A New Mutation Operator for the Elitism-Based Compact Genetic Algorithm

Rafael R. Silva, Heitor S. Lopes*, and Carlos R. Erig Lima

Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR),
Av. 7 de setembro, 3165 80230-901, Curitiba (PR), Brazil
rafael.rsi@gmail.com, hslopes@pesquisador.cnpq.br, erig@utfpr.edu.br

Abstract. A Compact Genetic Algorithm (CGA) is a genetic algorithm specially devised to meet the tight restrictions of hardware-based implementations. We propose a new mutation operator for an elitism-based CGA. The performance of this algorithm, named emCGA, was tested using a set of algebraic functions for optimization. The optimal mutation rate found for high-dimensionality functions is around 0.5%, and the low the dimension of the problem, the less sensitive is emCGA to the mutation rate. The emCGA was compared with other two similar algorithms and demonstrated better tradeoff between quality of solutions and convergence speed. It also achieved such results with smaller population sizes than the other algorithms.

1 Introduction

Since long ago, Genetic Algorithms (GA) have been used as efficient tools for optimization problems, not only in Computer Science, but also in Engineering [4]. For most applications, GAs are implemented in software running on general-purpose processors. However, for applications that require the algorithm to run in real-time, hardware-based implementations are more adequate. Reconfigurable logic can be used for such implementations, by using high-performance FPGA (Field Programmable Gate Array) devices [2]. Even the most advanced devices have limited resources, specially regarding available memory. The Compact Genetic Algorithm (CGA) was devised to meet tight requirements of hardware-based implementations. For instance, a CGA represents a population of individuals by using a single probability vector, thus reducing significantly the amount of memory needed.

The CGA has been sparsely explored in the recent literature. Therefore, there is much room for theoretical development and research that can improve its efficiency. In this work we present a new mutation operator for an elitism-based CGA, capable of better controlling the selective pressure and improving the quality of solutions found. This is achieved without significant decrement of the convergence speed of the algorithm. Following the terminology previously

* This work was partially supported by the Brazilian National Research Council – CNPq, under research grant no. 305720/2004-0 to H.S. Lopes.

B. Beliczynski et al. (Eds.): ICANNGA 2007, Part I, LNCS 4431, pp. 159–166, 2007.

used by other authors, the proposed algorithm is named "Elitism with Mutation Compact Genetic Algorithm", or, in short, emCGA. The performance of the new operator is analyzed by means of computational simulations. We also perform simulations comparing the proposed algorithm with other algorithms proposed in the literature.

This paper is divided as follows. In Sect. 2 a formal description of the CGA is presented, with special focus on those works that use elitism to minimize the perturbations caused by the crossover operator in a CGA. The emCGA is presented in details in Sect. 3. Section 4 presents the results of the computational simulations. Finally, Sect. 5 presents the conclusions of the work.

2 The Compact Genetic Algorithm

The Compact Genetic Algorithm was first proposed by [5]. This algorithm uses a random-walk approach to represent a conventional genetic algorithm in a compact way. This technique is a stochastic process that formalizes successive steps towards a random direction. CGA simulates random-walks for each bit in the chromosome.

In short, in the CGA, individuals are generated randomly based on a probability vector and, at each generation, a tournament among individuals takes place. The probability vector is then updated towards the tournament winner. The elements of this vector represents the probability that each bit in the individual's chromosome to be either 0 or 1. The population of individuals is, therefore, represented compactly and it converges when all the elements of the probability vector reach either 0% or 100%. Since the probability vector represents itself the whole population, the amount of memory necessary to hold the population is much smaller when compared to a conventional GA. The memory resources needed by a conventional GA are estimated in $N \times L$ bits, where N is the population size and L is the number of bits of the chromosome. For a CGA, the elements of the probability vector are quantized with resolution $1/N$. Therefore, the amount of memory resources needed by a CGA falls down to only $L \times log_2(N)$. This feature of CGA makes it an appealing alternative for hardware implementations, rather than the conventional GA.

In the CGA, the selection phase is called random generation of chromosomes, or simply, generation, and the reproduction phase is known as updating the probability vector, or simply, updating. In a conventional GA, the Darwinian principle of survival of the fittest is most evidenced in the selection phase, where high-fitted individuals have more chance to survive and spread their genetic material. In a CGA, this phase is embedded in the generation. Recalling that the probability vector is updated towards the direction of the tournament winner at each generation, the chromosome will tend to retain the genetic information of the best individuals, thus influencing the upcoming generations.

Reproduction, the other phase of the algorithm, is typically represented in a conventional GA by the application of genetic operators, mainly crossover and mutation. Crossover is an operator capable of recombining parts of parental

chromosomes and generating two new offsprings. Throughout generations, the repeated applications of crossover leads to a decorrelation of the genes in a population. According to [5], in such decorrelated state, a simple probability vector can be a more compact and suitable representation for the whole population of individuals. In this case, updating the probability vector is analog to the crossover operator action in the conventional GA. Most of the CGAs reported in the literature do not use the mutation operator.

The typical stopping criterion for a conventional GA is when the number of generations achieves a predefined number of generations. In a CGA, the stopping criteria cannot be other than the convergence of the probability vector.

Crossover can generate critical perturbations in problems with high-order building blocks [4]. Harik and colleagues [5] demonstrated that such perturbations can be minimized when the selective pressure is increased. Further, [1] showed that elitism is even more suited for this purpose. This fact has motivated the emergence of several elitism-based CGAs in recent years [1], [3].

When elitism is used in a conventional GA, one or more top-fitted individuals are copied with no change to the next generation. In the original CGA, at each generation, two individuals are randomly generated using the probability vector. These two individuals compete in a tournament. In an elitism-based CGA, only one individual is randomly generated and the other competitor of the tournament is a copy of the best individual of the current generation. Examples of elitism-based CGAs are: persistent elitist CGA (pe-CGA) and nonpersistent elitist CGA (neCGA) [1], and CGA with elitism and mutation (mCGA) [3].

Depending on the nature of the problem dealt by a CGA, the use of elitism can induce a too high selective pressure. Hence, some technique for controlling selective pressure is necessary. Inheritance control of the best individuals was proposed by [1] in the neCGA. Also, a mutation operator was proposed by [3] in the mCGA. Both works present important improvement in the quality of solutions obtained by the algorithm.

Two features of CGA make it appealing for hardware implementations: the binary representation of solutions and its small demand of memory resources. However, a CGA does not explore all the typical features of a conventional GA and, thus, a more limited performance is expected. Therefore, this fact suggests that, by using specific features of a conventional GA in a CGA it is fair to believe that a better performance can be achieved, as shown later in this work.

3 The emCGA

The previously mentioned works (neCGA and mCGA) perform better than the original CGA [1]. This is obtained by minimizing the perturbation provoked by the crossover operator, and thus, leading to a significant improvement in the quality of solutions. In this work we propose a new mutation operator, aimed at improving even more the quality of obtained solutions but, also, keeping a reasonable convergence speed. This operator allows a more efficient control of the selective pressure, adjusting the population diversity as the consequence of

the manipulation of the probability vector. Comparing the proposed mutation operator with that of mCGA [3], the new operator decreases the number of tournaments per generation and, consequently, the total number of fitness evaluations per generation. The consequence is a significant improvement in the convergence speed of the algorithm. On the other hand, the mutation in the mCGA works on the best individual of the current generation. This method requests a new tournament and, consequently, one more fitness evaluation.

The new CGA resulting from the use of the proposed mutation is named emCGA (elitism with mutation CGA). Basically, the proposed mutation operator changes the random generation phase, by modifying the chromosome generated by the probability vector.

4 Computational Experiments and Results

Mathematical functions have been frequently used as a benchmark for optimization algorithms, including GAs. In this work we selected a suite of complex mathematical functions defined in a n-dimensional space ($\mathcal{R}^n$). This suite includes the following problems, with respective dimensions: Sphere(15), Circle(2), Shaffer-F6(2), Griewank(15), Discrete-1 (15) and Discrete-2(15). Such problems were chosen because some of them were used to evaluate performance of evolutionary computation algorithms [6].

For the Sphere problem (Sect. 4.1) we used a population of 255 individuals, whereas for the remaining problems (Sect. 4.2) we analyzed the performance of the algorithm using 31, 63, 127, 255, 511, 1023 and 2047 individuals. These values were chosen based on the current literature. The stopping criterion is not based on a predefined number of generations, but in the convergence of the probability vector. This approach leads to a variable number of generations, and this in an observable parameter in these experiments.

All functions represent minimization problems, and the optimal solution is a multidimensional null vector. For each problem, a different representation was used for the elements of the vectors applied to the functions. For the Circle problem we used 16 bits to represent the range from -32.767 to 32.768. For Discrete-1 and Discrete-2, we used 16-bit integers, ranging from -32768 to 32767. For the remaining problems, we used 18 bits to represent the range -131.071 to 131.072.

For each problem, 100 independent runs were done. Values reported in Sect. 4.1 and Sect. sec-compare are: the average best fitness value and the average number of generations until the convergence of the probability vector.

4.1 Analysis of the Mutation Parameter in the emCGA

The first group of experiments aimed at finding the best value for the mutation rate in the emCGA. The higher the dimension of the problem, the larger the chromosome size and the more complex the problem becomes. Therefore, this experiment will evaluate how the dimension of the problem affects the best mutation

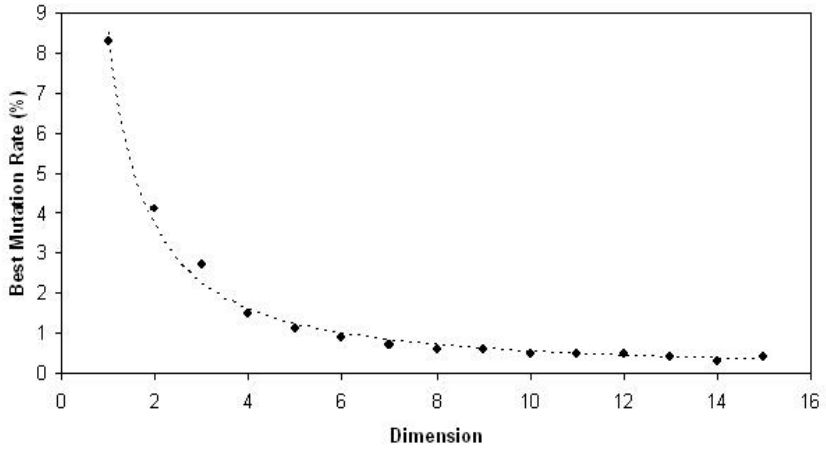

Fig. 1. Best mutation rate as a function of the dimension, for the Sphere problem

rate. For this experiment we used the Sphere problem up to 15 dimensions, and mutation rate ranging between 0 and 15%, with resolution of 0.1%.

In this experiment we observed that the best value for the mutation rate is a function of the dimension of the problem. The best mutation rate is inversely proportional to dimension. That is, the mutation rate that leads to the smallest fitness value depends on the dimension of the problem. It was also observed that exists a range of mutation rates for which the best values of fitness are found, resembling a plateau. This range narrows as the dimension grows. Also, at the extremes of that plateau, the behavior is exponential. As the dimension decreases, the problem becomes less complex and, therefore, it also becomes less sensitive and more tolerant to different values of the mutation rate. Figure 1 shows in a simplified way the behavior of the best mutation rate for all dimensions of the problem tested. It can be observed in this figure a nonlinear relationship between the dimension and the best mutation rate. For high dimensions of the problem, the mutation rate that leads the algorithm to the best performance (regarding fitness values) is around 0.5%. On the other hand, when the dimension of the problem is low, the value for the best mutation rate tends to grow exponentially.

The same experiments done for the Sphere problem were repeated for the other problems. Results obtained (not shown here) were qualitatively equivalent

Table 1. Best mutation rates for the problems tested

Dimension	Problem	Best mutation rate
2	Circle, Shaffer-F6	8.5%
15	Sphere, Griewank, Discrete-1, Discrete-2	0.5%

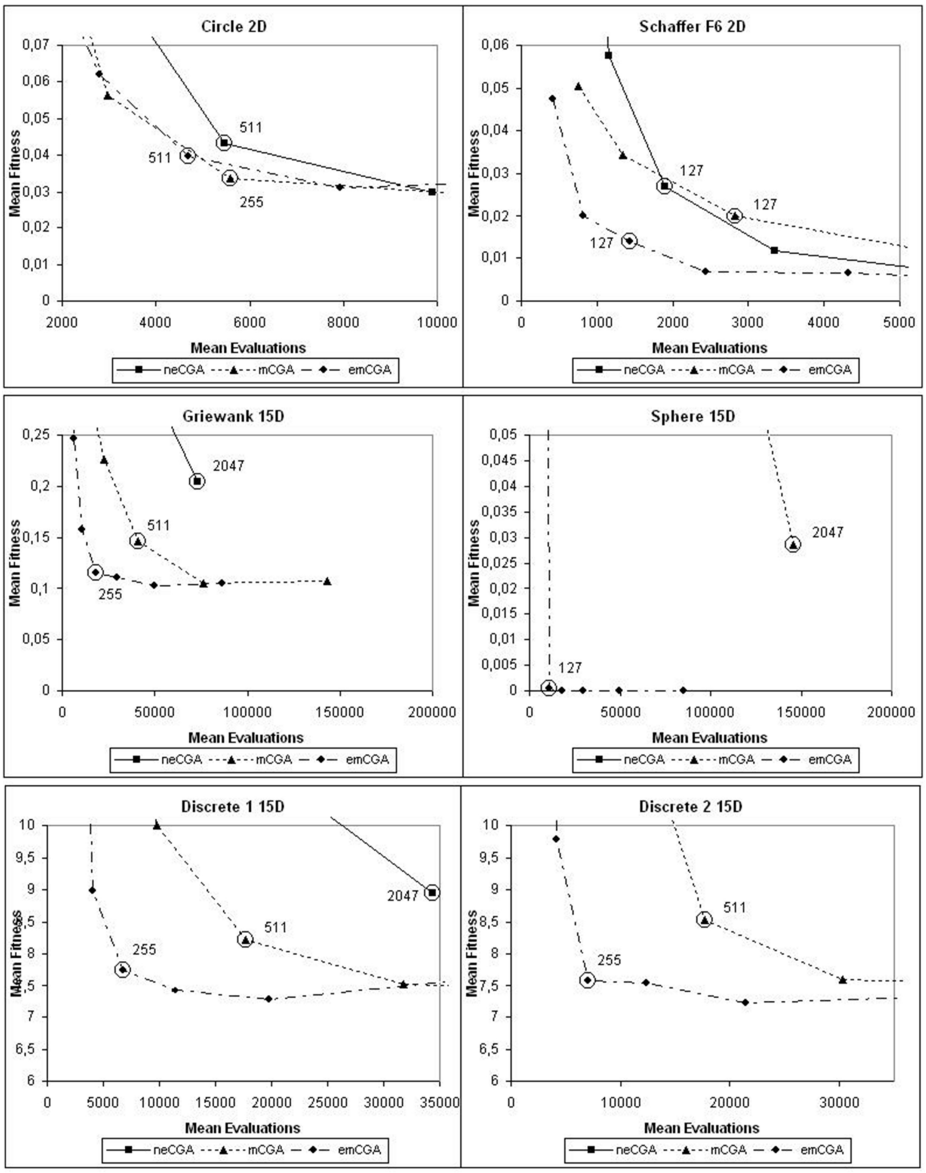

Fig. 2. Comparative analysis in the Pareto front of three CGAs

to those obtained with the Sphere problem. That is, even if we consider different problems, with different dimensions, the behavior is similar to that shown in Fig. 1. This fact suggests that the best mutation rate for a given problem is really dependent on the problem and its dimension. According to these experiments, the best mutation rate found is shown in Table 1.

4.2 Comparison of emCGA With Other Algorithms

Here we compare the performance of the proposed emCGA with other algorithms published in the recent literature, the mCGA [3] and the neCGA [1]. The parameters used in these algorithms are those suggested by respective authors in their publications. In particular, for the mCGA, the mutation rate used was 5% and, for the neCGA the inheritance length was set to 10% of the population size. For the emCGA the mutation rate was set according to the problem, that is, 8.5% for Circle and Shaffer-F6, and 0.5% for the remaining problems.

Considering the nature and purpose of a CGA, for this type of analysis it is important not only the quality of the solution (that is, the best fitness), but also, the number of fitness evaluations to achieve such quality (that is, the number of generations). For a CGA it is important to obtain a good solution in as few as possible generations. Since both objectives are contradictory, the analysis in the Pareto plane can be more useful instead of analyzing both objectives separately. In the Pareto plane, the best tradeoff between the two objectives is that closest to the origin of coordinates system.

Therefore, we used the Pareto plane to compare the behavior of the three algorithms for the test problems (excluding the Sphere problem, used in Sect. 4.1), regarding the best fitness and the number of evaluations. This comparison is shown in Fig. 2. All values in these plots are the average of 100 independent runs. Recall that 100 independent runs were done for each of the several population sizes and for each algorithm. In the plots we show only the results regarding the population size that yielded the best performance for the algorithms. The closest point to the origin is highlighted in the plot with the corresponding population size used. Some of the algorithms had a very poor performance. Consequently its value for fitness and/or number of evaluations was so high that the corresponding point could not fit the scale of the plot.

5 Conclusions

In this work we proposed a new mutation operator for an elitism-based CGA. We analyzed the performance of the proposed operator on several test problems, for different mutation rates and problem dimensions. We also compared the performance of the proposed algorithm with other two recently published algorithms.

The performance analysis of the proposed emCGA reveals that using specific mutation rates for each problem (and each dimensionality) leads to better performance compared with a fixed rate for any problem. Also, we observed a nonlinear relationship between the dimension of the problem and the mutation rate that gives best performance (regarding average fitness). For high-dimensionality problems, a mutation rate of 0.5% seems to be appropriated. For low-dimensionality problems, higher values give better results. However, the low the dimension of the problem, the less sensitive it is to the mutation rate.

Comparing the proposed emCGA with mCGA and neCGA, we observed that, except for the Circle problem, emCGA achieved, at the same time, better

convergence speed and better fitness value than the other algorithms. The Circle problem is the simplest one of the suite, furthermore it is a low-dimensionality problem. Possibly, these are the reasons why emCGA achieved a performance quite similar to mCGA, however with a better convergence speed.

For the high-dimensionality problems (and also for Shaffer-F6 problem), emCGA found solutions using smaller populations than those used by the other algorithms. This is an important issue since such algorithms were proposed for hardware implementations, where memory resources are limited.

Figure 2 shows that emCGA is the algorithm that has the better tradeoff between quality of solution and convergence speed. This fact suggests that emCGA is an interesting alternative for implementations that require compact genetic algorithms.

Future work will focus on evaluating the overall limits of the proposed approach, by implementing emCGA in a FPGA device do deal with a real-world difficult problem such as [7], [8].

References

1. Ahn, C., Ramakrishna, R.: Elitism-based compact genetic algorithms. *IEEE Trans. Evolutionary Computation* **7** (2003) 367–385.
2. Becker, J., Hartenstein, R.: Configware and morphware going mainstream. *J. Systems Architecture* **49** (2003) 127–142.
3. Gallagher, J., Vigraham, S., Kramer, G.: A family of compact genetic algorithms for intrinsic evolvable hardware. *IEEE Trans. Evolutionary Computation* **8** (2004) 111–126.
4. Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, Reading (1989).
5. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. In: *Proc. IEEE Conf. on Evolutionary Computation*, (1998) pp. 523–528.
6. Krink, T., Filipic, B., Fogel, G., Thompsen, R.: Noisy optimization problems – a particular challenge for differential evolution ? In: *Proc. IEEE Conf. on Evolutionary Computation*, (2004) pp. 332-339.
7. Lopes, H.S., Moritz, G.L.: A graph-based genetic algorithm for the multiple sequence alignment problem. *Lecture Notes in Artificial Intelligence* **4029** 420–429.
8. Moritz, G.L., Jory, C., Lopes, H.S., Erig Lima, C.R.: Implementation of a parallel algorithm for pairwise alignment using reconfigurable computing. *Proc. IEEE Int. Conf. on Reconfigurable Computing and FPGAs*, (2006) pp. 99-105.