1-1-2007

Greedy Population Sizing for Evolutionary Algorithms

Ekaterina Smorodkina

Daniel R. Tauritz
Missouri University of Science and Technology, tauritzd@mst.edu

Recommended Citation

E. Smorodkina and D. R. Tauritz, "Greedy Population Sizing for Evolutionary Algorithms," *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (2007: Sep. 25-28, Singapore)*, pp. 2181-2187, Institute of Electrical and Electronics Engineers (IEEE), Jan 2007.
The definitive version is available at https://doi.org/10.1109/CEC.2007.4424742

Greedy Population Sizing for Evolutionary Algorithms

Ekaterina Smorodkina and Daniel Tauritz, *Member, IEEE*

***Abstract*— The number of parameters that need to be manually tuned to achieve good performance of Evolutionary Algorithms and the dependency of the parameters on each other make this potentially robust and efficient computational method very time consuming and difficult to use. This paper introduces a Greedy Population Sizing method for Evolutionary Algorithms (GPS-EA), an automated population size tuning method that does not require any population size related parameters to be specified or manually tuned a priori. Theoretical analysis of the number of function evaluations needed by the GPS-EA to produce good solutions is provided. We also perform an empirical comparison of the performance of the GPS-EA to the performance of an EA with a manually tuned fixed population size. Both theoretical and empirical results show that using GPS-EA eliminates the need for manually tuning the population size parameter, while finding good solutions. This comes at the price of using twice as many function evaluations as needed by the EA with an optimal fixed population size; this, in practice, is a low price considering the amount of time and effort it takes to find this optimal population size manually.**

I. Introduction

The multitude of parameters that need to be tuned to achieve a successful performance of Evolutionary Algorithms (EAs) makes EAs difficult to use. EA experts spend a significant amount of time manually tuning various parameters before performing experiments or before using an EA to solve real-world problems. Moreover, non-experts tend to shy away from EAs because the parameter tuning process requires extensive EA expertise. It is our belief that automating EA parameter settings while maintaining competitive performance will be beneficial to both experts and non-experts, as it would save experts much preparation time that they usually spend on a priori parameter tuning and will allow non-experts to utilize EAs for hard computational problems.

In this paper we examine automating the tuning of the population size parameter, which in traditional EAs has to be set by the user a priori and remains unchanged during the evolutionary process. The optimal population size depends on a particular problem instance, a particular EA configuration [1], and the maximum number of function evaluations the EA will be allowed to use. Setting population size to a value that is too small for a particular problem instance and a particular EA configuration would result in convergence to suboptimal solutions, whereas using a population size that is too large would result in a very slow convergence rate. In this paper we introduce an EA with a Greedy Population Sizing method (GPS-EA), an algorithm that borrows ideas from [1] and does not require setting the population size parameter (or any other population size related parameters) a priori, while maintaining a competitive performance.

We examine the theoretical number of function evaluations needed for any given configuration of GPS-EA to achieve the performance of the EA with the same configuration and a Optimal fixed Population Size (OPS-EA). Theoretical analysis is supported by experimental comparison of GPS-EA and OPS-EA on the Spears' multimodal problem instances [2] using various EA configurations.

E. Smorodkina and D. Tauritz are with the Department of Computer Science, University of Missouri-Rolla, Rolla, MO 65401 USA, email: {eas7d3, tauritzd}at umr.edu.

[1] In this paper we use the term EA *configuration* to refer to the set of operators and parameters used by the EA, with the exception of the population size parameter and the maximum number of function evaluations. We use the term EA *setup* to refer to an EA configuration together with the maximum number of function evaluations.

II. Related Work

In traditional EAs the population size is a user-defined parameter that remains unchanged throughout the evolution, whereas some researchers have indicated that adapting the population size during the evolution may be of benefit [3], [4]. Two well known EAs that are claimed to have an adaptive population size are: the Genetic Algorithm with Adaptive Population Size (APGA) [3] and the Genetic Algorithm with Varying Population Size (GAVaPS) [5]. In both of these algorithms each individual is given a lifetime, which depends on the individual's fitness. At each generation, the lifetime of the individuals is decreased by one. The main difference between APGA and GAVaPS is that in APGA the lifetime of the fittest individual at each generation remains unchanged, while in GAVaPS the lifetime of all individuals in the population is decreased at each generation. Both APGA and GAVaPS eliminate the need for the population size parameter, but introduce two additional parameters $MinLT$ and $MaxLT$, which stand for minimum and maximum lifetime of an individual. Unfortunately, these parameters still need to be manually tuned a priori.

Harik and Lobo [1] introduced a parameter-less Genetic Algorithm (parameter-less GA), an algorithm that evolves several populations of various fixed sizes in parallel and does not require setting the population size parameter a priori. The sizes of the evolving populations are powers of 2 starting with a population of size 4 and each time doubling the size of the largest population to produce the next largest population. Smaller populations are preferred; this preference is specified by a parameter which tells how many generations of a population are evolved before the next larger population has a chance to evolve one generation; in [1] this parameter is a fixed value. A population is deleted in the course of the evolution if a) its average fitness is exceeded by the average fitness of the immediate larger population, or b) the population converged to a solution.

1-4244-1340-0/07$25.00

In a later study [6] the performance of EAs with various population sizing schemes, including the parameter-less GA of [1], was compared on a set of Spears' multimodal problem instances [2], although Lobo and Lima [7] later showed that this comparison was unfair to the parameter-less GA. The population sizing methods under comparison in [6] were: the GAVaPS from [5], the APGA from [3], traditional genetic algorithm (TGA), the Population Resizing on Fitness Improvement Genetic Algorithm (PRoFIGA) introduced in [6], and the parameter-less GA of [1]. APGA was reported to be the winner of this competition, while the parameter-less GA was reported to have the worst performance.

As mentioned earlier, Lobo and Lima [7] showed that the comparison of the population sizing methods described in [6] was unfair to the parameter-less GA; that is because: a) the number of allowed function evaluations was too small, which is demonstrated by the very low success rate of all algorithms on larger problem instances, b) the 7 parameters needed for PRoFIGA were hand tuned a priori, and c) the parameter-less GA was not properly implemented. Lobo and Lima repeated some of the experiments of [6] allowing more function evaluations and using a proper implementation of the parameter-less GA and showed the superiority of the parameter-less GA over APGA on the problem instances used in their experiments. Moreover, in [7] it is shown both theoretically and empirically that if at each generation of APGA two new individuals are created, then APGA is not capable of truly adapting the population size during the course of evolution, instead the population size converges to roughly $MinLT + MaxLT$, which are hand tuned parameters. Using the same analysis as in [7] we can conclude that if at each generation of APGA, c children are created, then the population size of APGA converges to $\frac{c}{2}(MinLT + MaxLT)$, thus again APGA does not truly adapt the population size.

The existing adaptive population sizing methods, with the exception of the parameterless-GA, eliminate the population size parameter, but introduce new parameters that are used to configure the population size during the course of the evolution. Thus using such methods still requires much time and expertise to tune the new parameters. Our Greedy Population Sizing (GPS) method eliminates all population size related parameters completely, while maintaining good performance. We base our algorithm on the existing method that does eliminate all population size related parameters, the parameter-less GA of [1], and try to improve on it by greedily selecting two populations to be evolved in parallel, instead of allowing an unbounded number of populations to evolve in parallel. Note that at the present time we do not compare the performance of our method to that of the parameter-less GA [1], and leave such comparison for the near future. Instead we establish some theoretical background for our method and empirically compare its performance to the performance of EAs with manually tuned (to nearly optimal) population size.

III. GPS-EA

A. Methodology

The GPS-EA evolves two populations in parallel, population P_i and population P_{i+1}, where the size of population P_{i+1} is twice the size of population P_i. This idea was borrowed from the parameter-less GA [1]. The difference between GPS-EA and the parameter-less GA is that GPS-EA limits the parallel evolution to two populations, while the parameter-less GA does not have a limit on the number of populations that can be evolved in parallel.

At each *iteration* of the GPS-EA, an equal number of function evaluations is allocated for each population and both populations are evolved until they use up their allocated number of function evaluations. The number of function evaluations allocated at each iteration for each population equals the number of function evaluations needed for the larger population, P_{i+1}, to evolve one generation (i.e., the number of offspring that will be produced by population P_{i+1} in the next generation; it is assumed that the number of offspring produced at each generation by population P_i is no more than the number of offspring produced at each generation by population P_{i+1}). For example, if P_{i+1} produces M offspring in each generation (and thus requires M function evaluations to evolve one generation) and P_i produces $M/2$ offspring in each generation (and thus requires $M/2$ function evaluations to evolve one generation), then at each *iteration* of the GPS-EA, population P_i is evolved for two generations, while population P_{i+1} is evolved for one generation. Thus at each *iteration* of the algorithm, both populations use the same number of function evaluations but do not necessarily evolve the same number of generations. This iterative process continues until population P_i expires. The expiration of population P_i is a sign that using a larger population would be beneficial. Population P_i expires if one of the following conditions is met:

1) The average fitness of population P_{i+1} becomes higher than the average fitness of population P_i with a 5% significance level using a one-tailed t-test.
2) The maximum fitness of population P_i reached a plateau[2] and both of the following conditions are true:
   - the maximum fitness of population P_i is higher than the maximum fitness of population P_{i-1} (this is necessary, because if the larger population (P_i) plateaued at a maximum fitness value that is lower than the maximum fitness of the smaller population (P_{i-1}), then it is not likely that increasing the population size would be beneficial)
   - the number of unused function evaluations is greater than or equal to the number of function evaluations used by population P_{i+1} so far[3] (this condition prevents the algorithm from creating a larger population if it is known that the algorithm

[2]In our experiments we defined plateau as lack of improvement after 800 function evaluations.

[3]This condition is only applicable when maximum number of function evaluations is defined.

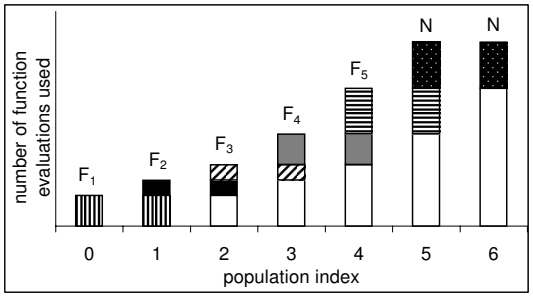


Fig. 1. An example run of GPS-EA. Each column corresponds to a population and the height of the column indicates the total number of function evaluations used by that population in the course of the algorithm execution. The sections of any two columns of the same color or pattern (other than white) correspond to the function evaluations used in the parallel evolution of those two populations. The white section of any column corresponds to line 11 of Algorithm 1, where a newly created largest population is given a chance to "catch up" with the immediate smaller population.

will be terminated before this larger population will have a chance to reach solutions that are competitive with the solutions found by the smaller populations).

When population P_i expires, the counter i is incremented and a new population P_{i+1} is created with twice the size of population P_i. The newly created population P_{i+1} is scheduled to evolve until it uses as many function evaluations as the population P_i has used so far. After this, the parallel evolution of the two populations takes place. The process of creating new populations with a larger size and evolving two populations in parallel continues until a user defined termination condition is met. The pseudo-code for GPS-EA is given in Algorithm 1.

Algorithm 1 GPS-EA

1: $i \leftarrow 0$
2: initialize P_i and P_{i+1}, where $|P_{i+1}| = 2|P_i|$
3: **while** termination condition did not occur **do**
4: $M \leftarrow$ number of function evaluations needed to evolve one generation of P_{i+1}
5: evolve P_i until it uses up M function evaluations
6: evolve P_{i+1} until it uses up M function evaluations
7: **if** population P_i expired **then**
8: $i \leftarrow i + 1$
9: initialize P_{i+1} such that $|P_{i+1}| = 2|P_i|$
10: $F \leftarrow$ number of function evaluations used by P_i
11: evolve P_{i+1} until it uses up F function evaluations
12: **end if**
13: **end while**

A graphical example of a run of GPS-EA is shown in Figure 1, where each column corresponds to a population and the height of the column indicates the total number of function evaluations used by that population in the course of the algorithm execution. The white color on any particular column corresponds to line 11 of Algorithm 1, where a newly created population P_{i+1} is given a chance to "catch up" with population P_i. Any two neighboring columns have a section of the same color or pattern (other than white), which corresponds to the function evaluations used during the parallel evolution of those two populations. In this example, populations P_0 and P_1 are evolved in parallel using F_1 function evaluations each (corresponding to the vertically striped sections of columns 0 and 1), then population P_0 expires and population P_2 is created and is evolved until it uses F_1 function evaluations (corresponding to the white section of column 2). Then populations P_1 and P_2 are evolved in parallel (corresponding to the black sections of columns 1 and 2). After P_1 uses up a total of F_2 function evaluations, it expires and population P_3 is created and evolved for F_2 function evaluations (corresponding to the white section of column 3), after which the parallel evolution of P_2 and P_3 takes place (corresponding to the diagonally striped sections of columns 2 and 3). This process continues and the algorithm is terminated after P_6 uses up a total of N function evaluations.

B. Theoretical bound on the number of function evaluations

For any EA configuration with a fixed population size, the optimal population size is the one that leads to a solution better than the solution found by the same EA configuration with any other population size using the same amount of resources (i.e., the same number of function evaluations). The optimal population size depends on the particular problem instance being solved, the EA configuration, and the allowed number of function evaluations. This means that the optimal population size of any given EA configuration after N function evaluations may not be the same as the optimal population size of the same EA configuration after $2N$ function evaluations. Suppose that after N function evaluations the optimal population size of a given EA configuration is between 2^{k-1} and 2^k, call this population P_k. The underlying assumptions of GPS-EA are:

1) if population P_i is better than population P_{i-1} after N function evaluations, then population P_{i-1} is better than population P_{i-2} after N function evaluations as well, for $i = 1, 2, \ldots, k$ and
2) if population P_i is better than population P_{i-1} after N function evaluations, then the average fitness of P_i is higher than the average fitness of P_{i-1} after N function evaluations, for $i = 1, 2, \ldots, k$.

Let F_i be the number of function evaluations after which the average fitness of P_i exceeds the average fitness of P_{i-1} for $i = 1, 2, \ldots, k$ (note that F_i may be zero). Let $F(max)$ be the upper bound on the total number of function evaluations the GPS-EA needs to find solutions of the same quality as those found by the EA with the optimal fixed population size (size of population P_k) after N function evaluations. Then,

$$F(max) = (2F_1) + (2F_2 - F_1) + \ldots + (2F_k - F_{k-1}) + (2N - F_k); \quad (1)$$

here each term in the parenthesis (with the exception of the last term) corresponds to an expiration of a smaller population. Equation 1 is equivalent to:

$$F(max) = F_1 + F_2 + \ldots + F_{k-1} + 2N. \quad (2)$$

Consider again the example run of GPS-EA shown in Figure 1. In this case, $k = 5$ and $F(max) = F_1 + F_2 + F_3 + F_4 + F_5 + 2N$.

Thus the efficiency of GPS-EA depends on the values of F_i, for $i = 1, 2, \ldots, k$. If the values of F_i for $i \leq k$ are negligible as compared to N, then the GPS-EA should find the same solution in roughly twice as many function evaluations as the one found by the EA with the optimal fixed population size. However, if the values of F_i are close to N then the solution found by the EA with the optimal fixed population size is probably not much better than the solution found by the same EA with a smaller population size.

We will now support this theoretical analysis with empirical evidence, by comparing the performance of the GPS-EA to the performance of an EA with a optimal fixed population size (see Section IV-C for the details on the optimal population size), where GPS-EA is allowed twice as many function evaluations as the EA with the optimal population size.

IV. Experiment Design

A. Test suite

At the present time the performance of GPS-EA is evaluated in the context of Genetic Algorithms (GPS-GA) on Spears' Multimodal Problem (SMP) instances [2]. These test problem instances have a controllable complexity, size and degree of multi-modality and have been argued to facilitate systematic studies of GA behavior [6]. SMP-generator creates P L-bit strings, each of which represents a peak in an L-dimensional space. The objective of the problem is to find the location of the highest peak. The values of the peaks are in the range [M,1], where $M \geq 0$ is a user defined value; the peaks can be generated using various functions: constant, linear, 1-square root, and logarithm based [6]. The highest peak for any such problem has the value of 1. The fitness of a bit string b, representing a candidate solution, of length L is defined as:

$$f(b) = \frac{L - D(b, Peak_n(b))}{L} \cdot height(Peak_n(b)) \quad (3)$$

where $Peak_n(b)$ is the nearest peak in Hamming space to the bit string b, and $D(b, Peak_i)$ is the Hamming distance between the candidate solution b and $Peak_n(b)$. A linear function was used to generate peaks, and the parameter M, the height of the lowest peak, was set to 0.5. The numbers of peaks used to generate problem instances were: 10, 100, 500, and 1000.

B. Performance measures

For each number of peaks, 60 independent runs of each algorithm were performed. The performance of the algorithms was measured based on the following three commonly used statistics:

1) Mean Best Fitness (MBF) - the average over all the runs of the best fitness found by the algorithm.
2) Success Rate (SR) - the percentage of the runs that resulted in the optimal solution.
3) Average number of Evaluations to a Solution (AES) - the total number of function evaluations for the runs that resulted in the optimal solutions averaged over the number of runs that resulted in the optimal solution. If no runs resulted in the optimal solution, this measure is undefined.

C. Algorithm setups

For each test problem, the performance of the GPS-GA was compared to the performance of a regular GA with population size 2^j using three EA configurations. The value j was manually chosen for each number of peaks, such that the MBF of the GA with population size 2^j was superior to the MBF of the same GA with a population size 2^{j-1} and a population size 2^{j+1}. In this context we refer to the population size 2^j as *Optimal Population Size* (OPS), but keep in mind that the true optimum is between 2^{j-1} and 2^{j+1}. We refer to a GA with OPS as OPS-GA. It is important to note that the optimal population size is typically not known and was computed manually for the purpose of evaluating the performance of the GPS-GA. Manual computation of the optimal population size is time consuming and the whole purpose of GPS-GA is to eliminate the need for manually tuning the population size. GPS-GA was allowed twice as many function evaluations as used by OPS-GA, as suggested by the theoretical analysis of Section III-B. The performance of the algorithms was compared using three algorithm setups, described in Table I, where parameters common to two or more EA setups span over the columns corresponding to the EA setups to which they apply.

V. Experimental Results

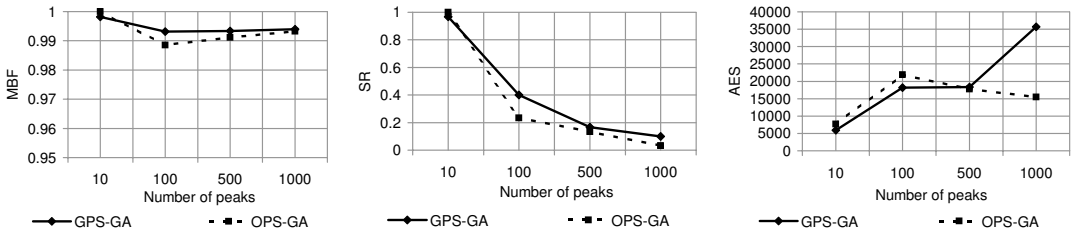
The optimal population sizes for the OPS-GA for each EA setup and each number of peaks were determined by running each EA setup on each number of peaks using populations of sizes 2^i 60 times each, where $i = 2, 3, \ldots, j+1$, such that 2^j is the OPS as described in Section IV-C. The resulting values of OPS for each EA setup and each number of peaks are shown in Table II. Since GPS-GA does not require a priori manual tuning of the population size parameter, the overall total number of function evaluations used by the GPS-GA in the course of the experiments was smaller than the overall total number of function evaluations used to configure and evaluate OPS-GA, as shown in Table III.

The results of the experimental comparison of MBF, SR, and AES of the GPS-GA versus the OPS-GA on the three EA setups are shown in Figure 2 - Figure 4. These results

TABLE I

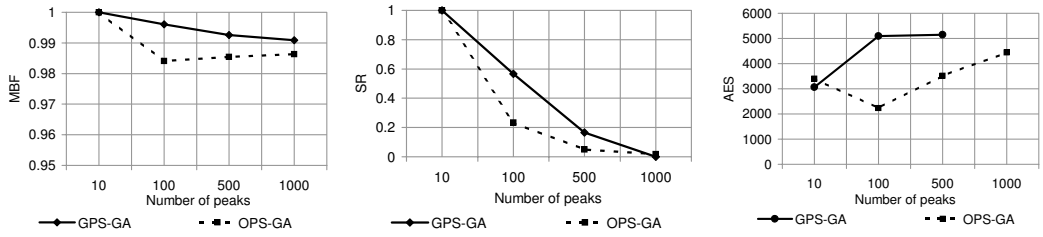
EA SETUPS USED IN THE EXPERIMENTS

	EA_1	EA_2	EA_3
Representation	bit-string		
Chromosome length (L)	100		
Initialization	random with a 50-50 probability of initializing a bit to 1 or 0		
Parent Selection	5-tournament		
Recombination	2-point crossover ($p_c = 1$)		
Mutation	bit-flip ($p_m = 1/L$)		self-adaptive mutation as in [8] with clutchsize=5, 16-bit binary encoding of the mutation rate, initial mutation rate is a random value between 0% and 25%, and maximum allowed mutation rate of 100%
Replacement	5-tournament between parents and children	children replace parents; the fittest individual does not get replaced and does not participate in reproduction	
Max no. of evals	40000 for OPS-GA 80000 for GPS-GA	5000 for OPS-GA 10000 for GPS-GA	



(a) MBF (b) SR (c) AES

Fig. 2. Results of EA_1.

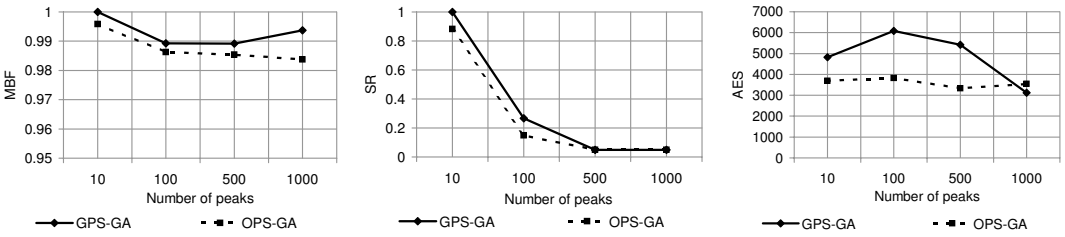


(a) MBF (b) SR (c) AES

Fig. 3. Results of EA_2.

show that the performance achieved by GPS-GA is highly competitive with that of the OPS-GA.

The MBF for all three EA setups and all numbers of peaks of the GPS-GA was as high as, or higher, than the MBF of the OPS-GA. A two-tailed t-test assuming unequal variances was performed to determine the statistical significance of the results and it was determined that the differences between the MBF values produced by both algorithms with EA_1 setup for all numbers of peaks were not statistically significant. The differences between the MBF values produced by the two algorithms with EA_2 setup were statistically significant for 100, 500, and 1000 peaks. Finally, on EA_3 setup, the differences between the MBF values were statistically significant for 10 and 1000 peaks. All t-tests were performed with 5% significance level. Thus in 12 out of 12 cases, doubling the number of function evaluations that were used by the OPS-GA was sufficient for the GPS-GA to achieve the value of the MBF statistic as good as or better than the value of the



(a) MBF (b) SR (c) AES

Fig. 4. Results of EA_3.

TABLE II
POPULATION SIZES USED FOR OPS-GA

Number of peaks:	10	100	500	1000
Population size for EA_1	32	64	64	64
Population size for EA_2	128	64	128	128
Population size for EA_3	32	32	32	32

TABLE III
TOTAL NUMBER OF FUNCTION EVALUATIONS USED TO CONFIGURE ALGORITHMS AND TO PRODUCE RESULTS THAT ARE USED IN COMPARING PERFORMANCE MEASURES OF GPS-GA AND OPS-GA (BASED ON 60 RUNS OF EACH ALGORITHM)

No. of peaks:		10	100	500	1000
EA_1	GPS-GA	1600000	1600000	1600000	1600000
	OPS-GA	12000000	14400000	14400000	14400000
EA_2	GPS-GA	600000	600000	600000	600000
	OPS-GA	2100000	1800000	2100000	2100000
EA_3	GPS-GA	600000	600000	600000	600000
	OPS-GA	1500000	1500000	1500000	1500000

MBF statistic achieved by the OPS-GA.

The SR in all but two cases (EA_1 on 10 peaks and EA_2 on 1000 peaks) of the GPS-GA was at least as high as the SR of the OPS-GA. Thus in 10 out of 12 cases, doubling the number of function evaluations that were used by the OPS-GA was sufficient for the GPS-GA to achieve the value of the SR statistic as good as or better than the value of the SR statistic achieved by the OPS-GA. The two cases where the SR statistic of the GPS-GA was worse (lower) than the SR statistic of the OPS-GA can be explained by possible high values of F_i, as defined in Section III-B.

The AES in all but two cases (EA_1 on 1000 peaks and EA_2 on 100 peaks) of the GPS-GA is no more than twice the AES of the OPS-GA (in the case of EA_3 on 1000 peaks, the AES of the GPS-GA is undefined because the SR of the GPS-GA with this setup on 1000 peaks is zero). Thus in 9 out of 11 cases, the AES statistic shows that the number of function evaluations needed to find the optimal solution by the GPS-GA is indeed within a factor of 2 of the number of function evaluations needed by the OPS-GA to find the optimal solution. The other two cases can again be explained by possible high values of F_i, as described in Section III-B.

We noticed an unexpected trend in the AES of the GPS-GA with EA_3, where the AES for 1000 peaks was less than the AES for 500 peaks. This could be an artifact of the self-adaptive mutation rate used by EA_3, but the real reason remains unknown.

Overall, these results indicate that GPS-GA is a practical and realistic method of removing the population size parameter, without sacrificing the quality of the solutions and without introducing any additional parameters.

VI. SUMMARY AND CONCLUSIONS

In this paper we introduced an EA with a Greedy Population Sizing method (GPS-EA), as a way of completely eliminating all population size related parameters in an EA. This method is of benefit to both EA experts and non-experts as it allows the experts to save the time they usually spend on the a priori tuning of the population size parameter and facilitates the usage of EAs by non-experts. We performed theoretical and experimental analysis of the GPS-EA, focusing on two issues: a) establishing the theoretical bound on the number of function evaluations needed by the GPS-EA to achieve a competitive performance, and b) experimentally verifying that the performance of the GPS-EA, given the theoretically established necessary number of function evaluations, is indeed competitive with the performance achieved by the EA with an Optimal fixed Population Size (OPS-EA).

The results of our theoretical and experimental analysis indicate that GPS-EA eliminates all population size related parameters by automating its configuration and tuning, while achieving solutions as good as or better than those found by using OPS-EA. This comes at the price of using twice as many function evaluations as needed by the OPS-EA. This is a relatively low price, considering the amount of time and expertise required for the manual tuning of the population size parameter to be used by the OPS-EA.

The GPS-EA approach is based on the parameter-less GA of [1] and in the near future we will compare the performance of the GPS-EA to the performance of the parameter-less GA. We also hope to improve the efficiency of the GPS-EA by using some of the evolved solutions to initialize the larger populations.

REFERENCES

[1] G. Harik and F. Lobo, "A parameter-less genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, 1999, pp. 258–265.

[2] J. Kennedy and W. Spears, "Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," in *IEEE Int'l Conference on Evolutionary Computation*, 1998, pp. 74–77.

[3] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart, "An empirical study on GAs without parameters," in *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 2000, pp. 315–324.

[4] A. E. Eiben, M. C. Schut, and A. R. de Wilde, "Is self-adaptation of selection pressure and population size possible? - A case study." in *PPSN 2006: The 9th International Conference on Parallel Problem Solving from Nature*, 2006, pp. 900–909.

[5] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS-a genetic algorithm with varying population size," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1994, pp. 73–78.

[6] A. Eiben, E. Marchiori, and V. Valkó, "Evolutionary algorithms with on-the-fly population size adjustment," in *PPSN VIII: Parallel Problem Solving from Nature*, 2004, pp. 41–50.

[7] F. G. Lobo and C. F. Lima, "Revisiting evolutionary algorithms with on-the-fly population size adjustment," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 1241–1248.

[8] J. Smith and T. Fograty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *IEEE Int'l Conference on Evolutionary Computation*, 1996, pp. 318–323.