit}, MaxLT)$$

(2) linear allocation:

$$MinLT + 2\eta\frac{fitness[i] - AbsFitMin}{AbsFitMax - AbsFitMin}$$

(3) bi-linear allocation:

$$\begin{cases} MinLT + \eta\frac{fitness[i]-MinFit}{AvgFit-MinFit} \\ \quad if\ \ AvgFit \geq fitness[i] \\ \frac{1}{2}(MinLT + MaxLT) + \eta\frac{fitness[i]-AvgFit}{MaxFit-AvgFit} \\ \quad if\ \ AvgFit < fitness[i] \end{cases}$$

where $MaxLT$ and $MinLT$ stand for maximal and minimal allowable lifetime values, respectively (these values are given as the GAVaPS parameters), and $\eta = \frac{1}{2}(MaxLT - MinLT)$.

The first strategy (proportional allocation) has come up from the idea of roulette-wheel selection: the value of lifetime for particular individual is proportional to its fitness (within limits $MinLT$ and $MaxLT$). However, this strategy has a serious drawback — it does not utilize any information about the "objective goodness" of the individual, which can be estimated by relating its fitness to the best value found so far. This observation motivates the linear strategy. In this strategy the lifetime value is calculated accordingly to the individual fitness related to the best value at present. However, if many individuals have their fitness equal or approximately equal to the best value, such strategy results in allocating long lifetime values, thus enlarging the size of the population. Finally, the bi-linear strategy attempts to make a compromise between the first two. It sharpens the difference between lifetime values of nearly-the-best individuals utilizing information about the average fitness value, however also taking into consideration the maximal and minimal fitness values found so far.

III. Experiments and Results

The GAVaPS algorithm was tested on the following functions:

G1:	$-x\sin(10\pi x) + 1$	$-2.0 \leq x \leq 1.0$
G2:	$\mathsf{integer}(8x)/8$	$0.0 \leq x \leq 1.0$
G3:	$x \cdot \mathsf{sgn}(x)$	$-1.0 \leq x \leq 2.0$
G4:	$0.5 + \frac{\sin^2\sqrt{x^2+y^2}-0.5}{(1+0.001(x^2+y^2))^2}$	$-100 \leq x, y \leq 100$

The functions were chosen to cover the wide spectrum of possible function types to be optimized. Functions G1 and G4 are multimodal functions with many local maxima. Function G2 cannot be optimized by means of any gradient technique, since there is no gradient information available. Function G3 represents a problem recognized as a "deceptive problem" [11]. While maximizing such function, two directions of growth can easily be recognized, but the boundaries are chosen in such way that only for one of them a global maximum can be obtained. In case of gradient-based techniques with random sampling this should result in frequent finding the local maximum.

GAVaPS performance has been tested and compared to the performance of the Goldberg's Simple Genetic Algorithm (SGA) [11]. Problem coding methods as well as genetic operators were identical for the SGA and GAVaPS (a simple binary coding has been used and two genetic operators: mutation and one point crossover).

For the experiments we have made the following assumptions. The initial size of any population was 20. In case of the SGA, the size of initial population remained constant through the entire simulation. Reproduction ratio ρ was set to 0.4 (this parameter is meaningless in case of the SGA). Mutation ratio was set to 0.015, and crossover ratio was set to 0.65. The length of chromosomes was 20. Through all our experiments we assumed that minimal and maximal lifetime values were constant and equal to $MaxLT = 7$ and $MinLT = 1$.

To compare SGA with GAVaPS, two parameters have been chosen: cost of the algorithm, represented by *evalnum* (the average of the number of function evaluations over all runs) and performance, represented by *avgmax* (the average of the maximal values found over all runs). Both algorithms have the same termination condition: they terminate if there is no progress in terms of the best value found for consecutive $conv = 20$ generations. Population was initialized at random, and there were 20 independent runs performed. Then, measures of performance and cost were averaged over these 20 runs giving the reported results. While testing the influence of a single parameter on the performance and the cost, the values of the parameters reported above were constant except the one which influence was tested.

Figure 2 shows the $PopSize(t)$ and the average fitness of the population for a single GAVaPS run for the function G4 with the bi-linear lifetime calculation (similar observations can be made for other functions and other strategies for allocating lifetime values). The shape of the $PopSize(t)$ curve seems very interesting. At first, when the fitness variation is relatively high, the population size grows. This means, that the GAVaPS makes a wide search for the optima. Once the neighborhood of the optimum is located, the algorithm starts to converge and the population size is reduced. However, there is still a search for an improvement. When a possibility for a better result occurs, another "demographic explosion" takes place, which is followed by another convergence stage. It seems that the GAVaPS incorporates a *self-tuning* process by choosing the population size at each stage of the evolution process.

Figures 3–4 show the influence of the reproduction ratio on the performance of the GAVaPS. For the SGA, this value has no meaning (since in this case there is a total overlap of the old population by the new one). In case of the GAVaPS, this value strongly influences the simulation cost, which can be decreased by lowering the reproduction ratio, however without loss of accuracy (see relevant values of *avgmax*). Judging from the experiments, it seems that the 'optimal' selection of ρ is approximately 0.4.

Figures 5–6 show the influence of the initial population size on the performance (*avgmax*) and computation cost (*evalnum*) of the algorithms. In the case of the SGA, the population size (for all runs) was constant and equal to its initial value. As expected, for the SGA low values of population size implied low cost and poor performance. Increasing population size at first improves the performance but also increases cost of computations. Then there is a stage of "performance saturation", while cost is still linearly growing. In case of the GAVaPS, the initial population size has in practice no influence on both performance (very good) and cost (reasonable and sufficient for the very good performance). Similar observations can be made by analysing the cost and the performance of both algorithms on remaining functions G1–G3. However, it is important to note that the SGA has the optimal behavior (best performance with minimum cost) for different values of the population size for all four problems. On the other hand, GAVaPS adopts the population size to the problem at hand and the state of the search.

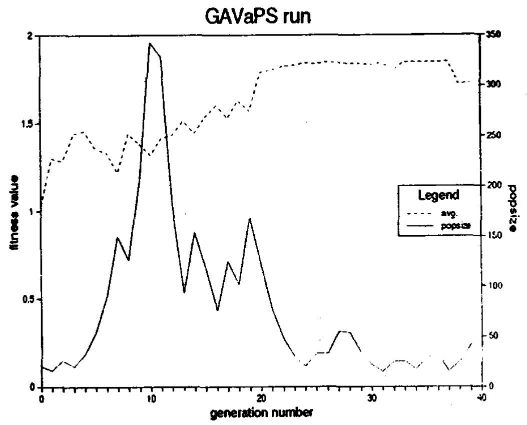

Figure 2: *PopSize(t)* and average fitness of the population for a single GAVaPS run

In the table below we report on performance and simulation cost obtained from the experiments with all testbed functions G1–G4. The rows 'SGA', 'GAVaPS (1)', 'GAVaPS (2)', and 'GAVaPS (3)' contain the best value found (V) and the number of function evaluations (E) for the SGA and GAVaPS with proportional, linear, and bi-linear lifetime allocations, respectively. The optimal population sizes for the SGA for test cases G1 – G4 were 75, 15, 75, 100, respectively.

Type of the algorithm	Function							
	G1		G2		G3		G4	
	V	E	V	E	V	E	V	E
SGA	2.814	1467	0.875	345	1.996	1420	0.959	2186
GAVaPS(1)	2.831	1708	0.875	970	1.999	1682	0.969	2133
GAVaPS(2)	2.841	3040	0.875	1450	1.999	2813	0.970	3739
GAVaPS(3)	2.813	1538	0.875	670	1.999	1555	0.972	2106

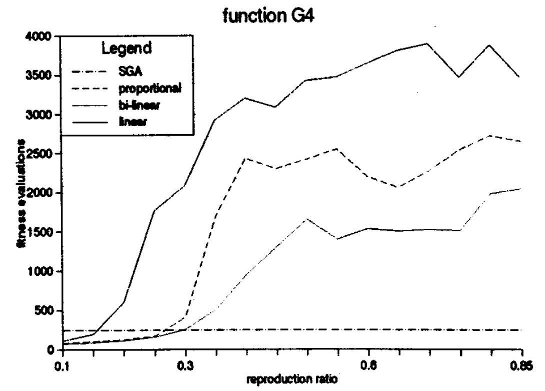

Figure 3: Comparison of the SGA and GAVaPS: reproduction ratio versus number of evaluations

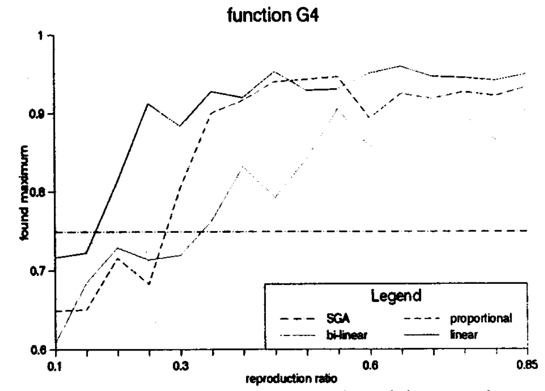

Figure 4: Comparison of the SGA and GAVaPS: reproduction ratio versus average performance

The linear strategy (2) is characterized by the best performance and (unfortunately) the highest cost. On the other hand, the bi-linear strategy (3) is the cheapest one, but the performance is not as good as in the linear case. Finally, the proportional strategy (1) provides the medium performance with the medium cost. It should be noted, that the GAVaPS algorithm with any lifetime allocation strategy (1)-(3) in most test-cases provides better performance than the SGA. The cost of the GAVaPS (in comparison to the SGA) is higher, however, the results of the SGA were reported for the optimal sizes of populations. If, for example, the population size for the SGA in the experiment with function G2 was 75 (instead of the optimal 15), then the SGA simulation cost would be 1035.

IV. Conclusions and Further Research

The knowledge about the proper selection of GA parameters is still only fragmentary and has rather empirical background. Among these parameters, the population size seems to be the most important, since it has strong influence on the GA simulation cost. It might be that the best way for its setting is to let it to self-tune accordingly to the GA actual needs. In this paper it has been shown that such a method may find an approximately minimal cost of the GA run sufficient to solve the problem.

Still, there are several problems left open. The major one is the choice of the optimal strategy of lifetime calculation and it deserves a deeper investigation. Also, some mathematical foundations of GAVaPS should be searched for; some demographic theories could be possibly helpful to approach these problems.

References

[1] Bäck, T., and Hoffmeister, F., "Extended Selection Mechanisms in Genetic Algorithms", in [3], pp.92–99.

[2] Baker, J.E., "Reducing Bias and Inefficiency in the Selection Algorithm", in [14], pp.14–21.

[3] Belew, R. and Booker, L. (Editors), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1991.

[4] Brindle, A., "Genetic Algorithms for Function Optimization", Doctoral Dissertation, University of Alberta, Edmonton, 1981.

[5] Cartwright, H.M., and Mott, G.F., "Looking Around: Using Clues from the Data Space to Guide Genetic Algorithm Searches", in [3], pp.108–114.

[6] Davis, L., "Adapting Operator Probabilities in Genetic Algorithms", in [17], pp.61–69.

[7] De Jong, K.A., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", (Doctoral dissertation, University of Michigan), *Dissertation Abstract International*, 36(10), 5140B. (University Microfilms No 76-9381).

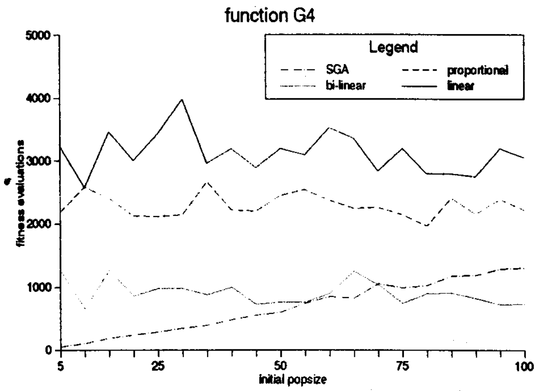

Figure 5: Comparison of the SGA and GAVaPS: initial population size versus number of evaluations

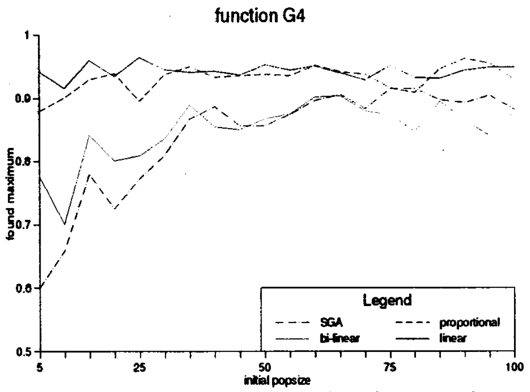

Figure 6: Comparison of the SGA and GAVaPS: initial population size versus average performance

[8] Fogarty, T.C., "Varying the Probability of Mutation in the Genetic Algorithm", in [17], pp.104–109.

[9] Forrest, S. (Editor), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1993.

[10] Goldberg, D.E., "Optimal Initial Population Size for Binary-Coded Genetic Algorithms", TCGA Report No.85001, Tuscaloosa, University of Alabama, 1985.

[11] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.

[12] Goldberg, D.E., "Sizing Populations for Serial and Parallel Genetic Algorithms", in [17], pp.70–79.

[13] Grefenstette, J.J., "Optimization of Control Parameters for Genetic Algorithms", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 16, No.1, pp.122–128, 1986.

[14] Grefenstette, J.J., (Editor), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[15] Jog, P., Suh, J.Y., Gucht, D.V., "The Effects of Population Size, Heuristic Crossover, and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem", in [17], pp.110–115.

[16] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, 1992.

[17] Schaffer, J., (Editor), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1989.

[18] Schaffer, J., Caruana, R., Eshelman, L., and Das, R., "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", in [17], pp.51–60.

[19] Schwefel, H.-P., "Evolution Strategies: A Family of Non-Linear Optimization Techniques Based on Imitating Some Principles of Organic Evolution", Annals of Operations Research, Vol.1, pp.165–167, 1984.

[20] Smith, R.E., "Adaptively Resizing Populations: An Algorithm and Analysis", in [9], pp.653.

[21] Whitley, D., "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best", in [17], pp.116–121.