On the Use of Compact Approaches in Evolution Strategies

Anderson SERGIO; Sidartha CARVALHO; Marco COSTA REGO

ats3@cin.ufpe.br, salc@cin.ufpe.br, maqce@cin.ufpe.br
Center of Informatics, Federal University of Pernambuco, Recife, Brazil

KEYWORD

Adaptive systems; Compact evolutionary; algorithms; Evolution strategies; Estimation of distribution algorithms

ABSTRACT

Compact evolutionary algorithms have proven to be an efficient alternative for solving optimization problems in computing environments with low processing power. In this kind of solution, a probability distribution simulates the behavior of a population, thus looking for memory savings. Several compact algorithms have been proposed, including the compact genetic algorithm and compact differential evolution. This work aims to investigate the use of compact approaches in other important evolutionary algorithms: evolution strategies. This paper proposes two different approaches for compact versions of evolution strategies. Experiments were performed and the results analyzed. The results showed that, depending on the nature of problem, the use of the compact version of Evolution Strategies can be rewarding.

1. Introduction

According to IDC (International Data Corporation), sales of embedded computer systems generate more than US$1 trillion in revenue annually. This sum will double over the next four years. The Intelligent Systems, which are a part of the embedded systems, will be representing more than one fourth of the volume of all embedded systems by 2019 and will be capturing more than 75% of the revenue [IDC 2014].

In many real-world applications, an optimization problem must be solved even in situations where massive computational power is not available due to budget limitations and/or space. This is a typical situation in embedded systems like robotics and process control problems. To overcome these adversities, use of Compact Evolutionary Algorithms (cEAs) was proposed.

A major constraint of embedded systems is the need of operation with low power consumption. The increase in the technology of processors, disks, memory, and communication has highlighted that the capacity of batteries is not following the growth of other technologies used in embedded systems [Paradiso, J. *et al*. 2005].

With less memory space needed, the embedded system can load more data applications than used to. About power consumption, when we have less memory addressed to an application, the probability to have memory miss is lower and the change of no need to save data in hard disk is higher. This can imply less power consumption. Carroll [Carroll, A. *et al*. 2010] showed that in some scenarios, memory RAM can overcome CPU in power consumption.

To help optimizations problems to use less memory, Compact Evolutionary Algorithms can be used. They belong to the category of Estimation of Distribution Algorithms (EDAs) [Larrañaga, J. *et al*. 2002]. In this class of algorithms, a population is not stored and processed. To continue the optimization process, a static representation of individuals is used. This feature allows a smaller number of parameters stored in

the memory, so that this implementation requires less storage space when compared to standard evolutionary algorithms. Several cEAs have been proposed, including the Compact Genetic Algorithm (cGA) [Harik, J. *et al.* 1999] and the Compact Differential Evolution (cDE) [Mininno, J. *et al.* 2011]. Although Evolution Strategy (ES) solutions are widely used, no compact version for these algorithms has been proposed. This study proposes two versions of compact development strategies, c(1+1)-ES and c(μ, λ)-ES. The main objective is to investigate the use of compact approaches of this classic algorithm through the analysis of results obtained in benchmark databases.
This paper is organized as follows: in session two, an overview of the compact algorithms and evolution strategies used as the basis for the approaches proposed in this paper is presented. Session three describes the algorithms c(1+1)-ES and c(μ, λ)-ES and discusses its operating principles and details. Session four shows the numerical results and is divided into two parts: the results of the algorithm c(1+1)-ES when compared with no compact version of it, and the results of c(μ, λ)-ES, with their proper comparisons. The conclusions and suggestions for future work are in section five.

2. Background

2.1.Compact Algorithms

Compact algorithms are estimates of distribution algorithms that mimic the behavior of a population base by means of a probable representation of the population of candidate solutions. These algorithms have a similar behavior in respect to the algorithms based on the population, but require a much smaller memory [Mininno, J. *et al.* 2011].
The first cEA proposed was the Compact Genetic Algorithm (cGA), introduced by [Harik, J. *et al.* 1999]. The cGA simulates the behavior of a Genetic Algorithm (GA) with binary encoding [Holland, J. *et al.* 1975]. The work proposed by [Harik, J. *et al.* 1999] demonstrate that performance of cGA is almost good as the GA one. As expected, the main advantage of a cGA with respect to a standard GA is the memory savings.
In the cGA, a binary vector of length *n* is generated randomly with probability of 0.5 for each gene that can be of values 0 or 1. This vector that describes the probabilities initialized with *n* values equal to 0.5 is called Probability Vector (PV).
Through the PV, two individuals are chosen and their fitness values are calculated. The solution characterized by higher performance changes PV based on a parameter called virtual N_p, which represents the population. More specifically, if the winner solution in their gene *i* position is of value 1, while the losing solution value is 0 in the same position, the probability value at position *i* of the PV is increased by $\frac{1}{Np}$. Otherwise, PV is reduced by $\frac{1}{Np}$. If genes at *i* position exhibit the same value for both, for the winner and for the loser, the probability of PV at position *i* is not modified.
This study is based on the compact genetic algorithm for real values (cGAr) that was introduced by [Mininno, J. *et al.* 2011]. The cGAr is a compact algorithm inspired by cGA exporting compact logic to a domain of real values, obtaining an optimization algorithm with a high performance, despite the limited amount of memory. In cGAr the PV is not a vector, but a $n \times 2$ matrix:

$$PV^t = [\mu^t, \sigma^t] \quad (1)$$

Where μ and σ are, respectively, the vectors containing, for each design variable, mean and standard deviation of a Gauss' probability distribution function truncated within the range [-1,1]. At the beginning of the optimization process, for each variable i, $\mu^1[i] = 0$ and $\sigma^1[i] = \lambda$, the λ is a large positive constant (λ = 10). The initialization values of σ[i] are made to simulate a uniform distribution. Subsequently, an individual is chosen as elite. A new individual is generated and compared with the elite. cGAr solution also changes the bias of PV as in cGA. The update rule for each element of μ is given by:

$$\mu^{t+1}[i] = \mu^t[i] + \frac{1}{N_p}(winer[i] - loser[i]) \quad (2)$$

Where N_p is the population size. The update rule for σ is:

$$(\sigma^{t+1}[i])^2 = (\sigma^t[i])^2 + (\mu^t[i])^2 - (\mu^{t+1}[i])^2 + \frac{1}{N_p}(winner[i]^2 - loser[i]^2) \tag{3}$$

μ is the mean of PV, σ is the standard deviation of PV, winner is the step of mutation that generated the best individual, and loser is the step of mutation that generated the worst individual.
Other compact versions of genetic algorithms can be found in literature. The *Extended Compact Genetic Algorithm* (ecGA) was proposed by [Sastry, J. *et al*. 2000]. In ecGA, the probability distribution used is a class of models known as marginal probability model product. A hybrid version of ecGA with the Nelder-Mead algorithm is proposed by [Sastry, J. *et al.* 2001] and the study of its scalability is shown by [SASTRY, J. *et al*. 2007]. Additionally, a memetic variant was presented in [BARAGLIA, J. *et al*. 2001], with the aim of increasing the convergence of the algorithm in the presence of a large number of dimensions. The various compact versions of genetic algorithms have been used in practical applications implemented in hardware, in which the computer resources can be smaller [Aporntewan, J. *et al.* 2001] [Jewajinda, J. *et al*. 2008].
With similar ideas to compact genetic algorithms, a compact version of the differential evolution called cDE is presented by [Mininno, J. *et al.* 2011]. In differential evolution, a set of vector parameters is randomly generated early in the process, covering the entire search space. The algorithm then recombines these vectors in order to minimize or maximize an objective function. The compact version of the differential evolution was applied to a case study related to the online training, a neural network implemented directly on a microcontroller. The numerical and experimental results confirmed the efficiency of the proposed algorithm.
A study that uses another type of approach is [Neri, F. *et al.* 2013]. This work proposes a compact Particle Swarm Optimization (cPSO) algorithm based on PSO principles.

2.2. Evolution Strategies

Evolution Strategies (ES) are a subclass of direct search algorithms based on nature and belonging to the class of evolutionary algorithms [Eiben, J. *et al.* 2003]. ES use mutation, recombination and selection applied to a population of individuals containing candidate solutions in order to find better solutions iteratively. The algorithm was first proposed in 1974 [Shwefel, J. *et al.* 1974].
An evolutionary algorithm is usually described according to characteristics such as representation of individuals, recombination and mutation types and parent selection.
ES are typically used for continuous optimization problems. The standard representation of individuals is given by a real vector $x_1, ..., x_n$, where each x_i is a floating point variable. However, most modern ES algorithms use this vector only as a part of the genotype. Individuals may contain some strategy parameters influencing directly the mutation operation. Thus, the genotype can be completely defined by $(x_1, ..., x_{n,}, \sigma_1, ..., \sigma_{n\sigma,} \alpha_1, ..., \alpha_{n\alpha})$.
The mutation operation is based on a normal distribution that requires the parameters: mean and standard deviation. In general, the engine uses a value for adding a noise, formed from this distribution. This value is called the step size of mutation, a part of the genotype to evolve. The operation of mutation can be accomplished in several ways. Below is a description of the mechanisms used in this work.
In non-correlated with one step, a single value per individual is calculated by multiplying this value by a lognormal distribution mutation. Below, the update equations:

$$\sigma' = \sigma * \exp(\tau * N(0,1)) \tag{4}$$

$$x_i' = x_i + \sigma' * N(0,1) \tag{5}$$

σ is the mutation step, x_i a gene of the genotype and τ a learning rate, usually proportional to $1/n^{1/2}$. The mutation step is the same in each direction.
In non-correlated mutation with n steps, there is a mutation step for each gene. The equations are as follows:

$$\sigma_i' = \sigma_i * \exp(\tau' * N(0,1) + \tau * N_i(0,1)) \tag{6}$$

$$x_i' = x_i + \sigma_i' * N_i(0,1) \tag{7}$$

In this case, $\tau'$is the global learning rate, the same for all individuals. τ is the specific learning rate, allowing the mutation in many directions. The first one is proportional to $1/(2n)^{1/2}$ and the second one to $1/(2n^{1/2})^{1/2}$.
In the recombination of evolution strategies, two parents generate a child. Acting for position, the recombination may be intermediate or discrete. In the intermediate recombination, the children are formed by averaging the values of the parents. In the discrete recombination, the son is generated by selecting the gene of one parent, from a probability distribution. The strategy parameters (mutation) are typically recombined in an intermediate manner, while the objective parameters (genes) are recombined discretely [Eiben, J. *et al.* 2003]. Recombination also differs with respect to the parents used: two parents previously selected to generate a child or two different parents for each position.
The parents are selected from a random uniform distribution. Thus, each individual has the same probability of being selected, regardless of their fitness.
The classic versions of ES coexist with more modern versions. One of these newer algorithms is the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [Hansen, J. *et al.* 2006]. In this algorithm, the dependencies between variables in the probability distribution are represented by a covariance matrix. The CMA-ES takes this matrix into account when performing the adjustment operations.
The Evolution Strategies are stronger at exploiting, to find the best local, where in Particle Swarm Optimization we have more exploration, stronger at global search. A hybrid algorithm is designed mainly to take the advantage of the balance between exploring and exploit. A hybrid version of Evolution Strategy with Particle Swarm Optimization was proposed by [Matsumura, Y. *et al.* 2013] and was applied to a dinosaur's gait generation problem. Experimental results by Matsumura *et al* indicate that hybrid algorithm converges faster in the beginning phase and finds maximum fitness.

3. Compact Evolution Strategies

In the following session, the two proposed versions for compact evolution strategies will be presented. Basically, they differ in the representation of the population.

3.1. C(1+1)-ES algorithm

Since the evolution strategies were first proposed, several approaches were investigated by varying parameters such as the type of mutation and the organization of the population. Another important change is the mechanism of selection of the survivors. In general, these different mechanisms are known as (μ, λ) and ($\mu + \lambda$). μ is the population size in a given iteration (the parents) and λ the offspring size. The first mechanism indicates that the population of the next iteration will not take into account the parents, and is therefore a non-elitist approach. In the second case, the next iteration population will be composed of both, the parents and the offspring.
In the approach known as (1+1)-ES, the population is made up of only two individuals who are being adapted along the algorithm execution. It suggests a compact version for this approach and does not provide memory saving therefore. But the c(1+1)-ES (compact version of (1+1)-ES) was proposed to investigate the use of compact approaches in a simpler algorithm evolution strategy, checking that its performance is acceptable.
The proposed algorithm is performed following the sequence of steps in Figure 1.

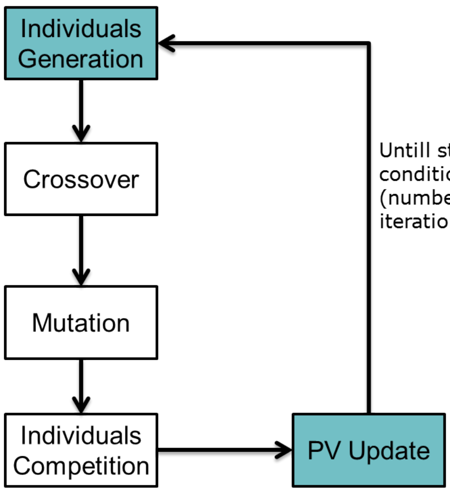

Fig. 1. c(1+1)-ES algorithm

Differently from conventional strategy, (1+1)-ES, the initialization of individuals with this approach uses a Probability Vector (PV). This PV stores the mean and standard deviation that is used to generate the values of the genes of the individuals. This approach uses the PV to induce individuals to be generated in a promising area of the search space. At the end of each generation, the PV is updated according to the equations described in the rcGA algorithm (equations 2 and 3).
In conventional (1+1)-ES, the individuals are randomly generated by a normal distribution of average 0 and standard deviation 1, leaving at total random the value of the gene. By comparing these approaches, one can see that the compact version is more likely to generate more fit individuals. Over time, the generation of individuals is directed to a promising area and does not move randomly.
The two approaches differ as mentioned above, in generation of individuals and use of PV. Other operators such as mutation, recombination and the evaluation function of fitness to each individual are equal.

3.2. c(μ, λ)-ES algorithm

Eiben and Smith [Eiben, J. *et al*. 2003] indicate that the mechanism that is not elitist in survivors' selection is the most widely used and offers the best results. According to these authors, (μ, λ) should be preferable because it's better in leave local optimum and move to promising regions, since it discards all parents and does not preserve outdated solutions. Additionally, by using the mechanism (μ + λ), bad mutation values may persist in the population for a long time.
Given these issues, this paper also proposes an investigation of use of compact approaches in (μ, λ)-ES, this approach will be called c(μ, λ)-ES. The algorithm c(μ, λ)-ES is inspired in the rcGA and cDE to give to (μ, λ)-ES a compact treatment. Figure 2 shows a general description of c(μ, λ)-ES.
First the PV is initialized with a mean and standard deviation. The average is generated from a uniform probability distribution between a lower and upper limit for each goal function. These limits are defined empirically. The standard deviation is generated from a normal distribution N(0, 1). The elite individual is also generated from a uniform distribution between the lower and upper limits of the objective function. The same logic is applied to the PV of mutation steps and elite mutation. The difference is in the limits. The mutation is given by -1 to 1 and in c(μ, λ)-ES the mutation approach is not correlated with the *n* steps.

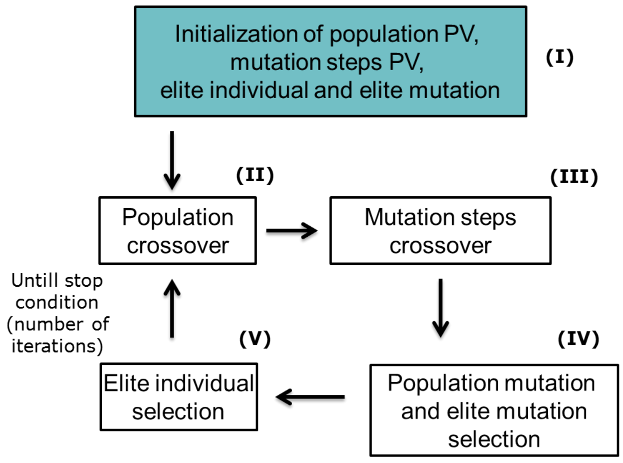
Initialization of population PV, mutation steps PV, elite individual and elite mutation
(I)
(II)
Population crossover
(III)
Mutation steps crossover
Untill stop condition (number of iterations)
(V)
Elite individual selection
(IV)
Population mutation and elite mutation selection

Fig. 2. c(μ, λ)-ES algorithm

After initialization of these vectors, the iterations are initiated corresponding to the number of generations. In each generation, the first step is to perform population recombination. Following the characteristics of rcGA and cDE, two parents are sampled from the population PV. The current individual is generated from the combination of these two parents. The combination may be intermediary or discrete. Then, similar logic is used for the generation of the current vector changes. Two parents are sampled from the PV mutations and recombined with intermediate or discrete approaches.
At this generation point, the algorithm takes into memory a current individual and a vector of current mutation steps. Following the standard update rules of Evolution Strategies, the individual is mutated by the mutation current vector changes and the elite mutations vector. The current individual is updated by the best individual generated from these two mutations, and this information is used to update the PV mutations. The update is similar to what occurs in c(1 + 1)-ES, given by equations 1 and 2.
After changing the current individual and the update of PV mutation steps, there is a competition between this individual and the elite individual. The result of this competition influences the population of PV update, again according to equations 1 and 2. All these operations are performed until the maximum number of generations is reached.

4. Results

To validate the results, the proposed algorithms were compared with respective non-compact versions. The c(1+1)-ES was compared with the (1+1)-ES and c(μ, λ)-ES was compared with the (μ, λ)-ES. The comparison was made by using the best fitness values achieved by algorithms in 10 functions. The functions used were, each one with dimension n = 2: Beale, Booth, Dixon, Griewank, Hump, Levy, Matyas, Rastrigin, Rosenbrock, Sphere. In all cases, the fitness is the function value for each problem. All optimizations functions are in [Surjanovic, S. *et al*. 2015].
To confer statistical validity to the comparisons, the unpaired Student's *t* test [Efron 1969] was used after 30 runs, with a confidence level of 95%. In tables 3 and 6, "+" indicates the case when the compact algorithm has overcome the traditional version at given function; an "=" indicates that the difference between the results is statistically irrelevant and, therefore, the algorithms have the same performance; a "-" indicates that the traditional version surpassed the compact approach.
In addition, not only performance is considered, we also investigated memory consumption in proposed approaches. In Table 1 we can see that for (μ, λ)-ES our proposed c(μ, λ)-ES can reduce memory usage by half, but in contrast (1+1)-ES have the same memory usage as c(1+1)-ES with similar or even better performance for some function tests that we investigated.

Algorithm	Memory Slots
(1+1)-ES	2
c(1+1)-ES	2
(μ, λ)-ES	$2N_p$
c(μ, λ)-ES	N_p

Table 1. Memory consumption

4.1.c(1+1)-ES algorithm

This session presents the results achieved by the algorithm c(1+1)-ES, the comparison with the original algorithm (1+1)-ES and the statistical test results to verify if the results are statistically equivalent. The compact algorithm was equivalent to conventional (1+1)–ES.
An adapted strategy of not correlated mutation with mutation step 1 was used. In the compact approach, each individual gene has a σ representing the mutation step for that gene. The conventional approach uses a σ for all the same individual genes. An intermediate population recombination, an adapted mutation version of not correlated mutation with one mutation step and 100 generations steps were used for both algorithms. The parameters of each algorithm are presented in Table 2.

Algorithm	Parameter Name	Value
(1+1)-ES	μ	1
(1+1)-ES	λ	1
c(1+1)-ES	μ	1
c(1+1)-ES	λ	1

Table 2. Parameters of the algorithms

Table 3 shows mean and standard deviations of the best fitnesses achieved by individuals. One can see that the values are very similar in the two approaches.
The third column of Table 3 shows the validation of the results using the Student's *t* test. One can see that in half of the functions the compact version of the algorithm was superior. In other functions, performance was equivalent, even if the absolute result of the compact version was better in all functions. Confirming that the first compact approach at least has equivalent performance to the conventional version, the next section shows the results for the compact version of a more sophisticated evolution strategy algorithm.

4.2.c(μ, λ)-ES algorithm

This session presents the results achieved by c(μ, λ)-ES and a comparison with the traditional version of the algorithm. For the traditional version of the algorithm, two sets of parameters were tested. The following parameters were selected according to indications of previous work or empirically. An discrete population recombination, an intermediate mutation steps recombination and 100 generations steps were used for (μ, λ)-ES-1, (μ, λ)-ES-2 and c(μ, λ)-ES algorithms. The lower limit for mutation step was -1 and the upper limit was 1 for all algorithms. The parameters of each algorithm are presented in Table 4. The number of generations and the upper and lower limits of the mutation steps were defined empirically. For the type of population and mutation steps recombination, the indications of [Eiben, J. *et al*. 2003] were used. μ and λ were varied in order to justify a more fair comparison, as explained in the following paragraphs.

Function	(1+1)-ES	c(1+1)-ES	*Student's t Test*
Beale	2.402333 (4.499564)	0.692157 (0.604413)	+
Booth	3.024351 (6.463611)	0.006861 (1.255636)	+
Dixon	9.390236 (31.056936)	0.102423 (0.083324)	=
Griewank	0.025824 (0.034090)	0.014437 (0.018447)	=
Hump	89.627260 (302.804438)	0.171184 (0.226865)	=
Levy	1.812646 (2.706391)	0.013224 (0.017839)	+
Matyas	0.142203 (0.228759)	0.008207 (0.010866)	+
Rastrigin	19.088306 (19.322502)	2.427251 (1.640313)	+
Rosenbrock	49.222358 (219.097925)	2.154793 (6.462810)	=
Sphere	3.336067 (12.948437)	0.038378 (0.083356)	=

Table 3. Mean of best fitnesses

Algorithm	Parameter Name	Value
(μ, λ)-ES-1	μ	10
(μ, λ)-ES-1	λ	10
(μ, λ)-ES-2	μ	10
(μ, λ)-ES-2	λ	20

Table 4. Parameters of the algorithms

Table 5 shows the mean of best fitnesses achieved by the algorithms in each tested function, in 30 runs. In brackets, standard deviation values are shown. Table 6 shows the statistical comparison between (μ, λ)-ES-1 and c(μ, λ)-ES and, similarly, a comparison between (μ, λ)-ES-2 and c(μ, λ)-ES.

Function	(μ, λ)-ES-1	(μ, λ)-ES-2	c(μ, λ)-ES
Beale	0.837853 (1.421673)	0.063001 (0.218540)	0.685472 (1.703286)
Booth	13.085368 (19.615617)	0.000698 (0.000756)	0.029533 (0.023305)
Dixon	1.526911 (4.876232)	0.000639 (0.000761)	0.017207 (0.019740)
Griewank	0.723866 (0.465882)	0.255029 (0.264967)	0.660719 (0.668455)
Hump	0.964940 (1.860968)	0.000641 (0.000499)	0.052358 (0.145693)
Levy	0.056244 (0.146464)	0.005408 (0.018034)	0.114306 (0.430628)
Matyas	0.389642 (0.494220)	0.000033 (0.000048)	0.001468 (0.002028)
Rastrigin	3.200705 (2.371614)	0.175241 (0.186175)	2.321291 (1.428543)
Rosenbrock	0.849357 (4.434919)	0.033554 (0.066951)	1.174561 (2.063353)

Sphere	0.263091 (0.368627)	0.000170 (0.000170)	0.007260 (0.007272)

Table 5. Mean of best fitnesses

By analyzing the tables, one can see that the proposed compact approach outperformed traditional ES in the first set of parameters, μ = 10 and λ = 10, in half of the test functions. In the other half, the difference was statistically irrelevant. In addition to the gain in the best fitness, it should be noted that the expected gain were due to the characteristics of a compact algorithm. The memory savings are evident, since in c(μ, λ)-ES the population is stored in a vector *n* x 2, being *n* the function dimension.
However, it is well known and it has been found empirically that performance of the (μ, λ)-ES approach can improve. The most straightforward idea to improve fitness is to increase the population size. Eiben and Smith [EIBEN, J. et al. 2003] indicate that the (μ, λ) approaches of evolution strategies improve significantly when λ > μ. This is precisely the characteristic of the approach (μ, λ)-ES-2, where λ = 20 and μ = 10. As expected, the performance improved significantly and c(μ, λ)-ES could not overcome the traditional ES with this set of parameters in any of the tested functions.

Function	(μ, λ)-ES-1 vs. c(μ, λ)-ES	(μ, λ)-ES-2 vs. c(μ, λ)-ES
Beale	=	-
Booth	+	-
Dixon	+	-
Griewank	=	-
Hump	+	-
Levy	=	-
Matyas	+	-
Rastrigin	=	-
Rosenbrock	=	-
Sphere	+	-

Table 6. Student's t test for validation

Taking into account the results obtained in the tested functions, one can infer that the performance of a non-compact approach tends to be better than the compact approach when the appropriate parameters are used. This behavior is expected since the compact approaches are inclined to lose performance due to probability vector usage rather than storage of population in memory. The advantage of the compact approach, as found empirically is to save memory. But even when compared with the approach (μ, λ)-ES-1 using a population size of 10, the performance of c(μ, λ)-ES was superior.

5. Conclusions and Future Works

This work introduced the concept of compact evolution strategy and proposed two variants of algorithms, c(1+1)-ES and the c(μ, λ)-ES. Both variants do not require powerful hardware in order to exhibit a high performance. On the contrary, the proposed algorithms make use of a limited amount of memory in order to perform optimization.
The algorithm c(1+1)-ES proved to be equal to (1+1)-ES original, thus validating the use of a probability vector. c(μ, λ)-ES version, however, showed that its performance is lower when the (μ, λ)-ES uses a high λ value, i.e., generates a lot of chromosomes during playback and uses a large amount of memory. Nevertheless, on those occasions when the compact version has lower performance, there is a great saving of memory, when you have less access to memory you can save energy, showing the usefulness of compact algorithm when there are few computing resources and the solution found does not need to be the best.

A possible future work would be the implementation of the evolution strategy with correlated mutation and its comparison with other algorithms already proposed in this article. Another important point would be the application of this algorithm in a real implementation problem in hardware, reinforcing its advantage in memory economy.

6. References

Aporntewan, Chatchawit; Chongstitvatana, Prabhas. A hardware implementation of the compact genetic algorithm. In: IEEE Congress on Evolutionary Computation. 2001. pp. 624-629.

Baraglia, Ranieri; Hidalgo, Jose Ignacio ; Perego, Raffaele. A hybrid heuristic for the traveling salesman problem. Evolutionary Computation, IEEE Transactions on, v. 5, n. 6, pp. 613-622, 2001.

Carroll, Aaron; Heiser, Gernot. An Analysis of Power Consumption in a Smartphone. In: Proceedings of the USENIX Annual Technical Conference. 2010.

Efron, Bradley. Student's t-test under symmetry conditions. Journal of the American Statistical Association, v. 64, n. 328, p. 1278-1302, 1969.

Eiben, Agoston E.; Smith, James E. Introduction to evolutionary computing. Springer Science & Business Media, 2003.

Hansen, Nikolaus. The CMA evolution strategy: a comparing review. In: Towards a new evolutionary computation. Springer Berlin Heidelberg, 2006. pp. 75-102.

Harik, Georges R.; Lobo, Fernando G.; Goldberg, David E. The compact genetic algorithm. Evolutionary Computation, IEEE Transactions on, v. 3, n. 4, pp. 287-297, 1999.

Holland, John H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, 1975.

IDC. Intelligent Systems to Exceed $1 Trillion in 2019 as the Market Continues to Disrupt Traditional Industries Including Manufacturing, Energy, and Transportation. Avaliable in: < http://www.idc.com/getdoc.jsp?containerId=prUS25204914>. Accessed on 2014, October 16th.

Jewajinda, Yutana; Chongstitvatana, Prabhas. Cellular compact genetic algorithm for evolvable hardware. In: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on. IEEE, 2008. pp. 1-4.

Larrañaga, Pedro; Lozano, Jose A. (Ed.). Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science & Business Media, 2002.

Matsumura, Yoshiyuki; Sugiyama, Kiyotaka; Yasuda, Toshiyuki; Ohkura, Kazuhiro. Evolutionary and Particle Swarm Hybrid Algorithm over Cloud Computing, with an Application to Dinosaur Gait Optimization. Proceedings of the IEEE/SICE International Symposium on System Integration, Kobe, Japan, December, 2013.

Mininno, Ernesto et al. Compact differential evolution. Evolutionary Computation, IEEE Transactions on, v. 15, n. 1, pp. 32-54, 2011.

Neri, Ferrante; Mininno, Ernesto; Iacca, Giovanni. Compact Particle Swarm Optimization. Information Sciences: an International Journal, 239, p.96-121, August, 2013.

Paradiso, Joseph A.; Starner, Thad. Energy scavenging for mobile and wireless electronics. Pervasive Computing, IEEE, v. 4, n. 1, pp. 18-27, 2005.

Sastry, Kumara; Goldberg, David E. On extended compact genetic algorithm. In: Late-Breaking Paper at the Genetic and Evolutionary Computation Conference. 2000. pp. 352-359.

Sastry, Kumara; Xiao, Guanghua. Cluster Optimization Using Extended Compact Genetic Algorithm. Urbana, v. 51, p. 61801, 1989.

Sastry, Kumara; Goldberg, David E.; Johnson, D. D. Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters. Materials and Manufacturing Processes, v. 22, n. 5, pp. 570-576, 2007.

Optimizations Functions. Definition of the Optimization Functions. Available in: < http://www.sfu.ca/~ssurjano/optimization.html >. Accessed on 2015, June 10th.

Shwefel, H. P. Numerische Optimierung von Computer-Modellen. PhD Thesis, 1974. Decision support methods for global optimization.