doi:10.2498/iti.2012.0382

GAAP. Genetic Algorithm with Auxiliary Populations Applied to Continuous Optimization Problems

Leonardo Corbalán, Waldo Hasperue, Laura Lanzarini
III-LIDI (Institute of Research in Computer Sciences LIDI)
Faculty of Computer Sciences. National University of La Plata.
50 y 120. 2° Piso. La Plata, Buenos Aires, Argentina
{corbalan, whasperue, laural}@lidi.info.unlp.edu.ar

Abstract. *Genetic algorithms have been used successfully to solve continuous optimization problems. However, an early convergence to low-quality solutions is one of the most common difficulties encountered when using these strategies.*

In this paper, a method that combines multiple auxiliary populations with the main population of the algorithm is proposed. The role of the auxiliary populations is dual: to prevent or hinder the early convergence to local suboptimal solutions, and to provide a local search mechanism for a greater exploitation of the most promising regions within the search space.

Keywords. Real coded genetic algorithms, continuous optimization, early convergence, exploration, exploitation, local search, auxiliary populations.

1. Introduction

In recent years, scientific research on continuous and discrete optimization has produced a large number of solutions based on deterministic and stochastic algorithms. Among the stochastic type, those belonging to the evolutionary computation (EC) field stand out due to their high performance, reliability, robustness, and global search capacity [1]. They mimic biological evolution as strategy for solving problems.

EC encompasses a large variety of methods and algorithms, among which evolutionary programming (EP) [7,8], evolution strategies (ES) [2], genetic algorithms (GAs), [12] and differential evolution (DE) [16] can be mentioned, among others. All of these are metaheuristics based on populations that follow an evolutionary process with slight differences, and are characterized for their versatility and ability to solve a large number of real-world problems.

Although initially genetic algorithms used almost exclusively binary alphabets to code their chromosomes —binary code [12] and Gray code [3]—, the theory that supported this codification was questioned, and other types of representations appeared [4,6,13]. In particular, real coded genetic algorithms (RCGAs) [14,17], which use real numbers to code solutions, have been successfully applied to continuous optimization problems. Various research works on RCGAs [5,10,11,15] have resulted in an increased interest for problem resolution with this type of algorithms.

In RCGAs, the same as with other EC methods, early convergence to low-quality solutions is a drawback that degrades algorithm performance. This is related to the loss of diversity in the population that reduces the exploratory ability, concentrating most of the individuals in the same region of the search space.

The GAAP strategy proposed in this paper uses an RCGA that delays its convergence through the addition of new genetic material. This addition comes from small, isolated populations, disperse in the search space. Also, the creation and elimination of these populations that evolve for a few generations in reduced environments around good solutions increases the exploitation capacity of promising regions without degrading the global exploration capacity of the main population. The early addition of these auxiliary populations reduces concentration in neighboring areas and prevents early convergence.

GAAP has shown a good balance between exploration and exploitation, combining a global evolutionary process with evolutionary local search areas to refine certain individuals in the population.

2. Optimization problems

Basically, an optimization problem consists in searching for the best mapping of values to a set of variables with the purpose of achieving a certain objective.

More formally, an optimization problem is defined through a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, the domains of each variable $D_1, D_2, \ldots, D_n$, a set of restrictions imposed on the variables, and a target function f that has to be maximized —or minimized,— where $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \Re$. The problem's *n*-dimensional search space S is formed by all possible variable mappings. Resolving the optimization problem implies finding the solution $X^* \in S$ that maximizes —or minimizes— the target function f, i.e., $f(X^*) \geq f(X)\ \forall X \in S$ —or $f(X^*) \leq f(X)\ \forall X \in S$—. In this case, X^* is known as the global optimum of the problem.

There are two significant categories of optimization problems: those that code solutions with discrete variables, called combination optimization problems, and those that code solutions with real variables, called continuous optimization problems. GAs and RCGAs have been used to solve these types of problems.

3. GAs and RCGAs

GAs are populational metaheuristics that work with individuals represented by their chromosomes. Each chromosome —genotype— codes a solution to the problem at hand —phenotype—. The fitness value assigned to each solution quantifies the quality of the solution, represents the extent to which the individual is adapted to its environment —problem to solve— and is directly proportional to the probability of being chosen for reproduction, crossing its genotype with that of another individual. The descendants produced by this crossover share the features of their parents. Thus, a new generation of solutions with a greater proportion of good features is produced. By favoring the crossover of the fittest individuals, the most promising areas of the search space are explored, ideally converging towards an optimal solution to the problem.

GAs most commonly code the solutions in a binary chromosome. However, real coding of RCGAs is the most natural alternative to tackle a certain type of problems, including continuous optimization. Coding chromosomes with real numbers allows each gene to directly represent a variable of the problem.

At first, despite the good practical results, the real coding of chromosomes was resisted because theory suggested that low-cardinality alphabets would be more efficient. However, the appearance of tools for the theoretical treatment of RCGAs allowed corroborating the good practical performance that had already been achieved [9].

4. Strategy proposed. GAAP

GAAP improves the performance of a simple RCGA by using auxiliary populations in reduced environments that are separate and preferably distant from each other. The effect of these populations is two-fold: they prevent the early convergence by keeping individuals in different regions of the search space, and they intensify the search in promising regions. Thus, a good balance between the exploration of the global space and the exploitation of good regions is obtained.

4.1. General structure of GAAP

Figure 1 shows the general scheme of a GAAP. For each generation in the main population, full evolutionary processes are carried out on small auxiliary populations. These populations are created; they evolve, transform the main population, and then disappear. The algorithm continues with the classic *selection*, *reproduction* and *replacement* sequence, completing one generation in the main population.

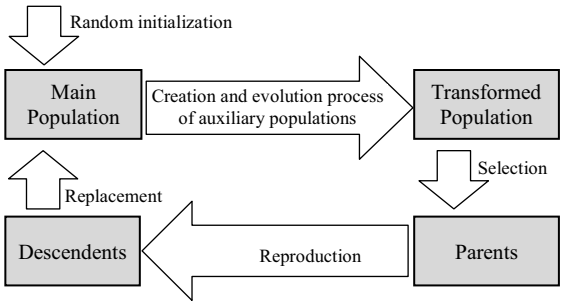

Figure 1. General GAAP scheme

Each auxiliary population is created randomly in a reduced environment around a centroid genotype. This centroid is especially chosen from the main population based on its fitness and

location in the search space.

The auxiliary populations, separated from each other, formed by a few genotypes, undergo short, independent evolutions. Then, all the individuals from the auxiliary populations are incorporated to the main population, replacing the lowest fitness genotypes. To do this, the total number of auxiliary individuals must be significantly lower than the size of the main population. Figure 2 shows a simplified scheme of the creation and evolution process of auxiliary populations and their subsequent replacement in the main population.

Auxiliary populations allow obtaining good solutions fast by intensifying the search in promising regions. But also, the correct selection of a disperse set of centroid genotypes makes early convergence to low-quality solutions difficult. The process for selecting centroid genotypes and generating auxiliary populations is detailed below:

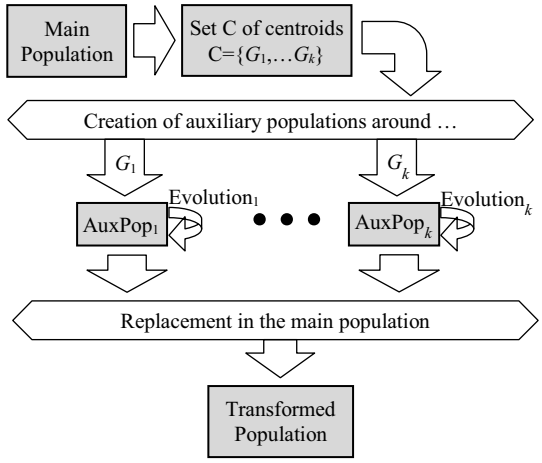

Figure 2. Creation and evolution process of auxiliary populations

4.2. Creation of auxiliary populations

Each auxiliary population AuxPop$_i$ is built by randomly generating individuals in a hyper-sphere of radius r and center in one given genotype G_i. The set of centroid genotypes is determined as follows:

Be the population of genotypes $\mathrm{P} = [G_1, G_2, \dots, G_m]$ a list sorted by fitness value such that $i < j \rightarrow fitness(G_i) \geq fitness(G_j)$. Be k a parameter of the algorithm that indicates the number of auxiliary populations that must be created. The creation process of these populations starts with the selection of k sets of candidates $\mathrm{C}_1, \mathrm{C}_2, \dots \mathrm{C}_k$. From each of these sets, exactly one centroid genotype will be chosen.

Each candidate set C_k has exactly k elements, selected in the order of list P as follows: $\mathrm{C}_1 = \{G_1\}$, $\mathrm{C}_2 = \{G_2, G_3\}$, $\mathrm{C}_3 = \{G_4, G_5, G_6\}$ and in general $\mathrm{C}_j = \{\mathrm{C}_{f(j)+1}, \mathrm{C}_{f(j)+2}, \dots, \mathrm{C}_{f(j)+j}\}$, with $f(j) = (j-1)j/2,\ j = 1 \dots k$. Thus, the genotypes of each set C_j will very likely have similar fitness values, and, therefore, will be equally significant for selecting the centroid G_j in relation with the quality of the solution they represent. It should be noted that $i < j \rightarrow |\mathrm{C}_i| < |\mathrm{C}_j|$, reflecting the fact that the selection of one genotype over another has a greater impact when the fitness values involved are higher.

Having a set of candidates that are equally eligible for each centroid allows for many selection methods. Set C of centroids is built by correctly choosing a genotype of each set C_j that favors the dispersion of the elements in C. This selection process is described below:

$\mathrm{C} \leftarrow \emptyset$
For $j \leftarrow 1..k$
  $\mathrm{C} \leftarrow \mathrm{C} \cup f(\mathrm{C}, \mathrm{C}_j)$
End For

With $f(\mathrm{C}, \mathrm{C}_j) = \{G_r \in \mathrm{C}_j : minDist(G_r, \mathrm{C}) > minDis(G_s, \mathrm{C}),\ \forall s \in \mathrm{C}_j,\ s \neq r\}$

Function $minDist(G_r, \mathrm{C})$ represents the distance between genotype G_r and the closest element to G_r in set C formed so far. Thus, $f(\mathrm{C}, \mathrm{C}_j)$ returns a unit set with the element in C_j that is farthest away from set C, that is, the element whose distance to the closest element in C is greater than that of any other element in C_j.

If G_j is the centroid chosen for the auxiliary population AuxPop$_j$, this population is created by randomly generating genotypes in the hyper-sphere with center in G_j and radius $r = \beta \cdot minDist(G_j, \mathrm{C} - \{G_j\})$, the constant β being a positive parameter of the algorithm ($0 < \beta < 1$). Thus, auxiliary populations are initialized in separate spaces. For the tests carried out as part of this paper, we have used $\beta = 1/4$ with good results.

4.3. Other implementation details

For all populations, the roulette-wheel selection method was used. The genetic operators used for the reproduction stage were crossover with a probability of 0.9, and mutation with a probability of 0.001. A variation of the crossover operator called extended line

recombination was used, which allows obtaining two descendents for each pair of parents G_1 and G_2 by applying equation $D = G_1 + \beta(G_2 - G_1)$ twice, with β selected randomly from the interval $[-0.25,1.25]$.

Elitism was applied to the evolution of all populations, with a copy of the best individual being transferred to the next generation.

Even though the real codification in chromosomes allows tackling the continuous optimization problem working directly on the domain of the function to optimize, GAAP differentiates between genotype and phenotype to achieve a greater independence from the problem. All populations work with genotypes that are coded by real numbers from the interval [0,1]. If the problem to solve is *n*-dimensional, a genotype G will be $G = (g_1, \dots, g_n)$ with $g_i \in [0,1]$. The corresponding phenotype X_G is easily calculated as follows: $X_G = F(g_1, \dots, g_n) = (f_1(g_1), \dots, f_n(g_n))$ with $f_i(g_i) = g_i(b_i - a_i) + a_i$ being $[a_i, b_i]$ the interval on which the i^{th} element of phenotype X_G is defined.

Thus, at the chromosome level, GAAP always works over a hypercube of side 1, and only when the fitness value of an individual has to be assessed, the corresponding phenotype is built by transforming the *n*-dimensional space of the problem as required.

5. Results obtained

GAAP was tested with the continuous function minimization problem for a set of six *n*-dimensional functions with n=30, and its performance was compared with that of a classic RCGA. The set of test functions was formed with a unimodal separable function: Sphere function —De Jong's function 1— in the domain $x_i \in [-5.12,\ 5.12]$ (equation 1); a non-separable unimodal function: the rotated hyper-ellipsoid function —Schwefel's Problem 1.2— in the domain $x_i \in [-65.536,\ 65.536]$ (equation 2); two multimodal separable functions: Schwefel generalized problem 2.6 in the domain $x_i \in [-500,\ 500]$ (equation 3) and Rastrigin function in the domain $x_i \in [-5.12,\ 5.12]$ (equation 4); and two multimodal non-separable functions: Rosenbrock function —De Jong's function 2, multimodal for n>2— in the domain $x_i \in [-2.048, 2.048]$ (equation 5) and Griewangk function in the domain $x_i \in [-600,\ 600]$ (equation 6). The global minimum of function f_{Sch2} for 30 dimensions is -12539.487; for the other functions, the global minimum is 0.

$$f_{Sph}(X) = \sum_{i=1}^{n} x_i^2 \quad (1)$$

$$f_{Sch1}(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \quad (2)$$

$$f_{Sch2}(X) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \quad (3)$$

$$f_{Ras}(X) = 10n + \sum_{i=1}^{n} \left[\begin{matrix} x_i^2 - \\ 10\cos(2\pi x_i) \end{matrix} \right] \quad (4)$$

$$f_{Ros}(X) = \sum_{i=1}^{n-1} \left[\begin{matrix} 100\left(x_{i+1} - x_i^2\right)^2 + \\ (x_i - 1)^2 \end{matrix} \right] \quad (5)$$

$$f_{Gri}(X) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (6)$$

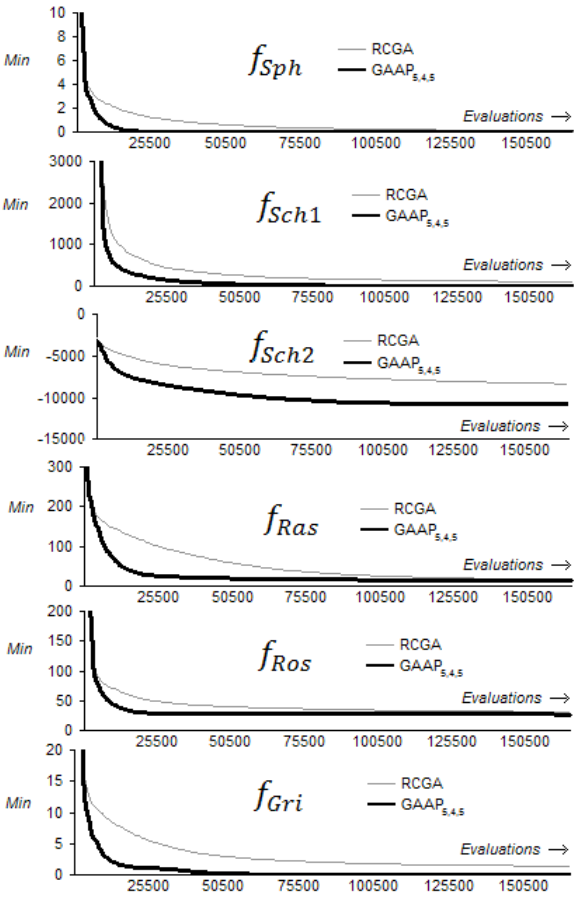

Figure 3. Minimum value found by RCGA and GAAP$_{5,4,5}$ as the algorithm progresses

Forty-two test cases were designed as follows: for each of the functions to optimize, the performance of a classic RCGA and the performance of 6 different configurations of the strategy presented were assessed: $GAAP_{5,8,2}$; $GAAP_{5,4,5}$; $GAAP_{4,10,2}$; $GAAP_{4,6,4}$; $GAAP_{2,10,5}$; $GAAP_{2,4,14}$; where $GAAP_{i,j,k}$ identified a configuration with i auxiliary populations of j individuals with evolutionary auxiliary processes of k generations.

For all GAAP configurations, the main population of 240 individuals underwent an evolutionary process of 460 generations, whereas in the case of RCGA, the population of 240 individuals evolved for 690 generations. Thus, in all the cases assessed, exactly 165840 assessments of the function to optimize are required.

Each of the 42 test cases was tested 50 separate times. In all these tests, every 500 assessments of the function to optimize, the minimum value achieved by the algorithm so far was logged. These logs, averaged for the 50 tests, show the performance of the algorithm as the evolutionary process is run (see Figure 3).

Table 1 shows a comparative statistical study of RCGA versus $GAAP_{5,4,5}$. Sample size was n=50, obtained with 50 separate runs for each method, and a t-Student test was performed to determine whether the difference between the means observed is statistically significant. With the exception of function f_{Ras}, for which the p-value obtained was 0.2099, in all other cases, the difference in the means observed was extremely statistically significant (two-tailed p-value < 0.0001).

Table 1. Unpaired t test results (α=0.05) RCGA vs. $GAAP_{5,4,5}$ (165840 assessments)

f	*Method*	*Mean (n=50)*	*SD (n=50)*	*two-tailed p-value*
f_{Sph}	RCGA	0.108556	0.030868	1.9199E-29
	GAAP	6.292E-07	3.328E-07	
f_{Sch1}	RCGA	106.8317	31.22045	1.3876E-28
	GAAP	1.793029	0.703	
f_{Sch2}	RCGA	-8308.12	690.3504	9.8291E-35
	GAAP	-10729.8	526.2997	
f_{Ras}	RCGA	15.96552	6.503284	0.2099663
	GAAP	14.36006	6.215502	
f_{Ros}	RCGA	31.18416	0.956809	2.740E-40
	GAAP	27.02643	0.501028	
f_{Gri}	RCGA	1.381624	0.128333	6.521E-53
	GAAP	0.011567	0.011328	

All tested GAAP configurations showed a better performance than RCGA, with values that were closer to the global minimum. Functions f_{Sch2}, f_{Ras} and f_{Ros} were the most difficult to solve for both methods. However, the advantages of GAAP over RCGA described before were always present. Figure 4 shows the boxplots corresponding to the minimum values obtained for functions f_{Gri}, f_{sch1}, and f_{sch2} in the 50 experiments carried out.

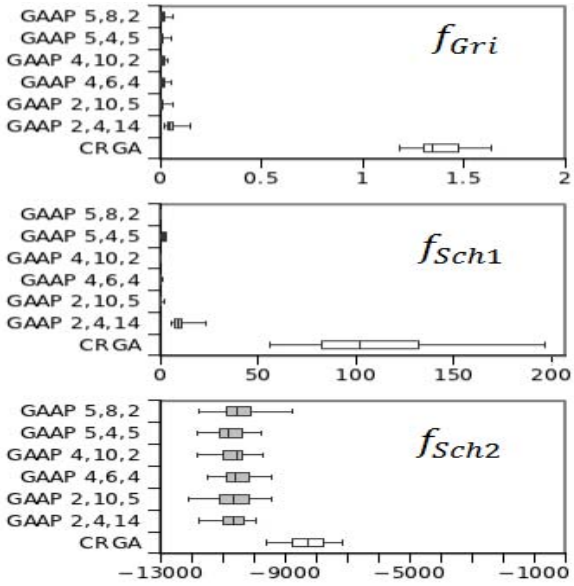

Figure 4. Boxplots

However, the greatest advantage of GAAP was the lower number of assessments required to achieve an acceptable minimum value. Table 2 shows the result of the same test from Table 1 but considering only 33360 assessments of the fitness function

Table 2. Unpaired t test results (α=0.05) RCGA vs. $GAAP_{5,4,5}$ (33360 assessments)

f	*Method*	*Mean (n=50)*	*SD (n=50)*	*two-tailed p-value*
f_{Sph}	RCGA	0.93517	0.31264	3.232E-26
	GAAP	0.00348	0.00235	
f_{Sch1}	RCGA	342.955	106.887	1.027E-21
	GAAP	109.989	44.6512	
f_{Sch2}	RCGA	-6535.99	687.295	1.430E-35
	GAAP	-8999.59	499.361	
f_{Ras}	RCGA	84.1676	22.4986	8.233E-27
	GAAP	22.1830	7.99190	
f_{Ros}	RCGA	44.9836	4.70336	3.421E-30
	GAAP	28.6404	0.55603	
f_{Gri}	RCGA	4.20294	0.90429	5.199E-31
	GAAP	0.82768	0.10618	

In all cases, the difference between the observed means is extremely statistically significant. This demonstrates that GAAP gets closer to the optimum faster and that it requires a lower number of assessments of the function to optimize.

As regards the various configurations of GAAP, the data gathered so far are not conclusive in relation to the advisable number of auxiliary populations, their size, and the number of generations the auxiliary evolutionary processes must produce. The detailed analysis of this issue is presented as a future work.

6. Conclusions and Future Work

We have presented GAAP, a new evolutionary method that extends a classic RCGA with short, independent evolutions of small auxiliary populations that contribute genetic material to the main population. The experiment carried out shows that GAAP applied to continuous optimization problems greatly improves the performance of a RCGA, both for unimodal and multimodal functions, be these separable or non-separable. The results obtained suggest that GAAP offers a good balance between global exploration and the exploitation of the best regions of the search space.

As a future work, the comparison of GAAP with other metaheuristics, both populational and trajectory-based, is proposed. A detailed study of algorithm parameters is also proposed to define exactly the most convenient ranges of values and when they should be applied.

7. Referencias

[1] Back T., Fogel D. B., and Michalewicz Z., Eds., Evolutionary Computation 2: Advanced Algorithms and Operators. Bristol, U.K.: Institute of Physics, (2000).

[2] Beyer HG, Schwefel HP Evolution strategies-A comprehensive introduction. Natural Computing 1(1):3-52 (2002)

[3] Caruana RA, Schaffer JD. Representation and Hidden Bias: Gray versus Binary Coding for Genetic Algorithms. In: Int. Conf. on Machine Learning, pp 153-162 (1988)

[4] Davis L Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)

[5] Deb K A population-based algorithm-generator for real-parameter optimization. Soft Computing 9:236-253 (2005)

[6] Deb K, Beyer H Self-adaptive genetic algorithms with simulated binary crossover. Evol. Comput. 9(2):195-219 (2001)

[7] Fogel DB Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, New Jersey (1995)

[8] Fogel LJ, Owens AJ, Walsh MJ Artificial Intelligence Through Simulated Evolution. John Wiley, New York (1966)

[9] Goldberg DE Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking. Complex Syst. 5:139-167 (1991)

[10] Herrera F, Lozano M, Sánchez AM Hybrid crossover operators for real-coded genetic algorithms: An experimental study. Soft Comput. 9(4):280-298 (2005)

[11] Hervás-Martínez C, Ortiz-Boyer D Analyzing the statistical features of CIXL2 crossover ofspring. Soft Comput. 9(4):270-279 (2005)

[12] Holland JH Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975)

[13] Michalewicz Z Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York (1992)

[14] Ono I., Kita H., and Kobayashi S., Advances in Evolutionary Computing. New York: Springer, Jan., ch. A Real-Coded Genetic Algorithm Using the Unimodal Normal Distribution Crossover, pp 213–237 (2003)

[15] Poojari CA, Varghese B Genetic algorithm based technique for solving chance constrained problems. Eur. J. Oper. Res. 185(3):1128-1154 (2008)

[16] Price KV, Storn R, Lampinen JA Diferential Evolution: A Practical Approach To Global Optimization. Springer Berlin, Heidelberg (2005)

[17] Tsutsui S., Yamamura M., and Higuchi T., "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in Proc. Genetic Evol. Comput. Conf. (GECCO'99), pp. 657–664. (1999)