On the Use of Compact Approaches in Evolution Strategies

Anderson Tenório Sergio, Marco Antônio Quidute Costa Rego, Sidartha Carvalho
Center of Informatics, Federal University of Pernambuco

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 little 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. And 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 third of the volume of all embedded systems by 2015 and will be capturing more than 75% of the revenue. [IDC 2015]

A major constraint of embedded systems is the need of operation with little 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].

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

Compact evolutionary algorithms 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 has been proposed for these algorithms. 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 (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

A. 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 in [HARIK, J. *et al*. 1999]. The cGA simulates the behavior of a genetic algorithm (GA) with binary encoding [HOLLAND, J. *et al*. 1975]. In [HARIK, J. *et al*. 1999] a cGA performance, almost as good as the GA one, can be seen. 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 assigning 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 top solution, the solution characterized for higher performance, changes the PV based on a parameter called virtual N_p, population. More specifically, if the winning solution in their gene *i* position is of value 1, while the loser solution is of value 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.

The version on which this study is based, the compact genetic algorithm for real values (cGAr), was introduced in [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 probability distribution of Gauss' 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 ($\lambda = 10$). 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. As the cGA in the victorious cGAr solution also changes the bias of PV. 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:

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

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. In [SASTRY, J. *et al*. 2000], the *Extended Compact Genetic Algorithm* (ecGA) was proposed. 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 can be seen in [SASTRY, J. *et al*. 2001] and the study of its scalability is

shown in [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, [MININNO, J. *et al*. 2011] presents a compact version of the differential evolution called cDE. 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.

B. 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. Next, each of these features will be discussed.

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 variable floating point. 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 subject is calculated by multiplying this value by a lognormal distribution mutation. Below, the update equations:

$$\sigma' = \sigma * \exp\big(\tau * N(0,1)\big) \quad (4)$$

$$x_i' = x_i + \sigma' * N(0,1) \quad (5)$$

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:

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

$$x_i' = x_i + \sigma_i' * N_i(0,1) \quad (7)$$

In this case, τ' 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 son. 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.

3 Compact Evolution Strategies

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

A. 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 (μ + λ). 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.

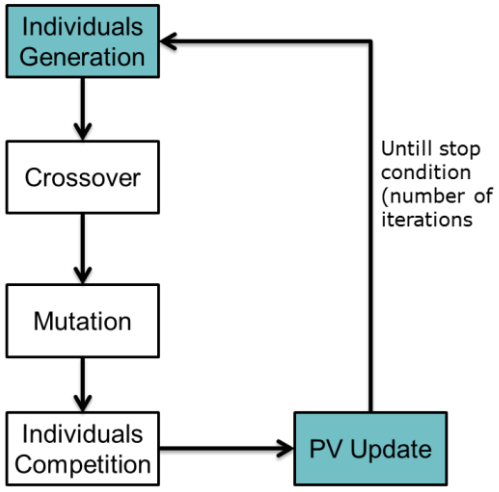

Fig. 1. c(1+1)-ES algorithm

Differently from conventional strategy, (1+1)-ES, the initialization of individuals with this approach uses a probability vector. 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 subjects 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.

B. 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 to these issues, this paper also proposes an algorithm that investigates the 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 scheme for the application of c(μ, λ)-ES. Then, the description of the algorithm.

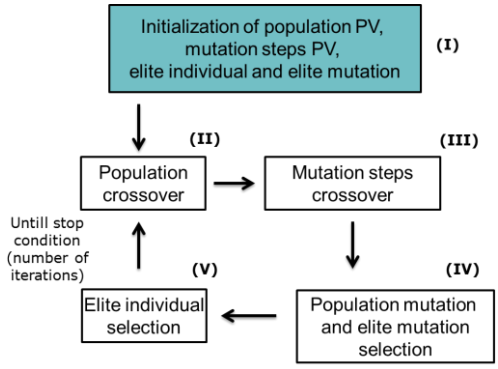

Fig. 2. c(μ, λ)-ES algorithm

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.
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 intermediate or discreetly.
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 one and two.
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 [http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar].
To confer statistical validity to the comparisons, the Wilcoxon test was used after 30 runs, with a confidence level of 95%. In the tables showing the results of statistical tests, "+" 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.

A. 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. Next, the parameters of each algorithm:

- (1+1)-ES: $\mu = 1$; $\lambda = 1$; generations = 100; population recombination = intermediate; mutation = adapted version of not correlated mutation with one mutation step.
- c(1+1)-ES: PV for individuals generations $\mu = 1$; $\lambda = 1$; generations = 100; population recombination = intermediate; mutation = adapted version of not correlated mutation with one mutation step.

Table I 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.
Table II shows the validation of the results using the Wilcoxon test. One can see that in most of the functions the algorithms are equivalent. Also, in the Rastrigin function, the compact algorithm fared better. In the Matyas function, the compact algorithm was overcome by the traditional version.
In general, the results were statistically similar. Confirming that the first compact approach has equivalent performance to the conventional version, the next section shows the results for the compact version of a more sophisticated evolution strategy algorithm.

Function	(1+1)-ES	C(1+1)-ES
Beale	0.004880	0.009391
	(0.026282)	(0.050574)
Booth	0.003839	0.006861
	(0.020678)	(0.036951)
Dixon	0.029809	0.002496
	(0.160531)	(0.013441)
Griewank	4.4862E-4	0.001237
	(0.002415)	(0.006662)
Hump	0.001659	0.004053
	(0.008939)	(0.021828)
Levy	5.8960E-4	0.001608
	(0.003175)	(0.008660)
Matyas	2.3911E-4	5.0806E-4
	(0.001287)	(0.002736)
Rastrigin	0.040865	0.008293
	(0.220068)	(0.044663)
Rosenbrock	0.001090	0.010790
	(0.005870)	(0.058107)
Sphere	2.2695E-4	0.001001
	(0.001222)	(0.005390)

Table. 1. Mean of best fitnesses

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

Table. 2. Wilcoxon test for validation

B. 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. Next, the set of parameters used.

- (μ, λ)-ES-1: $\mu = 10$; $\lambda = 10$; generations = 100; population recombination = discrete; mutation steps recombination = intermediate; lower limit for mutation step = -1; upper limit for mutation step = 1.
- (μ, λ)-ES-2: $\mu = 10$; $\lambda = 20$; generations = 100; population recombination = discreta; mutation steps recombination = intermediate; lower limit for mutation step = -1; upper limit for mutation step = 1.

- c(μ, λ)-ES: generations = 100; population recombination = discreta; mutation steps recombination = intermediate; lower limit for mutation step = -1; upper limit for mutation step = 1.

Table III 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 IV 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	0.105397	0.065293	0.685472
	(0.216403)	(0.218540)	(1.703286)
Booth	3.873424	0.000698	0.029533
	(5.991175)	(0.000756)	(0.023305)
Dixon	0.286664	0.000639	0.017207
	(0.525051)	(0.000761)	(0.019740)
Griewank	0.461445	0.255029	0.660719
	(0.265004)	(0.264967)	(0.668455)
Hump	0.218676	0.000641	0.052358
	(0.377594)	(0.000499)	(0.145693)
Levy	0.007013	0.005408	0.114306
	(0.010092)	(0.018034)	(0.430628)
Matyas	0.092499	0.000033	0.001468
	(0.162598)	(0.000048)	(0.002028)
Rastrigin	2.310114	0.175241	2.321291
	(1.650039)	(0.186175)	(1.428543)
Rosenbrock	0.849357	0.033554	1.174561
	(1.187724)	(0.066951)	(2.063353)
Sphere	0.063853	0.000170	0.007260
	(0.086762)	(0.000170)	(0.007272)

Table. 3. 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. 4. Wilcoxon 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, 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. In a comparison with other studies and literature, algorithms must also be taken into account. 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, J. *et al*. 2001] APORNTEWAN, Chatchawit; CHONGSTITVATANA, Prabhas. A hardware implementation of the compact genetic algorithm. In: IEEE Congress on Evolutionary Computation. 2001. pp. 624-629.

[BARAGLIA, J. *et al*. 2001] 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.

[EIBEN, J. *et al*. 2003] EIBEN, Agoston E.; SMITH, James E. Introduction to evolutionary computing. Springer Science & Business Media, 2003.

[HANSEN, J. *et al*. 2006] HANSEN, Nikolaus. The CMA evolution strategy: a comparing review. In: Towards a new evolutionary computation. Springer Berlin Heidelberg, 2006. pp. 75-102.

[HARIK, J. *et al*. 1999] 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, J. *et al*. 1975] 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 2015] 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 2015, March 18th.

[JEWAJINDA, J. *et al*. 2008] 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, J. *et al*. 2002] LARRAÑAGA, Pedro; LOZANO, Jose A. (Ed.). Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science & Business Media, 2002.

[MININNO, J. *et al*. 2011] MININNO, Ernesto et al. Compact differential evolution. Evolutionary Computation, IEEE Transactions on, v. 15, n. 1, pp. 32-54, 2011.

[PARADISO, J. *et al*. 2005] PARADISO, Joseph A.; STARNER, Thad. Energy scavenging for mobile and wireless electronics. Pervasive Computing, IEEE, v. 4, n. 1, pp. 18-27, 2005.

[SASTRY, J. *et al*. 2000] 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, J. *et al*. 2001] SASTRY, Kumara; XIAO, Guanghua. Cluster Optimization Using Extended Compact Genetic Algorithm. Urbana, v. 51, p. 61801, 1989.

[SASTRY, J. *et al*. 2007] 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.

[SHWEFEL, J. *et al*. 1974] SHWEFEL, H. P. Numerische Optimierung von Computer-Modellen. PhD Thesis, 1974. Decision support methods for global optimization.