# NEW EVOLUTIONARY PARTICLE SWARM ALGORITHM (*EPSO*) APPLIED TO VOLTAGE/VAR CONTROL

Vladimiro Miranda
vmiranda@inescporto.pt

Nuno Fonseca
nfonseca@power.inescn.pt

INESC – Instituto de Engenharia de Sistemas e Computadores do Porto
P. República 93 – 4050 Porto – Portugal – Fax 35122084172
FEUP – Faculdade de Engenharia da Universidade do Porto, Portugal

**Abstract – This paper presents a new optimization model – EPSO, Evolutionary Particle Swarm Optimization, inspired in both Evolutionary Algorithms and in Particle Swarm Optimization algorithms. The fundamentals of the method are described, and an application to the problem of Loss minimization and Voltage control is presented, with very good results.**

***Keywords: Evolutionary Algorithms, Particle Swarm Optimization, Voltage/Var control***

## 1. INTRODUCTION

This paper has the main objective of introducing to the Power System community a new and powerful meta-heuristic hybrid variant called EPSO – Evolutionary Particle Swarm Optimization.

EPSO is a general-purpose algorithm and it can thus be applied to a diversity of problems in any scientific area. However, in order to illustrate the technique, we have selected a problem in the Power Systems environment and will therefore present in the paper, in the application sections, a solution for the Voltage/Var control problem obtained by EPSO and the comparison of its performance with existing methods

The Particle Swarm Optimization is an optimization algorithm that was introduced in 1995 by Kennedy [1]. We will refer to it as: Classic PSO. Imagine that we have a population of particles looking around in a given search space for the global optimum. This particle movement mimics, in a way, the coordinated movement of flocks of birds, schools of fish or swarms of insects: this is a good image of a PSO optimization algorithm.

In Evolutionary Algorithms, there is no coordination in the movement of individuals within the search space. However, the powerful selection procedure allows solutions with superior characteristics to pass these from generation to generation, while the mutation (and recombination) schemes produce diversity in the solution pool.

EPSO joins together the "best of two worlds". It is a Particle Swarm algorithm, because there is exchange of information among solutions, when they are successively moved around in the search space; and it is an Evolutionary Computation method, because solution characteristics are mutated and passed to the following generations by the action of a selection mechanism.

In a classical PSO model, particle movement is conditioned by three strategic parameters: inertia, memory and coordination (information exchange).

In previous applications of the Classic PSO and other variants like CPSO (Cooperative PSO) [2], the strategic parameters of the algorithms were set to certain values that had already been used with god results. But there is no valid explanation to sustain that we should use a particular value for those parameters if we have a different problem.

Also, we can't say that a certain value is the best during all the process of optimization. That had already been realized for the inertia factor. This parameter is usually decreased as the number of iteration increases [3].

EPSO [4] defines these parameters as the genotype of a moving solution. Therefore, they are subject to mutation and the particles holding them as phenotypes are subject to selection. This scheme turns out to be a successful self-tuning mechanism, a self-adaptive evolutionary process acting on "strategic parameters", to use the language of the Evolution Strategy community.

As we will show, EPSO has a better behavior than Classical PSO (namely, it is robust, insensitive in a large degree to initial values of parameters) and it also has a better behavior than other meta-heuristics (in this paper, a comparison will be made with simulated annealing).

The hybrid characteristics of Evolutionary and of Particle Swarm model give it guaranteed convergence properties. In terms of efficiency, therefore, lower bounds are guaranteed, but experience demonstrates that there is an effective acceleration and a better search for the optimum than classical approaches.

## 2. *EPSO* DESCRIPTION

In EPSO, each particle (solution at a given stage) is defined by the following characteristics:

- position in the search space ($x_i^k$; value of the coordinate position *i*, for the *k* particle)
- velocity ($v_i^k$; value of the coordinate velocity *i* for the *k* particle).

At a given moment, there is at least one particle that holds the best position in the search space. The population of particles is aware of such position, represented as ($x_i^{best}$, value of the coordinate position *i*, for the *best* particle).

Each particle also keeps track of its previous best position ($x_i^{k,mem}$, value of coordinate position *i,* memorized as its previous best, for the *k* particle).

The particles will reproduce and evolve along a number of generations, according to the following steps:

- Replication: each particle is replicated a number *r* of times, giving place to identical particles (in this paper we take *r* = 1).
- Mutation: the strategic parameters of the replicated particles undergo mutation according to:

$$*\, w_{i,j}^k = w_{i,j}^k + \tau N(0,\sigma^2) \qquad (1)$$

where $\tau$ is a learning dispersion parameter and N(0,1) is a random number following a the normalized Gaussian distribution with zero mean and variance $\sigma^2$.

The strategic parameters are randomly set between 0 and 1 at the beginning of the algorithm. In each iteration, the strategic parameters of the replicated particles are mutated according to equation (1). In this equation, *j* can be the inertia, memory or the coordination factor.

- Reproduction (movement): each particle generates as offspring a new particle according to the transformation process, similar to the Classic PSO basic equation:

$$*v_i^k = w_{i,inertia}^k v_i^k + w_{i,mem}^k (x_i^{k,mem} - x_i^k) + w_{i,coop}^k (x_i^{best*} - x_i^k) \qquad (2)$$

$$*x_i^k = x_i^k + *v_i^k \qquad (3)$$

The offspring is held separately for the original particles and for the mutated particles.

Furthermore, instead of defining a crisp best-so-far point as a target, the particles are attracted to a sort of "foggy best-so-far region" (another change relative to Classic PSO).

This is done by introducing random noise in the definition of the best-so-far point:

$$x_i^{best*} = x_i^{best} + \tau' N(0,1) \qquad (4)$$

$\tau$' is a noise dispersion parameter, usually small, and N(0,1) is a random number following a the normalized Gaussian distribution with zero mean and variance 1.

- Evaluation: each offspring particle plus the originals are evaluated according to their current position.
- Selection: among the offspring of a particle, with and without mutated parameters, a stochastic tournament is played to select the particle that will survive to the next generation.

As in many other meta-heuristics, EPSO deals with inequality constraints through a penalty strategy. In the case of EPSO, the selective pressure applied helps in eliminating the individuals or particles with excursions outside the feasible domain, which receive a penalized fitness value.

## *3.* TESTING *EPSO*

In this section we illustrate the superiority of EPSO regarding the Classic PSO algorithm, in solving classical difficult test problems.

### *3.1 Test functions*

Schaffer's function:

$$f_1(x) = 0.5 + \frac{(\sin\sqrt{x^2+y^2})^2 - 0.5}{(1.0+0.001(x^2+y^2))^2} \qquad (5)$$

Rosenbrock function:

$$f_2(x) = \sum_{i=1}^{n} (100 \times (x_{i+1} - x_i^{\,2})^2 + (x_i - 1)^2) \qquad (6)$$

Sphere function:

$$f_3(x) = \sum_{i=1}^{n} x_i^{\,2} \qquad (7)$$

Alpine function:

$$f_4(x) = \sin(x_1)\sin(x_2)\sqrt{x_1 x_2} \qquad (8)$$

The parameters used in these functions are presented in Table 1. The threshold used as the stopping criterion is listed in the "Stop" column.

| Function | n | Domain | Stop |
|---|---|---|---|
| $f_1$ | 2 | $[-50, 50]^n$ | 1.0E-10 |
| $f_2$ | 30 | $[0, 30]^n$ | 100 |
| $f_3$ | 30 | $[-50, 50]^n$ | 0.01 |
| $f_4$ | 2 | $[0, 100]^n$ | 98.9627 |

**Table 1:** Parameters used in the test functions.

*3.2 Results in the test functions*

The following pictures illustrate the typical convergence in the test functions, for the EPSO and PSO.

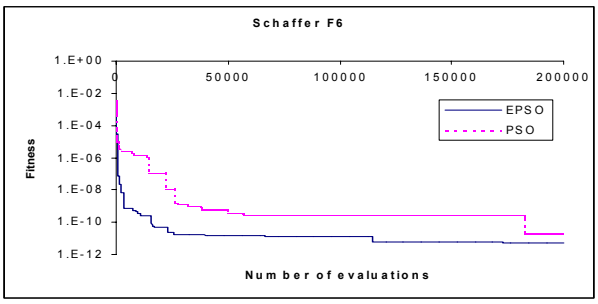


Figure 1 - Typical convergence in the Schaffer's function.

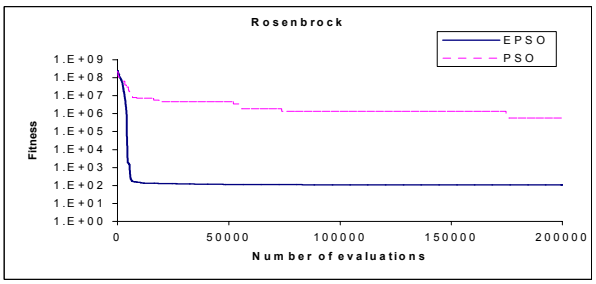


Figure 2 - Typical convergence in the Rosenbrock function.

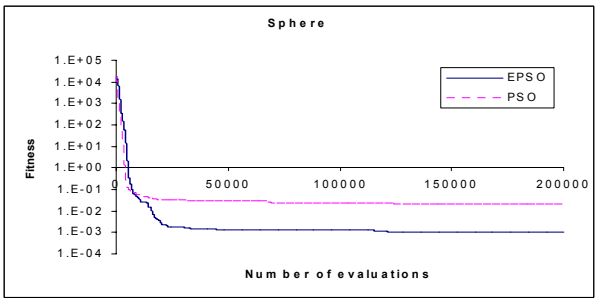


Figure 3 - Typical convergence in the Sphere function.

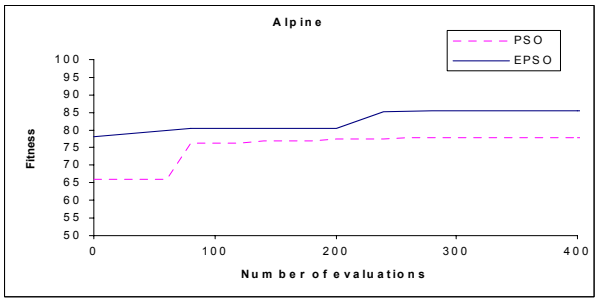


Figure 4 - Typical convergence in the Alpine function.

The results presented in Figures 1, 2, 3 and 4 show a clear superiority of the EPSO algorithm. The PSO results could be optimized if we've tuned by hand the strategic parameters. EPSO was able to provide better results independently of the strategic parameter initialization.

If we are trying to optimize a different problem (ex: Optimal Power Flow), where we don't now which are the better strategic parameters, then EPSO is certainly better because of the self-tuning mechanism.

To demonstrate the superiority of EPSO over the Classic PSO we compare the average number of evaluations that both algorithms need to reach the stopping criterion. The maximum number of evaluations was fixed in 200000. Table 2 presents the results of this test.

| Function | EPSO | PSO |
|---|---|---|
| $f_1$ | 11862.2 | 59547.0 |
| $f_2$ | 27005.3 | 180310.8 |
| $f_3$ | 16421.4 | 161625.0 |
| $f_4$ | 78539.8 | 199190.1 |

**Table 2:** Comparison of EPSO with the Classical PSO: average number of evaluations.

We also compare the average results of both algorithms for a fixed number of evaluations. So, considering a number of evaluations of 200000 (20 particles over 5000 iterations in the EPSO algorithm and 20 particles over 10000 iterations in the Classical PSO) we obtained the following results:

| Function | EPSO | PSO |
|---|---|---|
| $f_1$ | 2.15E-13 | 5.45E-11 |
| $f_2$ | 33.8828 | 114.443 |
| $f_3$ | 7.81E-04 | 1.91E-02 |
| $f_4$ | 98.9627 | 86.1071 |

**Table 3:** Comparison of EPSO with the Classical PSO: average results.

The results presented in Table 2 and Table 3 were obtained with a population of 20 particles, over 500 simulations.

Notice that the position and the strategic parameters (inertia, coordination factor and memory) were always initialized randomly for each particle. The results of the PSO would have perhaps a margin for improvement if the strategic parameters were even better tuned by hand, but this would involve a tedious work of experimentation case

by case. The EPSO algorithm was able to provide immediately good results independently of such initialization.

This is a very important improvement in the algorithm, because the results of Classic PSO are reported to be very dependent of the strategic parameter initialization [3], and this has been confirmed by our experience. This statement does not mean that one could not find solutions with "less good" initial PSO parameters – but the performance of the PSO algorithm, in our experiments, never reached the quality of EPSO, did not display on average the same quality of results and, most important, did not display the same robustness, which is vital for a practical application – the users must trust the algorithm, must believe it gives reliable and consistent results, must be confident that, if they run it a number of times, they will get the same kind of answer.

## 4. APPLICATION OF *EPSO* TO VOLTAGE/VAR CONTROL

### *4.1 Loss reduction in distribution systems*

The application of PSO-like algorithms to the Voltage/Var control problem was pioneered by authors in Japan and reported in [6][7][8]. Their models included a form of blending evolutionary concepts with the PSO algorithm, with positive results. However, their valuable work remained one step away from a true self-adaptive approach, which is what this paper now presents.

We illustrate EPSO in a loss reduction-Voltage/Var control problem for a didactic example with the IEEE 24 nodes/36 branches network defined in [9]. This network also includes 31 transmission lines, 5 transformers, 11 capacitor banks and 9 synchronous generators. The size of a problem of this nature, however, is not related really with the size of the network but with the number of controllers available.

For the sake of a comparison with a competing algorithm, based on Simulated Annealing, we took as control variables only the set point of transformers and capacitor banks.

This Simulated Annealing [10] [11], algorithm is a well tested application developed by INESC Porto and included in a commercial DMS, used by a number of utilities. The problem of Voltage/Var control can be formulated as follows:

$$\text{Minimize} \quad \Im\ (u,x) \tag{9}$$

$$\text{Subject} \quad \varphi\ (u,x,p) = 0 \tag{10}$$

$$\phi\ (u,x,p) \le 0 \tag{11}$$

Equation (9) is the objective function of this problem and, in general, represents the active losses. The constraints of this problem, (10) and (11), are respectively the power flow equations and operation limits, namely bands of admissible voltage values at nodes.

All these equations, written in a general form, must be understood as representing a full AC model, with losses evaluated, for instance, with a Newton-Raphson algorithm. Because these are well known equations, we felt we could take the liberty of adopting the above representation.

We may have also other objectives, such as the preference for keeping control margins, i.e., searching for solutions that do not require the set points of controllers to be at their maxima or minima. This means that one is facing a multicriteria problem, with two objectives:

- Minimize losses
- Minimize distance of control variables from nominal set points (usually, the center on the intervals defining their range of variation).

In fact, this is achieved in practice by applying a penalty factor to the fitness function, such as depicted in Figure 5.

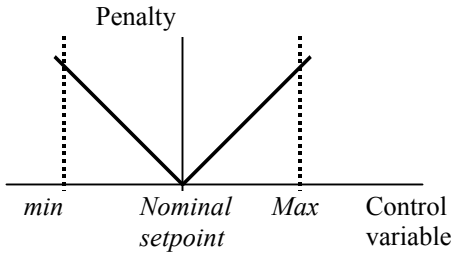


Figure 5 – Example of penalty function to be added to the loss function (per control variable, scaled by a weighting factor) to favor solutions that do not push controls to their limits

The Voltage/Var control problem in distribution systems is usually a problem of minimizing losses and controlling voltage levels, by acting on transformer taps and on capacitor bank taps. It is rare to find synchronous generators directly connected to the network where one could act on their excitation. However, EPSO can deal with these variables as well, with excellent results.

### *4.2 Results of the loss reduction problem*

In order to compare EPSO results with those obtained with the Simulated Annealing (SA) application we needed to establish the same stopping criterion. As the Simulated Annealing already had this criterion fixed as a certain number of iterations without improvement in the best-so-far solution, we used the same criterion. For this particular exercise, the maximum number of iterations allowed without improvement in the solution was fixed in 270.

In this particular application, all the variables of control are discrete (set point of transformers and capacitor banks). There is a version of Discrete PSO [12], but as for now, the EPSO only deals with continuous variables.

We've used "probabilistic rounding" to solve this problem. Instead of using simple rounding, i.e., consider the nearest value, we've considered that the probability of rounding to the nearest discrete value increases as the distance decreases. On average, the value of the variable is probabilistically rounded to the nearest discrete value, but there is always the possibility that it is not, at any given point.

This scheme avoids trapping in local discrete values, and has all the flavor of the techniques used in evolutionary computing.

In terms of convergence comparison between both algorithms, we can immediately reveal that:

- EPSO finds its best solution in less iterations.
- the initial solution is better for the EPSO, because it has a population of particles, while the Simulated Annealing only starts with one initial solution.
- there is an extra computing effort in applying EPSO, when compared to the Simulated Annealing option (measured in the number of load flows run);
- EPSO consistently discovers better solutions than the Simulated Annealing algorithm.

A typical convergence pattern observed for both algorithms can be observed in Figure 5, where EPSO (as usual) found a better solution than SA.

We tested EPSO with different population sizes. The objective of this test was to find out the influence of population size in the results. As it can be seen in Table 4, the quality of the solution improves if we increase the number of particles. Of course, there is a price to pay in terms of computing effort.

As for now, we were not able to establish a secure rule for defining the optimal number of particles. Our assumption is that this will depend on the complexity of the problem, and with increasing complexity it will be necessary to increase the number of particles.

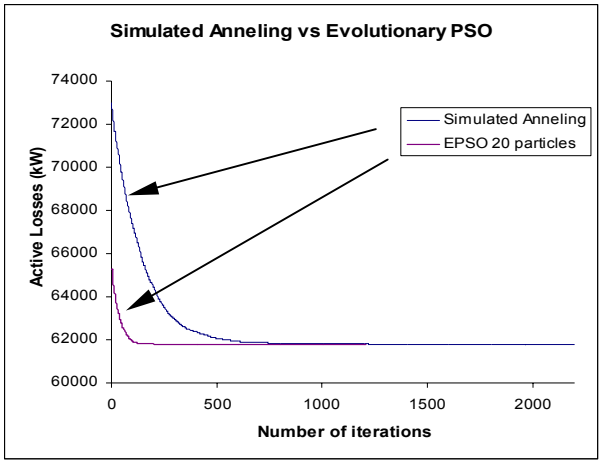


Figure 5 - Comparison in convergence between EPSO and Simulated Annealing

| | Average losses (MW) | Std. Deviation (kW) to the best solution |
|---|---|---|
| EPSO 2 particles | 61.7947 | 3.10334 |
| EPSO 5 particles | 61.7912 | 2.64314 |
| EPSO 10 particles | 61.7889 | 1.96607 |
| EPSO 20 particles | 61.7880 | 1.48473 |
| Simulated Annealing | 61.7921 | 9.81175 |

**Table 4:** Comparison of EPSO with Simulated Annealing

As we can see in Table 4, the EPSO reveals superiority in terms of the solution found (both in the best solution discovered and in the average optimum obtained in 1000 runs, as seen in the Table) and in terms of its robustness (evaluated as the root of the mean square error, or standard deviation, relative to the best solution found).

In particular, EPSO gives consistently a near-optimum result, while the Simulated Annealing model failed many times to reach a solution as good (and that's why the dispersion of results in this case is much larger than with EPSO).

Therefore, EPSO is a much more reliable algorithm for practical applications.

### *4.3 Voltage control*

For this test we've increase the reactive load in bus 8 of the same IEEE 24-bus system. The voltage at this bus became very low and we run the EPSO algorithm to re-dispatch the reactive power in order to set the voltage back inside the limits.

As it can be seen in Figure 6, the algorithm was able to find a new set point, to both transformers and capacitor banks, which forced the voltage into the acceptable limit (0.9 – 1.1 p.u.).

The Simulated Annealing algorithm failed to obtain a feasible solution for this case.

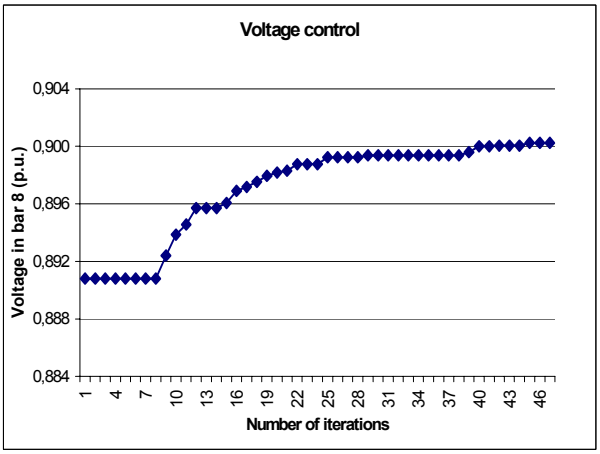


Figure 6 - Voltage Control with EPSO – evolution of the controlled voltage along the iterations of an EPSO algorithm.

## 5. CONCLUSIONS

This paper reports two important results:

- A new optimization technique, with roots both in Evolutionary Computing and in Particle Swarm algorithms.
- A new model for loss minimization and voltage control

First of all, there is a new successful meta-heuristic tool, available for optimization of complex problems with multiple local optima – EPSO, the Evolutionary Particle Swarm Optimization method.

EPSO joins together the characteristics of Evolutionary Algorithms and of Particle Swarm Algorithms.

From an Evolutionary Computing point of view, there is another operator introduced, side by side with recombination and mutation, which generates new (and promising) solutions in the search space – it is Reproduction in the form of Particle Movement.

From a Particle Swarm point of view, there is a self-adaptive tuning of the algorithm by evolutionary adjustment of the parameters controlling particle movement.

Both points of view are legitimate and justify the remarkable convergence characteristics of the method.

The second important result is that EPSO proves very successful in solving a Power System optimization problem – minimizing losses in a transmission system. In fact, EPSO performed better than a Simulated Annealing model that has been used by utilities, both in the quality of the solution discovered and in the robustness of the result (dispersion around the best result, found in a number of repeated runs).

In the tests done, the Simulated Annealing algorithm demanded a somewhat smaller computer effort (measured in the number of load flows required) but failed completely to discover the best solutions, while EPSO was able to converge to them in all cases.

Furthermore, EPSO was again successful in a Voltage Control problem, easily discovering a solution for a difficult problem where other techniques experiment difficulties in converging.

One expects that EPSO may be applied with equal success to other problems in Power Systems.

## REFERENCES


[1] Kennedy, J., R.C. Eberhart, “Particle Swarm Optimization”, IEEE International Conference on Neural Networks, Pert, Australia, IEEE Service Center, Piscataway, NJ., 1995

[2] F. van den Bergh, A.P. Engelbrecht, “Training Product Unit Networks using Cooperative Particle Swarm Optimization”, International Joint Conference on Neural Networks (IJCNN), Washington D.C., 2001.

[3] Yuhui Shi, Russell C. Eberhart, “Parameter Selection in Particle Swarm Optimization”, Proceedings of the Seventh Annual Conference on Evolutionary Programming, 1998.

[4] Vladimiro Miranda, Nuno Fonseca, “EPSO-Evolutionary self-adapting Particle Swarm optimization”, internal report INESC Porto, July 2001 (obtainable from the authors by request).

[5] F. van den Bergh, A.P. Engelbrecht, “Effects of Swarm Size on Cooperative Particle Swarm Optimizers”, Proceedings of the Genetic Evolutionary Computation Conference (GECCO), , 2001.

[6] Yoshida, H., Fukuyama, Y., Takayama, S. and Nakanishi, Y., “A particle swarm optimization for reactive power and voltage control in electric power systems considering voltage security assessment”, IEEE Proc. of SMC '99 , Vol. 6, pp.497 -502, 1999.

[7] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, “A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment”, IEEE Trans. on Power Systems, vol. 15, no. 4, pp.1232-1239, Nov. 2000.

[8] Fukuyama, Y., Yoshida, H., “A particle swarm optimization for reactive power and voltage control in electric power systems”, IEEE Proc. of Evolutionary Computation 2001 , Vol.1 , pp. 87 -93, 2001.

[9] Reliability Test System Task Force of the Application of Probability Methods Subcommittee, “IEEE Reliability Test System ”, IEEE Trans. On Power Apparatus and Systems, vol. PAS-98, no. 6, Nov./Dec. 1979.

[10] Jorge Pereira, J. Tomé Saraiva, Maria Teresa Ponce de Leão, "Identification of Operation Strategies of Distribution Networks Using a Simulated Annealing Approach", Proceedings of IEEE Budapest Power Tech'99, paper BPT99-357-17, August 1999.

[11] Manuel Matos, Maria Teresa Ponce de Leão, J. Tomé Saraiva, J. N. Fidalgo, et al., "Meta-heuristics Applied to Power Systems", Proceedings of MIC´2001 - 4th Metaheuristics International Conference, Porto, Portugal, vol.2, pp.483-488, July, 2001.

[12] Kennedy, J. and Eberhart, R. C., “A discrete binary version of the particle swarm algorithm”, Proc. Conf. on Systems, Man, and Cybernetics, 4104–4109. Piscataway, NJ: IEEE Service Center, 1997.