Improving Genetic Algorithms Performance via Deterministic Population Shrinkage

Juan Luis J. Laredo
Department of Architecture and Computer Technology
University of Granada. ETSIT. Spain.
C/ Periodista Daniel Saucedo
18071 Granada (Spain)
juanlu@geneura.ugr.es

Carlos Fernandes
Department of Architecture and Computer Technology
University of Granada. ETSIT. Spain.
LASEEB-ISR/IST. University of Lisbon, Portugal
cfernandes@laseeb.org

Juan Julián Merelo
Department of Architecture and Computer Technology
University of Granada. ETSIT. Spain
C/ Periodista Daniel Saucedo
18071 Granada (Spain)
jmerelo@geneura.ugr.es

Christian Gagné
Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique, Université Laval
Québec (Québec), Canada G1V 0A6
christian.gagne@gel.ulaval.ca

ABSTRACT

Despite the intuition that the same population size is not needed throughout the run of an Evolutionary Algorithm (EA), most EAs use a fixed population size. This paper presents an empirical study on the possible benefits of a Simple Variable Population Sizing (SVPS) scheme on the performance of Genetic Algorithms (GAs). It consists in decreasing the population for a GA run following a predetermined schedule, configured by a speed and a severity parameter. The method uses as initial population size an estimation of the minimum size needed to supply enough building blocks, using a fixed-size selectorecombinative GA converging within some confidence interval toward good solutions for a particular problem. Following this methodology, a scalability analysis is conducted on deceptive, quasi-deceptive, and non-deceptive trap functions in order to assess whether SVPS-GA improves performances compared to a fixed-size GA under different problem instances and difficulty levels. Results show several combinations of speed-severity where SVPS-GA preserves the solution quality while improving performances, by reducing the number of evaluations needed for success.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search, Heuristic Methods*; G.1.6 [**Mathematics of Computing**]: Numerical Analysis—*Optimization*

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

General Terms

Algorithms

Keywords

Adaptation, Self-adaptation, Genetic Algorithms, Parameter Tuning, Performance Analysis

1. INTRODUCTION

Setting an adequate population size is a key to obtain good performances in a Genetic Algorithm (GA), that is, to preserve a good quality in the solutions without spending extra computational efforts. That way, a small problem instance will require a smaller population size than a larger instance of a more difficult problem [13, 14]. Additionally, improving the GA performance is also possible by varying the population size during the GA run, adapting it as the algorithm is converging toward some solutions [5, 16].

Population sizing theory, based on Goldberg's facetwise decomposition for designing competent GAs [12], focuses on determining population sizing according to problem difficulty. Despite the issue that has been pointed out in Lobo and Lima's review of adaptive population sizing schemes [19], such a theory has not received much attention with respect to variable population sizing schemes. There are just a few studies that have taken it into account, with as prime example population sizing through estimation of schema variances by Smith and Smuda [21], with selection errors probability and adjustment made according to an expected selection loss provided by the user.

Therefore, the aim of this paper is to provide a general framework for evaluating performance of variable population sizing schemes from the population sizing theory perspective. For that purpose, a method based on bisection [20] is used to estimate the minimum size of the initial population P_{init} required to supply enough building blocks (BBs) so that a fixed-size selectorecombinative GA[1] will converge

[1]The assumption of a selectorecombinative GA (without

toward optimal solutions. That population size is then used for bootstrapping another selectorecombinative GA, which is using for the tested variable population sizing scheme. This allows assessment of the solutions quality and computational effort required. Given that both fixed and variable sizing schemes are provided with the same initial supply of BBs, some conclusions can be drawn from the difference of performances. Additionally, a scalability analysis is performed on different instances and complexities of decomposable trap functions [1]. Hence, it is possible not only to analyze a given performance, but also, to analyze how it is affected by problem size and complexity.

We propose to test with a simple variable population sizing scheme (SVPS). The SVPS is a deterministic function that monotonically reduces the population size during the GA run. The function slope can be controlled by setting two parameters, one for controlling the shrinking speed and the other for controlling the severity of population resizing. The intuition justifying the approach is that, at an early stage of a GA run, larger population sizes produce a wider exploration of the search landscape. Meanwhile, smaller population sizes increase the exploitation of promising regions as the GA converges [19].

Nevertheless, our intent in the current work is to address the following points:

1. Provide a general framework to test different variable population sizing schemes from the population sizing theory perspective. Following the same steps, SVPS could be replaced by new or suitable schemes in the literature.

2. Check whether variable population sizing schemes in GAs are able to outperform an equivalent parameterized GA using a fixed size scheme. If the SVPS-GA improves the fixed-size GA performance, that would mean that there are better schemes for population sizing than the standard fixed size scheme.

3. Gain some insight into the dynamics of the SVPS scheme. The idea of SVPS is to maintain a good quality in the solutions using less evaluations. Nevertheless, the seamless run of a GA makes it difficult to estimate the desired population size decreasing. Additionally, a given strategy could be more or less adequate to different problem instances or problem complexities.

The rest of the paper is organized as follows: Section 2 presents the state of the art in variable population sizing schemes. The proposed methodology is described in Section 3 and the experimental setup in Section 4. Section 5 explores the dynamics of the SVPS, showing that a variable population sizing scheme can yield a better performance than a fixed population sizing scheme. Finally, results are discussed in Section 6.

2. RELATED WORKS

This section provides a brief description of some other varying population methods previously proposed in the Evolutionary Computation research field. Following the classification of parameter control mechanisms given by Eiben *et al.* [6], we may say that some of the techniques described below fall into the adaptive methods categories (GAVaPS, for instance), while others, like RVPS [4] and PRoFIGA [7] are deterministic methods. Our proposal may be classified as deterministic, because the population size in each generation is defined in the beginning of the search by two parameters; the size is always forced to decrease, with more or less speed, and ends with more or less individuals, depending on those two parameters. Other methods follow a different policy and adapt the size during the run according to the state of the search. One of those algorithms, proposed in 1994 by Arabas *et al.* [2], is the Genetic Algorithm with Varying Population Size (GAVaPS).

GAVaPS does not hold an explicit selection mechanism. As in natural systems, population size is defined by the birth and death of individuals occurring at each iteration. A parameter called lifetime is introduced. It defines the number of generations in which each individual is allowed to remain alive. After its creation, the chromosome is assigned to a specific lifetime, according to its fitness. Three lifetime calculation methods are proposed. The algorithm proceeds in a generational manner, at each time step increasing each individual's age. When an individual's age exceeds its lifetime, the chromosome is removed from the population. Since fittest individuals remain in the population for more generations, thus having a higher probability to be engaged in a reproduction process and generate offspring, GAVaPS' chromosomes have equal probability to be selected to reproduce, independently of their fitness value. This concept of lifetime/age provides the algorithm with the necessary selection pressure, which reduces the need for selection strategies: GAVaPS randomly pairs the chromosomes for crossover operations. The intensity of the pressure is controlled by two parameters, $minLT$ and $maxLT$, that define, respectively, the minimum and maximum lifetime allowed for each chromosome. Higher difference between the two values leads to a more selective algorithm. However, this process may have a serious drawback since increasing the $maxLT$ parameter will result in larger populations and, as stated above, an increasingly high population size is a characteristic of GAVaPS. The algorithm also introduces another parameter: reproduction rate (ρ); its value defines the number of new chromosomes created in each generation t, depending on the size of the current population. GAVaPS was tested for the studies presented in [11] and [7] and the results lead to the conclusion that GAVaPS is extremely sensitive to the reproduction rate, and very often the population grows exponentially or becomes extinct.

A similar approach was attempted with The Adaptive Population size Genetic Algorithm (APGA) [3]. The only difference between this algorithm and GAVaPS resides in reproduction rate, which in APGA has a fixed value of two individuals. This technique follows the reproduction strategy of the Steady-State GA and prevents the population from growing out of control as it often happens with GAVaPS. On the other hand, such a low reproduction rate results in populations with few individuals unless a high value for $maxLT$ is used. But, even in the last case, the population size is very stable and apparently does not react to the evolution process and different search stages [3]. The algorithm appears to perform well on some problems and clearly outperformed GAVaPS when applied to the Spears' multi-modal problems [7]. However, Lobo and Lima [18] questioned the results in [7] and proved that there is an upper bound equal

mutation) is made as the only source of diversity is then the initial supply of BBs.

to $2maxLT + 1$ for APGA's population size, after $maxLT$ generations.

Eiben, Marchiori and Valkó proposed in [7] the Population Resizing on Fitness Improvement GA (PRoFIGA). The variation process of PRoFIGA is based on the improvement of the best fitness in the population. The process intends to balance exploration and exploitation by growing the population in earlier and exploratory stages and gradually decreasing it in later stages of the search. When the population becomes trapped in local optima, the process is supposed to generate another growing phase of the population, thus increasing diversity and escaping the local optima. The authors present a heuristic for size variation during the run that increases or decreases the population size according to whether or not the best fitness of the population has been improved and, if the later case is observed, for how long it has remained unchanged. The main drawback of this algorithm are its extra six parameters, which makes it very hard to tune for a non expert user.

In the Random Variation of Population Size GA (RVPS) [4] the population size is randomly changed during the run. The authors concluded that in some cases the performance of RVPS is equivalent to the standard GA. So, when there are no hints about the optimal population size for some problem, it may be appropriate to randomly set and vary the population size of the GA.

Like PRoFIGA and RVPS, the Saw-Tooth Genetic Algorithm [15] is an example of a deterministic method used in the variation of the population size. In this algorithm the population size varies according to a predefined function with a saw-tooth shape. The authors concluded that the Saw-Tooth GA performed well on some particular test functions. However, besides a variable population size, the Saw-Tooth GA also uses a re-initialization mechanism to introduce genetic diversity in the population.

The Self-Regulated Population Size EA (SRP-EA), proposed by Fernandes and Rosa in [11], follows GAVaPS guidelines but it manages to control the population size, thus avoiding the typical extinction and demographic burst observed in the dynamics of Arabas' algorithm. SRP-EA self-controls the population size (indirectly) via genetic diversity. There is a threshold value that adapts during the run and that defines the Hamming distance value above which two chromosomes are allowed to crossover and generate offspring and like GAVaPS the individuals are provided with a lifetime that defines the of generations that they are allowed to remain in the population. The algorithm was compared with APGA and CHC [10] and outperformed both in the proposed test set.

Finally, there could be other reasons to use variable population sizing schemes. Laredo *et al.* expose in [17] the case of a fully distributed EA in which the individuals have to decide on their own state of reproduction without any central control, using instead estimations about the global population state for decision making. The population size varies at run-time as a consequence of such a decentralized reproduction and a self-adjusting mechanism based on autonomous selection [8] tries to keep it stable.

Our strategy, described in the next section, does not aim at adapting the population size or varying it deterministically according to the state of the search. Alternatively, and although it may be classified as a deterministic scheme, together with PROFIGA and the Saw-Tooth GA, SVPS tries to explore the premise that states that larger populations are needed in the beginning of the search, while towards the end the GA can manage to converge with a smaller population [19].

3. METHODOLOGY

The methodology followed takes into account Goldberg's facetwise decomposition for designing competent GAs [12] in order to answer whether a GA using varying population size improves a fixed-size GA.

The proposed method consists in the following three steps:

1. Bisection method, to estimate size n' of population P'_{init};
2. Refinement of size of the population size, to obtain the necessary supply of BBs;
3. Simple variation of the population size using a predetermined schedule.

The two first steps are used to estimate the minimum initial population size ($P_{init} = \{ind_1, ind_2, \ldots, ind_n\}$) required to supply enough BBs for a reliable convergence to the problem optimum in a fixed-size population GA.

In the third step, P_{init} is used as the initial population of the SVPS-GA in which the population decreases according to a parameterized speed (τ) and severity (ρ). Any combination τ-ρ has to preserve the quality of solutions while improving the number of evaluations.

Steps 1, 2, and 3 are exposed in sections 3.1, 3.2 and 3.3, respectively. Note that the SVPS-GA (step 3) could be changed for another variable, adaptive or self-adaptive population sizing scheme. Therefore, this methodology provides a framework to test different population resizing schemes.

3.1 Bisection Method

The bisection method [20] estimates the optimal population size n' to solve a problem instance, that is, the lowest n' for which 98% of the runs find the problem optimum. A fixed-size selectorecombinative GA is used to search the minimum population size required using random initialization, to provide enough BBs to converge to the optimum using only recombination and selection mechanisms.

Algorithm 1 depicts the method based on bisection. The method begins with a small population size which is doubled until the algorithm ensures a reliable convergence. We define the reliability criterion as the convergence of the algorithm to the optimum 49 out of 50 times (0.98 of Success Rate). After that, the interval (min, max) is halved several times and the population size adjusted within such a range until $\frac{max-min}{min} > threshold$, where min and max stand respectively for the minimum and maximum population size estimated and *threshold* for the accuracy of the adjustment within such a range. This parameter has been set to $\frac{1}{16}$ in order to obtain a good adjustment of the initial population size.

3.2 Refining Initial Supply of Building Blocks

The initial supply of BBs given by a population P'_{init} might be a bit oversized because the minimum population size n' is estimated stochastically. Such a precision is usually accurate enough for scalability analysis, but in the case of study, additional BBs will induce smaller values for τ and ρ, leading to a slightly unfair comparison.

Algorithm 1 Method based on bisection

$n' =$ initial population size
while reliability of P'_{init} with size $n' < 98\%$ **do**
  $min = n'; max = 2n'; n' = 2n'$
  P'_{init} size $\leftarrow n'$
end while
while $\frac{max-min}{min} > \frac{1}{16}$ **do**
  $n' = \frac{max+min}{2}$
  P'_{init} size $\leftarrow n'$
  if reliability of P'_{init} with size $n' < 98\%$ **then**
    $min = n'$
  else
    $max = n'$
  end if
end while

Algorithm 2 Refinement of P'_{init}

$P_{init} \leftarrow P'_{init}$
while reliability of P_{init} with size $n \geq 98\%$ **do**
  $P_{init} \leftarrow$ randomly removes 1% of individuals in P_{init}
end while

Therefore, Algorithm 2 shows an iterative process to refine P'_{init} into P_{init} by randomly subtracting 1% of the individuals until P_{init} is minimized.

3.3 Varying Population Size

The SVPS-GA uses the following deterministic function to vary its size:

$$n_g = \begin{cases} n_0 \times \left(1 - (1-\rho)\left(\frac{g}{g_{max}}\right)^{\tau}\right), & g \leq g_{max} \\ n_{g_{max}}, & g > g_{max} \end{cases} \quad (1)$$

In this equation, n_g stands for the population size at generation g, n_0 the initial population size, and $n_{g_{max}}$ the population size when a maximum number of generations (g_{max}) of the schedule is reached. To scale the shape of the function into the SVPS-GA runtime, g_{max} is estimated on the necessary runtime of the fixed-size GA. τ and ρ are respectively the speed and severity parameters. As shown in Figure 1, the values of τ and ρ influence the shape of the population sizing schedule. A smaller τ leads to a faster reduction of the population size. ρ belongs to the interval $]0, 1]$, where a value of $\rho \rightarrow 0$ means that the run ends with a nearly empty population, and where a value of $\rho = 1$ does not modify the initial population size.

Algorithm 3 shows the procedure for iterating on different values of τ and ρ. Only those results which guarantee a success rate of 0.98 are stored (e.g. a very small τ and ρ could lead to an unreliable convergence of the SVPS-GA). Lower bounds for τ and ρ have been fixed to pessimistic values

Algorithm 3 Varying the population with τ-ρ

for $\tau = 0.125, ..._{*1.5}, 32$ **do**
  for $\rho = 0.25, ..._{+0.5}, 1$ **do**
    if SVPS-GA reliability (P_{init},τ,ρ) $\geq 98\%$ **then**
      store the combination τ-ρ
    end if
  end for
end for

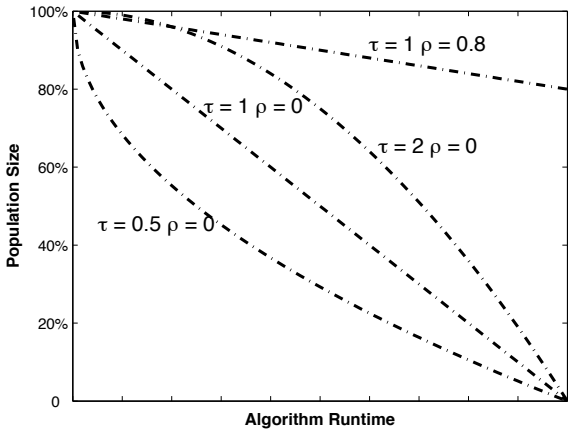

Figure 1: SVPS shapes for different values of τ-ρ.

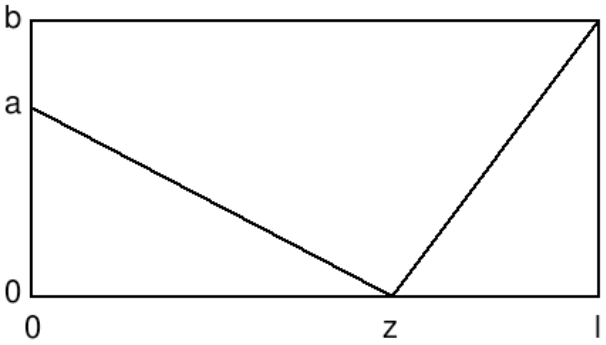

Figure 2: Generalized *l-trap* function.

that will not meet the reliability condition (e.g. $\tau = 0.125$ means that the population size will converge too quickly to ρ), while upper bounds stand for equivalent sizes to the fixed-size population (e.g. $\rho = 1$ means that the population size does not shrink, or $\tau = 32$ that the population shrinks in the last few generations).

4. EXPERIMENTAL SETUP

Following Lobo and Lima's recommendations [19] on the selection of a test suite with known population requirements and investigating the scalability on landscapes of different characteristics, experiments were conducted on trap functions [1]. A trap function is a piecewise-linear function defined on unitation (the number of one values in a binary string). There are two distinct regions in the search space, one leading to a global optimum and the other leading to the local optimum (see Figure 2). In general, a trap function is defined by the following equation:

$$trap(u(\overrightarrow{x})) = \begin{cases} \frac{a}{z}(z - u(\overrightarrow{x})), & \text{if } u(\overrightarrow{x}) \in [0, z] \\ \frac{b}{l-z}(u(\overrightarrow{x}) - z), & \text{if } u(\overrightarrow{x}) \in]z, l] \end{cases} \quad (2)$$

where $u(\overrightarrow{x})$ is the unitation function returning the number of one values in bit string $\overrightarrow{x}$, a is the local optimum, b is the global optimum, l is the problem size and z is a slope-change location separating the attraction basin of the two optima.

Trap instances	
BB size (l)	$2, 3, 4$
Number of sub-functions (m)	$2, 4, 8, 16, 32, 64$
GA Setup	
GA	selectorecombinative GA selectorecombinative SVPS-GA
Population size	Bisection + Refinement
Selection of Parents	Binary Tournament
Recombination	One-point crossover, $p_c = 1.0$
Individual Length	$l \times m$
SVPS Setup	
Speed (τ)	$0.125, ..._{*1.5}, 32$
Severity (ρ)	$0.25, ..._{+0.05}, 1$

Table 1: Parameters of the experiments

For the following experiments, 2-trap, 3-trap and 4-trap functions were designed with the following parameter values: $a = l - 1$, $b = l$, and $z = l - 1$. With these settings, 2-trap is not deceptive, 4-trap is deceptive and 3-trap lies in the region between deception and non-deception. Under these conditions, it is possible not only to examine the scalability on trap functions, but also to investigate how the scalability varies when changing from non-deceptive to deceptive search landscapes. Scalability tests were performed by juxtaposing m trap functions and summing the fitness of each sub-function to obtain the total fitness.

All settings are summarized in Table 1, operators as binary tournament or one-point crossover are standard in GAs [9]. The baseline for comparison is a generational selectorecombinative GA. Since the methodology imposes a 98% success rate in the results, the Average Evaluations to Solution (AES) has been used as an appropriate metric to measure the computational effort to reach the success criterion. A more efficient algorithm requires a smaller number of evaluations.

5. RESULTS

From the graphics in Figure 4 it can be concluded that SVPS-GA scales better than the fixed-size GA. This fact proves that variable population sizing schemes can improve the performance of GAs. Student's t-test conducted on the results in Table 2 shows that, except for the smaller instances (i.e. some combinations for $\tau - \rho$ in $m = 2, 4$ and 8), improvements are statistically significant.

Such improvements are more remarkable when the GA requires large population sizes which is directly related with the problem size and difficulty. As a general pattern, the larger the number of sub-functions (m) or the more difficult the problem, the larger the initial population is. Hence, SVPS performs better under large instances of difficult problems.

The relationship between the initial population size and the improvement in the number of evaluations is depicted in Figure 3. It shows the saved evaluations by the SVPS-GA with respect to the fixed-size GA. The improvement keeps a proportionality of order $\Theta(n^{1.54})$ to the initial population size. However, it is our belief that other variable population sizing schemes could outperform this mark since we have just contemplated the shrinkage of the population.

In order to gain some insight into the way the population shrinks, Figure 5 shows the set of strategies (i.e. combinations $\tau-\rho$) in which the SVPS-GA outperforms the fixed-size GA. They present the following behavior:

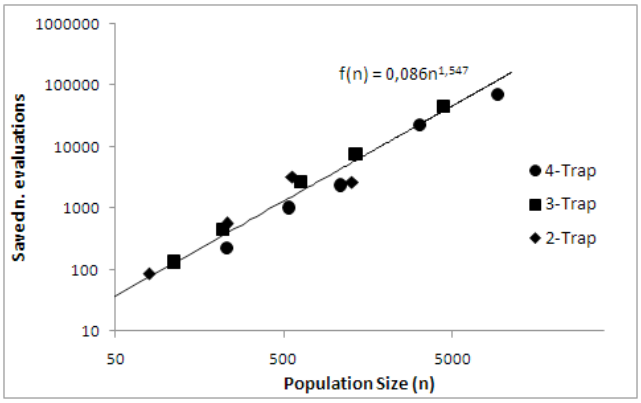

Figure 3: Improvement in the number of evaluations of the SVPS-GA with respect to the fixed-size GA. Results are depicted as a function of the initial population size used by the different problem instances of 2-trap, 3-trap, and 4-trap functions.

- For a given problem instance, values of ρ decrease as values of τ increase.
- Small values of τ usually report a better GA performance.
- As the problem scales, the set of strategies is shifted to higher values of τ.

Therefore, as a good strategy, the GA tends to end with a high percentage of the initial population size (ρ) but losing individuals from a very early stage on the GA run (τ). Nevertheless, large instances require that the population remains almost intact for a longer period, allowing a shrinkage mainly at the last stage of the GA run.

Despite the fact that SVPS has not been designed to find an optimal variability in the population size, it improves the GA performance. Such a result is significant since it proves that variable population sizing schemes are an open issue in the design of GAs.

6. CONCLUSIONS

In this paper we have presented a framework to test variable population sizing schemes. Additionally, a Simple Variable Population Sizing (SVPS) scheme is proposed in order to gain some insights into the population requirements for the different stages of a GA run. The framework consists in a three step methodology. The first and the second steps provide the initial population to be used for the variable population sizing scheme. This initial population represents the minimum initial supply of raw BBs that a fixed-size selectorecombinative GA needs to converge to the optimum solution. The third step is a deterministic resizing schedule configured by a speed and a severity parameter.

The framework has been designed to be compliant with the recommendations of Lobo and Lima [19] for the analysis of variable population sizing schemes:

- The test suite (i.e. trap functions) has known population sizing requirements.

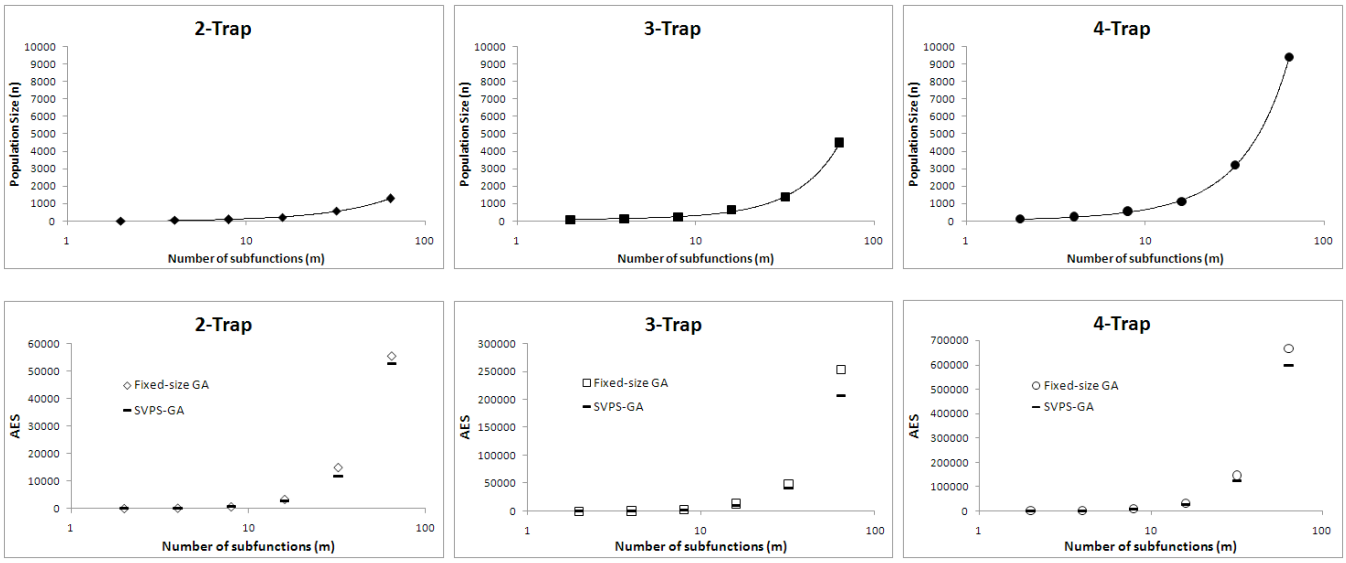

Figure 4: Scalability with trap functions. Optimal population size (top) and Average Evaluations to Solution (AES) (bottom) values for a fixed-size GA and the SVPS-GA.

	2-Trap					3-Trap					4-Trap				
	Fixed-size GA		SVPS-GA			Fixed-size GA		SVPS-GA			Fixed-size GA		SVPS-GA		
m	n	AES	AES	τ	ρ	n	AES	AES	τ	ρ	n	AES	AES	τ	ρ
2	19.0	30.02 ±33.57	**22.53 ±8.17**	0.42	0.25	67	101.69 ±49.76	100.96 ±45	7.2	0.25	110	268.57 ±182	259.8 ±152	2.13	0.25
			23.61 ±11.47	0.18	0.5								264.02 ±168	2.84	0.3
			24.76 ±13.92	0.28	0.3								267.3 ±161	0.125	0.75
4	49	170 ±104	**144.02 ±96.8**	0.63	0.7	112	597.9 ±221	**467.04 ±136**	0.28	0.7	230	1768.46 ±620	**1549.14 ±428**	2.84	0.7
			145.56 ±64.5	0.28	0.75			562.57 ±188	3.2	0.25			1636.63 ±563	0.42	0.95
			155.54 ±83.4	0.9	0.65			581.06 ±211	2.13	0.55			1677.57 ±549	3.20	0.65
8	80	668.49 ±179.8	**582.98 ±169**	7.2	0.55	220	2641.98 ±648.5	**2192.42 ±381**	0.9	0.9	543	8714.10 ±1952	**7713.2 ±1362**	4.8	0.6
			625.10 ±194	10.81	0.4			**2235.63 ±363**	0.18	0.95			**7822.95 ±1205**	7.2	0.35
			637.63 ±199	24.32	0.3			**2452.55 ±397**	7.2	0.4			**7892.44 ±1687**	0.9	0.95
								2471.52 ±532	4.8	0.65			**8223.72 ±1153**	10.81	0.25
16	230	3355.57 ±584.9	**2804.08 ±436**	0.125	0.95	639	13451.8 ±1861	**10763.38 ±1120**	0.42	0.95	1097	30721.04 ±3348	**28384.51 ± 2259**	10.81	0.55
			2896.72 ±373	2.13	0.8			**11585.95 ± 988**	4.8	0.6			**28602.94 ±2646**	2.13	0.95
			2984.48 ±336	7.2	0.25			**11636.61 ±891**	7.2	0.25			**29212.24 ±2348**	16.21	0.35
			3015.1 ±381	4.8	0.5			**11684.92 ±923**	3.20	0.8			**29456.2 ±2674**	7.2	0.8
								12023.58 ±1188	2.13	0.9			**29460.75 ±2606**	24.32	0.25
32	568	14943.96 ±1475	**11661.64 ±875**	0.63	0.95	1353	48521.98 ±5257	**40911.83 ±2980**	4.8	0.75	3225	146072.28 ±10777	**123758.85 ±6228**	4.8	0.8
			12401.14 ±834	3.20	0.8			**44037.66 ±2985**	10.81	0.6			**127900.89 ±6957**	7.2	0.7
			13223.79 ±1032	10.81	0.35			**46072.4 ±3342**	16.21	0.5			**134020.22 ±6396**	16.21	0.3
			13327.55 ±1285	7.2	0.65			**46519.18 ±2925**	24.32	0.25			**134052.36 ±9115**	3.2	0.95
			13802.71 ±1102	16.21	0.25								**138275.59 ±7375**	24.32	0.25
64	1280	55415.1 ±3501	**52773.36 ±2408**	16.21	0.7	4480	252368.92 ±13604	**207263.65 ±8435**	3.2	0.9	9408	666693.86 ±37471	**596770.36 ±22940**	10.81	0.65
			52843.26 ±2494	24.32	0.25			**227992.93 ±9315**	10.81	0.65			**619930.93 ±25075**	16.21	0.55
			52963.61 ±2843	10.81	0.85			**229857.77 ±8830**	16.21	0.3			**632222.06 ±25380**	24.32	0.45
			54897.51 ±3868	7.2	0.95										

Table 2: Experimental results, values in bold are statistically significant.

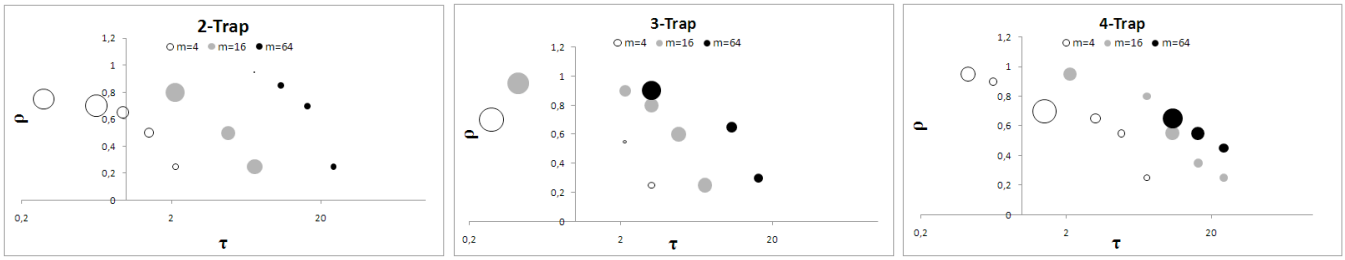

Figure 5: SVPS-GA combinations of τ-ρ converging to a SR of 0.98. From left to right, 2-trap, 3-trap, and 4-trap are represented for sub-functions values of $m = 4$, $m = 16$, and $m = 64$. The area of the circles stands for the AES improvement with respect to the fixed-size GA.

- A scalability analysis has been conducted by varying the size and the complexity of the problem instances.
- The initial population size is well adjusted so that the GA will converge to the problem optima with a success rate of 0.98. The initial supply of BBs and the solution quality are fixed for both population sizing schemes, allowing a fair comparison based on the difference of the number of evaluations.

Preserving the condition of optimality (i.e. success rate of 0.98), the SVPS provides not a single but a set of strategies. Independently of the problem instance, the set follows a common pattern: a large value of τ implies a small one of ρ, that is, the population shrinkage strategy has to keep a balance between severity and speed of the population shrinkage. Results show that SVPS requires a smaller number of evaluations than the fixed population sizing scheme, and therefore, improves the GA performance.

Future work will include studying how the addition of variation operators such as mutation affect performance and its scaling, and also fine-tuning which values of ρ and τ are the most appropriate for a wide range of applications.

Acknowledgements

This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025, and the Granada University PIUGR 9/11/06 project. The authors are grateful to Annette Schwerdtfeger for proofreading this manuscript.

7. REFERENCES

[1] D. H. Ackley. *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[2] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS – A genetic algorithm with varying population size. *International Conference on Evolutionary Computation (IEEE-CEC 1994)*, pages 73–78 vol.1, Jun 1994.

[3] 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)*, pages 315–324, London, UK, 2000.

[4] J. Costa, R. Tavares, and A. Rosa. An experimental study on dynamic random variation of population size. *IEEE conference on Systems, Man, and Cybernetics (SMC 1999)*, 1:607–612 vol.1, 1999.

[5] F. F. de Vega, E. Cantú-Paz, J. López, and T. Manzano. Saving resources with plagues in genetic algorithms. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 272–281, 2004.

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, 1999.

[7] 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)*, pages 41–50, 2004.

[8] A. E. Eiben, M. Schoenauer, D. W. F. van Krevelen, M. C. Hobbelman, M. A. ten Hagen, and R. C. van het Schip. Autonomous selection in evolutionary algorithms. In *Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 1506–1506, 2007.

[9] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[10] L. Eshelman. The CHC adaptive search algorithm: how to have safe search when engaging in non-traditional genetic recombination. In *Foundations of Genetic Algorithms (FOGA 1991)*, pages 265–283, 1991.

[11] C. Fernandes and A. Rosa. Self-regulated population size in evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN IX)*, pages 920–929, 2006.

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

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

[14] G. Harik, D. E. Goldberg, E. Cantú-paz, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. In *IEEE Conference on Evolutionary Computation (IEEE-CEC 1997)*, pages 7–12, 1997.

[15] V. Koumousis and C. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, Feb. 2006.

[16] J. Laredo, P. Castillo, A. Mora, J. Merelo, and C. Fernandes. Resilience to churn of a peer-to-peer evolutionary algorithm. *Int. J. High Performance Systems Architecture*, 1(4):260–268, 2008.

[17] J. Laredo, A. Eiben, M. van Steen, and J. Merelo. On the run-time dynamics of a peer-to-peer evolutionary algorithm. In *Parallel Problem Solving from Nature (PPSN X)*, pages 236–245, 2008.

[18] F. G. Lobo and C. F. Lima. Revisiting evolutionary algorithms with on-the-fly population size adjustment. In *Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 1241–1248, 2006.

[19] F. G. Lobo and C. F. Lima. Adaptive population sizing schemes in genetic algorithms. In *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence, pages 185–204. Springer Berlin / Heidelberg, 2007.

[20] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Technical Report 2002004, University of Illinois at Urbana-Champaign, Urbana, IL., 2001.

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