New experiments with EPSO – Evolutionary Particle Swarm Optimization

V. Miranda, *Fellow IEEE* Naing Win Oo
INESC Porto and also FEUP, Faculty of Engineering of the University of Porto, Portugal
vmiranda@inescporto.pt

Abstract – This paper presents some new ideas to improve the performance of EPSO (Evolutionary Particle Swarm Optimization). It discusses a Stochastic Star communication scheme and differential dEPSO. The paper presents results in a didactic Unit Commitment/Generator Scheduling Power System problem and results of a competition among algorithms in an intelligent agent platform for Energy Retail Market simulation where EPSO comes out as the winner algorithm.

I. Introduction

THIS paper discusses some recent results obtained with a model that has been called EPSO – Evolutionary Particle Swarm Optimization.

The interesting aspect of this model, from a conceptual point of view, is that it allows a double interpretation on how it works, because it may be seen from two perspectives: either as a variant of the PSO – Particle Swarm Optimization, or as a variant of Evolutionary Algorithms. This hopefully will help in understanding how the method works and how should one manipulate its characteristics to obtain better convergence characteristics in specific problems.

The name EPSO, as far as the author is aware of, has been first coined in 2002, in [1]. Subsequently, a few papers more presented some interesting results and applications [2][3][4][5]. However, more recently (1995) some other publications have started to use the acronym EPSO for other type of algorithms, such as “extended PSO” [6],“enhanced PSO”[7] or even “emotional PSO”[8].

It is unfortunate that this may lead to some confusion to readers and researchers. We wish to make clear from the start that by EPSO we refer to a class of algorithms of the family of self adaptive EA – Evolutionary Algorithms, where the classical operators for recombination are replaced by a rule similar to the particle movement of PSO; alternatively, we refer to a class of algorithms of the family of PSO where the weights associated to the components of the movement rule are made to evolve in a self-adaptive mechanism.

II. Evolutionary Algorithms

A general Evolutionary Algorithm has the following steps:

```
Procedure EA
    initialize a random population P of $\mu$ elements
    REPEAT
        reproduction (introduce stochastic perturbations in the new
            population) – generate $\lambda$ offspring…
                …by recombination
                …by mutation
        evaluation - calculate the fitness of the individuals
        selection - of $\mu$ survivors for the next generation, based on
            the fitness value
        test for termination criterion (based on fitness, on number of
            generations, etc.)
    Until test is positive
End EA
```

The driving force of EA is the selection operator. However, it requires the action of the replication or reproduction operators – recombination and mutation – which generates new points in space to be evaluated.

Mutation is an operator acting on a single individual, particle or chromosome. Recombination generates a new individual by combining features from a set of individuals in the population. The definition of the mutation operator may be dependent of the particular problem under analysis. In Evolution Strategy/Evolutionary Programming, where typically (not necessarily) an individual is composed of a string of real variable values, the classical mutation procedure adopts Gaussian or Lognormal random perturbations of each variable. The amplitude of mutation is governed by the variance of the distribution regulating mutations, and its square root (standard deviation) is often called *learning rate*.

In self-adaptive Evolutionary Algorithms, the learning rates are transformed into variables and added to the chromosome [9][10]. During the process, they are themselves subject to mutation and selection and eventually they acquire values that allow a near optimal progression rate towards the optimum [11]. Self-adaptation has been used exclusively on the operator mutation.

For the operator recombination many forms have been proposed. The most important are:

Uniform crossover – in this variant, the value for each variable in the newly formed individual is obtained by randomly selecting one of the μ parents to “donate” its value.

Intermediary recombination – in this variant, the value of any variable in the offspring receives a contribution from all parents. This could result either from averaging the values of all parents (global intermediary recombination) or from averaging the values from a subset of the parents only, randomly chosen (local intermediary recombination). In these processes, one may still chose to average values with equal weights or to randomly define weights for a weighted average. In the case of μ = 2, one could have the value of a variable given by

$$x_k^{new} = u_k x_{k,j1} + (1 - u_k) x_{k,j2}$$

where the indices j1 and j2 denote the two parent individuals and u_k is sampled from an uniform distribution in [0,1]. This leads to new individuals in the line segment between the two parents, but the concept may extend to points in the line but external to the segment if u_k is allowed to be sampled in a larger interval.

Point crossover – in this variant, parallel to the one adopted in genetic algorithms, first one randomly defines π crossover points, common to all individuals in the set of parents, and then the offspring successively receives a part from each parent, in turns.

Some EA give relevance to the mutation operator while others relies mostly on the recombination operator.

III. The Movement Rule of PSO

PSO – Particle Swarm Optimization [12] is not an Evolutionary method. In fact, it does not rely on a selection operator as its driving force. It depends on *movement rule* that allows the generation of new individuals in space and this rule is such that, by itself, pushes the search towards the optimum.

Variants have been proposed but the basic movement rule, producing a new individual **X** for iteration (k+1) is based on

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{V}_i^{(k+1)}$$

where $\mathbf{V}_i$ is called the particle i velocity and is defined by

$$\mathbf{V}_i^{(k+1)} = \mathbf{A}\mathbf{V}_i^{(k)} + \mathbf{B}(\mathbf{b}_i - \mathbf{X}_i^{(k)}) + \mathbf{C}(\mathbf{b}_G - \mathbf{X}_i^{(k)})$$

where the first term of the summation represents inertia or habit (the particle keeps moving in the direction it had previously moved), the second represents memory (the particle is attracted to the best – past – point in its trajectory) and the third represents cooperation or information exchange (the particle is attracted to the best point found by all particles).

The parameters **A, B, C** are diagonal matrices with weights fixed in the beginning of the process (index m is for the memory weights and index c is for the cooperation weights). In a classical formulation, the parameter **A** is affected by a decreasing value with time (iterations), while the parameters **B** and **C** are multiplied by random numbers sampled from a uniform distribution in [0,1].

A more elaborate version of PSO adopts the so called constriction factor [13] but we will not discuss it further because it does not invalidate or contradict our conclusions.

IV. EPSO as an Evolutionary Algorithm

A. Recombination in EPSO

In 1992 we have proposed an algorithm called EPSO [1] where instead of using the classical mutation and recombination operators to produce new individuals we have adopted the general scheme of the movement rule of PSO.

If we examine this scheme, we conclude that a new particle $\mathbf{X}_i^{(k+1)}$ is formed as a combination of four other points:

- Its direct ancestor $\mathbf{X}_i^{(k)}$
- The ancestor of its ancestor $\mathbf{X}_i^{(k-1)}$
- A (possibly) distant past best ancestor $\mathbf{b}_i$
- The current global best of the swarm $\mathbf{b}_G$

We can give a different aspect to the movement rule:

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{A}(\mathbf{X}_i^{(k)} - \mathbf{X}_i^{(k)}) + \mathbf{B}(\mathbf{b}_i - \mathbf{X}_i^{(k)}) + \mathbf{C}(\mathbf{b}_G - \mathbf{X}_i^{(k)})$$

$$\mathbf{X}_i^{(k+1)} = (1 + \mathbf{A} - \mathbf{B} - \mathbf{C})\mathbf{X}_i^{(k)} - \mathbf{A}\mathbf{X}_i^{(k-1)} + \mathbf{B}\mathbf{b}_i + \mathbf{C}\mathbf{b}_G$$

and we realize now that the sum of the parameters multiplying the four contributors to generate the offspring is equal to 1. Is is therefore tempting to identify this expression with an intermediary recombination in EA with $\mu = 4$ and a special rule to determine who the parents are (they are not randomly selected). This means that we are considering an *enlarged population* including not only the active particles but also the direct ancestors and the set of the past best ancestors.

It is a recombination rule that has the remarkable property of pushing the population towards the optimum, as the PSO algorithms have demonstrated. Therefore, if joined together with a selection mechanism, which also pushes the population towards the optimum, one may expect that some cumulative effect may improve the performance of an optimizing algorithm.

B. Self-adaptive recombination

To determine the best values to use in **A**, **B** and **C**, EPSO relies on a self-adaptive mechanism. These parameters, considered as strategic, will be subject to selection and hopefully will evolve to values adapted to the type of landscape being searched.

Given a population with a set of particles, the general scheme of EPSO is the following:

- REPLICATION - each particle is replicated r times
- MUTATION - each particle has its strategic parameters mutated
- REPRODUCTION - each mutated particle generates an offspring through recombination, according to the particle movement rule, described below
- EVALUATION - each offspring has its fitness evaluated
- SELECTION - by stochastic tournament or other selection procedure, the best particles survive to form a new generation, composed of a selected descendant from every individual in the previous generation

Mutation of a parameter w into w* is ruled by multiplicative Lognormal random numbers such as in $w_i^* = w_i\,[\log N(0,1)]^\tau$ or by additive Gaussian distributed random numbers such as in $w_i^* = w_i + \sigma N(0,1)$. The learning parameter (τ or σ) must be fixed externally.

The recombination rule for EPSO becomes

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \mathbf{V}_i^{(k+1)}$$

$$\mathbf{V}_i^{(k+1)} = w_{i1}^*\mathbf{V}_i^{(k)} + w_{i2}^*(\mathbf{b}_i - \mathbf{X}_i) + w_{i3}^*(\mathbf{b}_G^* - \mathbf{X}_i)\mathbf{P}$$

where $\mathbf{b}_i$ – best point found by particle i in its past life up to the current generation

$\mathbf{b}_G$ – best overall point found by the swarm of particles in their past life up to the current generation

$\mathbf{X}_i^{(k)}$ – location of particle i at generation k

$\mathbf{V}_i^{(k)} = \mathbf{X}_i^{(k)} - \mathbf{X}_i^{(k-1)}$ – is the velocity of particle i at generation k

w_{i1} – weight conditioning the *inertia* term

w_{i2} – weight conditioning the *memory* term

w_{i3} – weight conditioning the *cooperation* or *information exchange* term

P – communication factor.

The symbol * indicates that the parameter will undergo mutation. In the most effective EPSO variant, not only the weights affecting the components of movement are mutated but also the global best $\mathbf{b}_G$ is randomly disturbed to give

$$\mathbf{b}_G^* = \mathbf{b}_G + w_{i4}^* N(0,1)$$

where w_{i4} is the forth strategic parameter (target weight) associated with particle i. It controls the "size" of the neighborhood of $\mathbf{b}_G$ where it is more likely to find the real global best solution.

Selction is modeled from the Stochastic Tornament concept: among the ofspring of each particle, one compares the best one with another particle randomly sampled, and the best is selected with probability (1 – luck), where the *luck* parameter is defined in [0,1] but is usually small. If luck = 0 we have elitist selection.

C. Communication topology

The communication factor **P** induces a *stochastic star* topology for the communication among particles. It is a diagonal matrix affecting all dimensions of an individual, containing binary variables of value 1 with probability p and value 0 with probability (1-p); the p value controls the passage of information within the swarm and is 1 in classical formulations (this the *star*).

This stochastic scheme oscillates between the star arrangement and a selfish version called *cognitive model* in [14], where no communication exists and a descendent of an individual is built only of contributions from its ancestor line.

Experimental results have suggested that a communication probability of p = 0.20 leads in many cases to better results than a classical deterministic star model with p = 1. One is lead to believe that restraining the free flow of information about the global best allows more local search by each particle and avoids premature convergence. As it is easily observed, this is yet another way of acting on the recombination operator.

D. EPSO as a PSO

Because EPSO adopts the movement rule of PSO, we can always look at how it works as a swarm whose movement is ruled by weights that self adapt in order to produce a global drift more adapted to the landscape. Furthermore, because these weights are subject to mutation, this may give an extra chance for the swarm to escape local minima (i.e., having particles that still explore other regions of space, because they may gain enough speed).

On the other hand, EPSO also shows ability to focus and zoom in the optimum, precisely because mutations in the weights may favor the selection of the cooperation factor and reduce the importance of inertia and memory, if this strategy proves successful. This may in part explain why EPSO has shown, in many tests, robustness by consistently reaching the same optimum in a number of runs.

V. Experiments with EPSO

A. Communication probability p in the Rosenbrock function

The effect of the communication probability p was observed in the Rosenbrock function, a very difficult one whose expression is

$$f(X) = \sum_{d=1}^{D-1} (1 - x_d)^2 + 100 \times (x_d^2 - x_{d+1})^2 , \quad X \in [0 ; 30]^D$$

Experiments in a space of D = 30, with a swarm of 20 particles, additive mutation with σ = 0.1, r = 2 and a maximum effort of 200,000 fitness function evaluations, supplied this comparison:

	p = 1	p = 0.2
Average in 20 runs	55.253326	27.097839
St. deviation	43.361223	1.4124137

The improvement is remarkable. Notice from the std. deviation value that some results in the case of p=1 were very good but others were bad. The results for p = 0.2 display good robustness (narrow variance) placing the solution in every case in the neighborhood of the point with coordinates (0,…,0), where f(X) = 29. The optimum is at point (1,…,1) but the basin of attraction is extremely narrow and the path to reach it a deep valley.

The same test but now using a multiplicative Lognormal mutation with τ = 0.1 gave

	p = 1	p = 0.2
Average in 20 runs	28.724101	26.114459
St. deviation	0.104350	0.958282

We see that multiplicative Lognormal mutations perform better than additive Gaussian mutations and that in this difficult problem restricting communication among particles releases the search from becoming too gripped to a given point earlier discovered.

B. The enhanced elitist version of EPSO

The difficulty with problems such as the Rosenbrock function is that during the evolution of the swarm, selection acts only on the offspring and never on the set of parents plus offspring (en Evolution Strategies language, it is a comma strategy and not a plus strategy). To introduce a plus strategy and limited elitism, we have introduced changes in EPSO by dealing with the particle that finds the global best in a different way to the rest of the swarm: the particle at the global best is not eliminated unless one of its descendents finds a better point and replaces it – it competes with its offspring and pure elitism applies, not stochastic tournament.

Furthermore, for this particle we generate not just r descendents but a set d > r – a local mini-swarm – and the mutations of the weights are done with a much narrower variance.

Applying this elitist version to the Rosenbrock function, with the same data and r = 2 but d = 4 and $\tau = 0.005$ just for the mini-swarm, we have obtained the following results:

	p = 1	p = 0.2
Average in 10 runs	26.226596	17.529253
St. deviation	0.5763617	9.0554841

This represents a noticeable improvement and indicates that exploring an elitist strategy is a way to improve the performance of EPSO in problems with narrow and difficult valleys such as is the case of the Rosenbrock function.

C. The differential evolution version of EPSO

There is more than a vague similarity between PSO and DE - Differential Evolution [15], an evolutionary algorithm that uses a recombination rule that resembles some characteristics of the movement rule of PSO. In its variant DE2, from a particle $\mathbf{X}^{(k)}$ at iteration k a new individual is produced through

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + w_{i2}(\mathbf{X}_{Ai} - \mathbf{X}_{Bi}) + w_{i3}(\mathbf{b}_G - \mathbf{X}_i)$$

where $\mathbf{X}_A$ and $\mathbf{X}_B$ are two different individuals sampled from the population and distinct from the current individual $\mathbf{X}$ and $\mathbf{b}_G$ is the global best. In this recombination operator, the attraction for the particle past best is replaced by the difference between to other individuals.

The general principle of DE relies on the perception by the algorithm of the topology of the function being optimized by sensing macro-gradients through differences of points. In a way, this is exactly what a PSO algorithm also does.

We decided to try to add this flavor to EPSO by changing the movement rule precisely in the memory element, in a version we will call dEPSO. Instead of using the difference between a particle and its past best, we will use in this term the difference between two particles A and B randomly selected among the population (this can be the enlarged population including the set of particles past bests). The movement rule of dEPSO becomes

$$\mathbf{V}_i^{(k+1)} = w_{i1}^*\mathbf{V}_i^{(k)} + w_{i2}^*(\mathbf{X}_{Ai} - \mathbf{X}_{Bi}) + w_{i3}^*(\mathbf{b}_G^* - \mathbf{X}_i)\mathbf{P}$$

The effectiveness of this model is tested in the following problem of unit commitment in power systems: given a set of generators and their generation cost curves, define which generators should be shut down and which should be in service and at which loading level, in order to minimize the overall cost (start up cost plus operation cost).

The problem is complex because of the cost functions for generators that must be considered – a mixed-integer non-linear program. Because of technical limits, the domain of a generator is not connected – there is a point (0,0) corresponding to generator shut down and then there is a gap until a point (P_{min}, $c(P_{min})$) corresponding to the technical minimum of the machine. And the general shape of the cost functions implies that the problem has a non-convex nature – therefore, many local optima may appear.

The comparative performance of dEPSO is tested in a particular problem. The data are:

- the number of generators – ngen = 5
- the parameters of the cost function of each generator – this function is assumed to be a cubic polynomial, with 4 parameters a_i for i=1 to 4:
  - $C = a_0 + a_1 P + a_2 P^2 + a_3 P^3$
- where C is the generation cost in \$/hour and P is the generator output in MW.
- the technical minimum and maximum of each generator P_{min} and P_{Max}
- the load, located at a single bus (transmission system neglected) – L = 15 MW

Cost curves and technical limits given as:

Generator	a_0	a_1	a_2	a_3	Pmin	Pmax
g1	1	0,5	0,1	0,03	0 or 1	10
g2	2	0,4	0,2	0	0 or 2	10
g3	4	0,3	0,3	0	0 or 7	10
g4	6	1,5	0,15	0	0 or 2	10
g5	0	4	0	0	0 or 1	10

The objective is to minimize the sum of the costs for the five generators, noting that the domain of each variable is not continuous. The EPSO model demands an individual defined by its object parameters (5 variables) and strategic parameters (4 weights). We used Lognormal mutations with $\tau = 1$, 20 particles, r = 2 and 1000 generations. Initial values of the weights were set to 0.5 except for the target weight where the value was set to 0.001.

The cost curves are depicted in Figure 1 as well as the optimal solution, which is

g1	**g2**	**g3**	**g4**	**g5**	**Cost**
3.414	4.586	7	0	0	33.9068

Figure 2 displays the evolution of best fitness during an EPSO run and Figure 3 shows the weights of the global best particle kept at each iteration, for the same run. The best particle pattern shows that improvements were made till late in the process, in fact tiny adjustments after the big jump before generation 200, after which the perturbation in the target weight faded away. In the last generations, we also see that the inertia weight was dominant over the other weights.

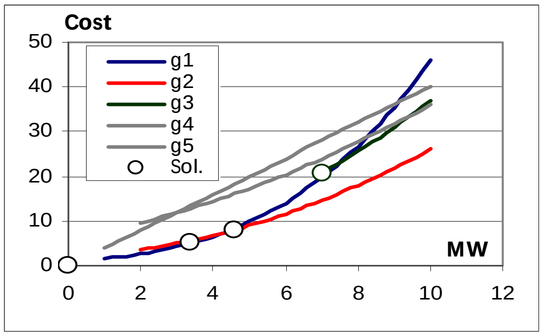

Figure 1 – Cost curves for the unit commitment problem and the best solution with 3 generators dispatched and 2 shut down.

Figure 4 shows the fitness evolution for a dEPSO algorithm and in Figure 5 shows the evolution of weights of a single specific particle in the optimization process. It suggests that the selection procedure indeed distinguished among the roles of the different components of the movement rule.

These are success cases, but how do the EPSO and dEPSO versions compare? We examined how many times an algorithm discovers the optimum value or the optimum neighborhood, in 1000 generations, in 20 runs:

	Av. 20 runs	Best found	No. times opt.
EPSO	34,81575	33,90696	7/20
dEPSO	34,07073	33,90683	18/20

This result suggests that the dEPSO variant deserves to be looked at seriously and new research is suggested. The fact is that the problem of Unit Commitment is one very important problem in Power Systems for the economical implications it has and efficient solutions are still being sought that may compete with classical dynamic programming or branch and bound approaches, that have the known limitations of the curse of dimensionality for real systems.

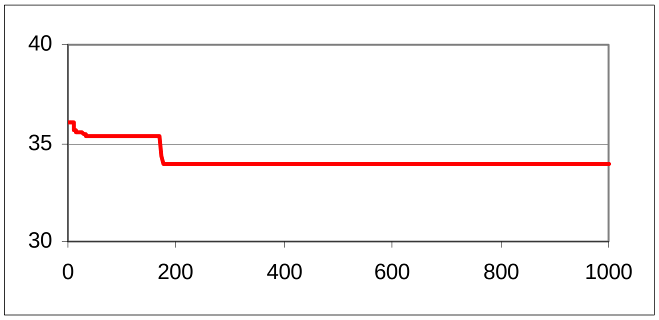

Figure 2 – Evolution of the best fitness during 1000 generations in one run of the EPSO algorithm

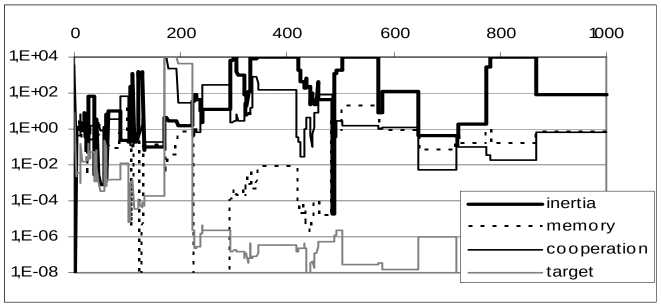

Figure 3 – Evolution of weights of the best particle at each iteration during the unit commitment optimization (1000 generations) using EPSO.

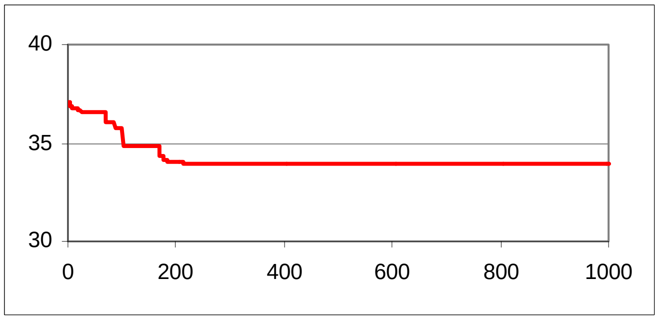

Figure 4 – Evolution of the best fitness during 1000 generations in one run of the dEPSO algorithm

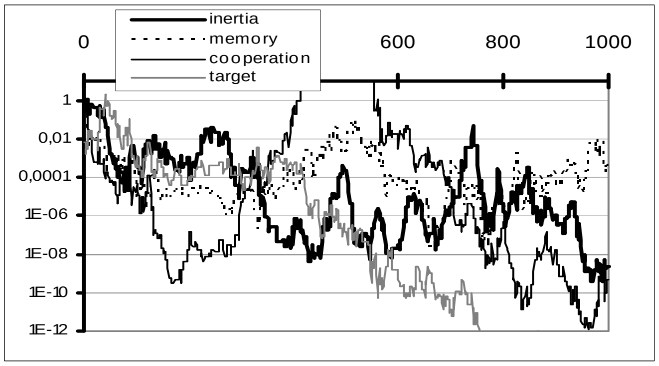

Figure 5 – Evolution of weights of a particle during the unit commitment optimization (1000 generations) using dEPSO.

VI. Optimizing Market Competition

Comparisons of the performance of EPSO with other algorithms have been made in demanding Power System problems by other authors. In [16] the authors compared EPSO with PSO and their adaptive variant of called APSO in co-generation plant operation optimization, with the advantage of EPSO. In [17] one finds a comparison of several methods in voltage stability assessment where EPSO also emerged as the winner.

In this section we will refer to a competition among algorithms that we organized in a very demanding problem, extremely suited for meta-heuristics, given that no analytic solution may be found. It is the case of a simulation of a retail market in energy distribution, where the general objective is to discover a market strategy that will give advantage to a market actor (energy retailer) in a competitive environment.

The simulation was set over the open source intelligent agent platform JADE, which is FIPA compliant.

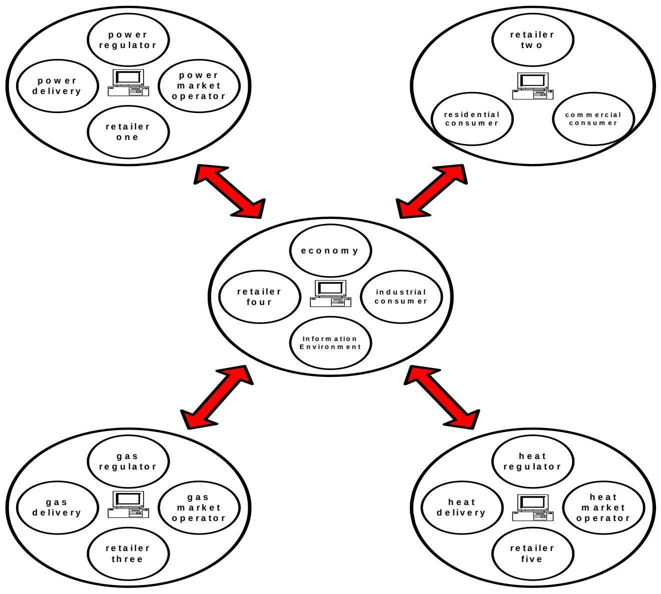

Figure 6 - Parallel arrangement of 19 Agents in a cluster of

5 PCs

We will not describe this platform here but a description may be found in [18]. The simulation is composed of 19 agents running in a parallel arrangement of 5 PCs, as shown in Figure 6.

The agents represented in the platform were:

1 Residential cons. group	1 Commercial cons. group
1 Industrial cons. group	2 Electricity Retail Suppl.
2 Gas Retail Suppliers	1 Heat Retail Supplier
1 Power Delivery comp.	1 Gas Delivery company
1 Heat Delivery company	1 Electricity Regulator
1 Electr. Market operator	1 Gas Market operator
1 Heat Market operator	1 Gas Regulator
1 Heat Regulator	1 Economy agent
1 Information Environm.	

Each agent has an internal logic that commands its decisions, which are influenced by load forecasts, network development, reaction of consumers, prices, investment decisions, regulatory constraints, etc. Agents are assumed independent and from their interaction a complex behavior emerges that determines a dynamic market equilibrium while the simulation develops along time.

One special type of agent is the retailer, which is equipped with a mechanism to predict the behavior of competitors in price setting. But the most unique feature of this agent is that, during the simulation, it performs regularly internal simulations of the market in order to try to optimize its policy for the next moves, acting on variables such as price of energy or money allocated to advertising, increasing company efficiency or investment.

This environment is dynamic because we have equipped the consumer agents with non-linear mechanisms that simulate their response to market conditions. Also, retailers depend on the action of delivery companies, because these agents make decisions about expanding their networks (having a GIS-type representation of the territory) and therefore potentially reaching new consumers.

This is an extremely complex environment to test a number of alternative meta-heuristics to optimize market strategy in the internal simulation of a retailer and to examine which agent (equipped with which algorithm) comes out as a winner (best profits) in a simulation extending for 2 years on a daily basis. We have reported some results in [19][20] and in this paper we add comparisons including classical PSO, together with EPSO and 3 versions of Genetic Algorithms.

The Genetic Algorithm versions were:

SSGA – Steady State Genetic Algorithm: a scheme where for a generation to the following we allowed the replacement of only 80% of the population, keeping the best 20%. It adds elitism to the selection, aiming at preserving in the population the most promising individuals so that they may have more chances of combining their good genes with other individuals.

DCGA – Genetic Algorithm with Deterministic Crowding: Deterministic Crowding is a selection technique where similar individuals in the population are paired before comparisons are made, and elitist selection acts on each pair formed. It aims at preserving diversity in the population.

MPGA – Multi-Population Genetic Algorithm: a scheme where we used two sub-populations, each running a SSGA, but exchanging two randomly selected individuals from one of the sub-populations to another before crossover took place. It also aims at preserving diversity.

The PSO algorithm used was the classical formulation with decreasing function in the inertia weight. The EPSO version was the basic one with Lognormal mutations.

The simulation length was of 24 months for market simulation and of 2 months for retailer internal simulations inside the evolutionary process.

The stopping criterion was the same for all algorithms: when performing the first internal simulation, the evolutionary process was stopped if there were no improvement in the fitness function after 50 consecutive generations; in all the following internal simulations, during the market simulation of 24 months, the limit was 10 generations.

In order to compensate for the influence of random events, we run every simulation 5 times. During market simulation, only one retailer was equipped with one type of EA and other retailers were not optimizing; also the investment function of Delivery agents was disconnected to avoid introducing noise in the simulation and disturb the interpretation of results.

The fitness function was the same in all cases: maximizing profit of the retailer.

In a first test, we have fixed the population for all methods as 20 individuals.

Figure 7 and Figure 8 show the weekly profits made by a retailer agent using PSO and EPSO in five runs. One may observe that EPSO displays robustness in the results. Figure 9 compares the performance of the 5 algorithms and we see that EPSO was the clear winner and MPGA is second best while DCGA was the clear loser.

However, one of the possible reasons of bad performance of some algorithms could be early termination because of the application of the stopping criterion. We see clearly in Figure 10 that the distribution of computing effort was quite unbalanced. In an attempt to have a fair comparison, we have defined an Experiment 2 and changed the size of populations such that EPSO and MPGA were kept constant at 20 individuals while the population size of DCGA, SSGA, and PSO was increased to 100, 80, and 40 respectively.

Figure 11 shows the result as an average of five runs. Clearly, still EPSO is the winner and SSGA emerges now as the second best, while DCGA is confirmed as being the poor performer.

Figure 12 allows us to see that the best run of SSGA outperformed all others but that on average the profits made by a retailer with EPSO are stable on all runs and better than achieved with any other algorithm. Figure 13 clearly shows that now the computing effort of all five algorithms is relatively balanced.

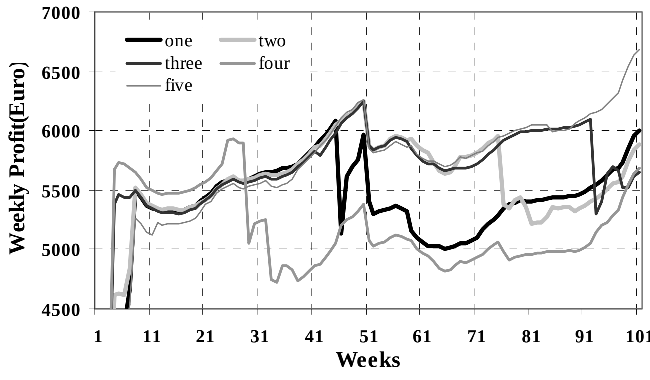

Figure 7 – Weekly profits made by a retailer equipped with PSO in 5 runs.

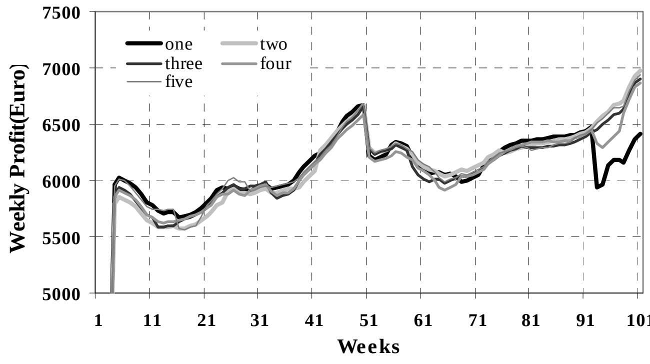

Figure 8 – Weekly profits made by a retailer equipped with EPSO in 5 runs.

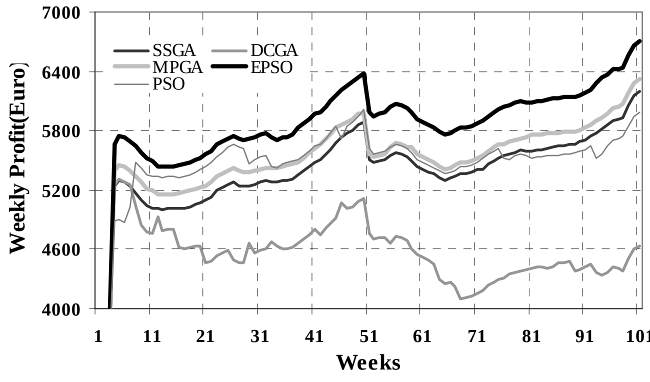

Figure 9 – Average of five runs for all algorithms

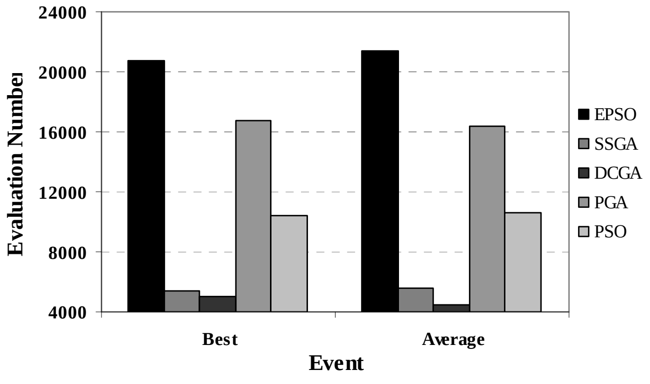

Figure 10 – Number of evaluations in the best run and on average of simulations done by the competing algorithms. Some algorithms experienced early termination because they could not achieve progress in a number of iterations.

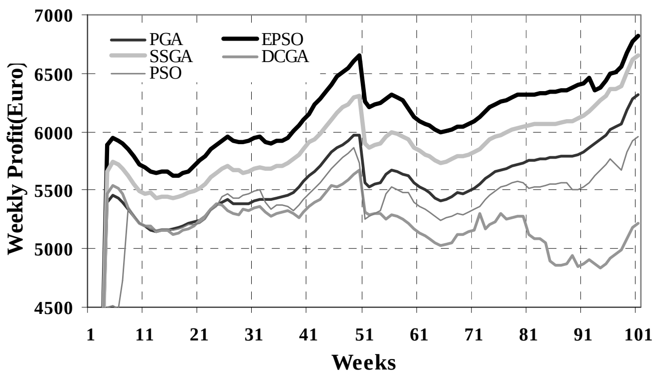

Figure 11 – Average of five runs with different population sizes (Exp. 2)

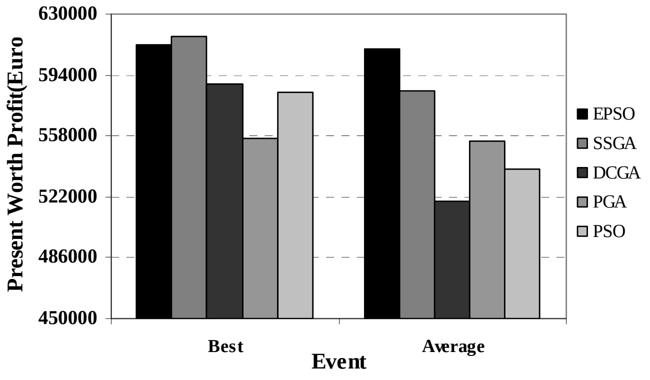

Figure 12 – Accumulated profits by a retailer equipped with each algorithm, for the best run and on the average of five runs of 2 year simulations (Exp. 2).

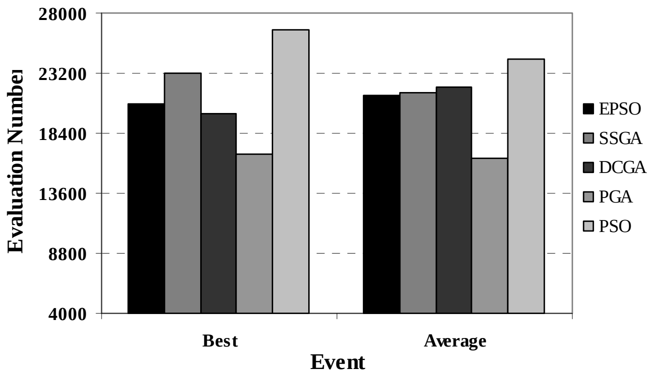

Figure 13 – Number of evaluations in the best run and on average of simulations done by the competing algorithms with different population sizes (Exp. 2).

VII. CONCLUSIONS

This paper presents new ideas for improving the Evolutionary Particle Swarm Optimization algorithm and displays new evidence of the robustness of the method (in achieving consistently the same good results). The two innovations are: a random parameter affecting communication among particles, creating a topology that we call *stochastic star*, which proved valuable in cases where the algorithm needs to escape premature convergence; and a variant inspired in differential evolution that we call dEPSO, which displayed interesting behavior in the mixed integer problem of unit commitment which is a rather difficult practical problem in Power Systems.

The paper provides also new evidence of the interest of the EPSO approach by reporting some results of a competition among distinct meta-heuristics in an extremely complex and dynamic environment: a market simulation platform composed of intelligent agents. The meta-heuristics were used to optimize the market strategy of retailer agents and EPSO emerged as the algorithm that provided competitive advantage to the retailer using it to predict and select market decisions.

These results provide new incentive to further research the potential of the algorithm.

VIII. Acknowledgment

The work reported has been partially financed by FCT, project EPSO POSC/EEA-ESE/60980/2004.

IX. References

[1] V. Miranda and N. Fonseca, "EPSO – Best-of-Two-Worlds Meta-Heuristic Applied to Power System Problems", *Proceedings of WCCI/CEC – World Conference on Computational Intelligence, Conference on Evolutionary Computation*, Honolulu (Hawaii), USA, June 2002

[2] V. Miranda and N. Fonseca, "Reactive Power Dispatch with EPSO - Evolutionary Particle Swarm Optimization", *Proceedings of PMAPS – International Conference on Probabilistic Methods Applied to Power Systems*, Naples, Italy, Set 2002

[3] I. M. S. Dias, N. Fonseca, H. M. Salgado and V. Miranda, "Optical Fibre Bragg Gratings Synthesis by Evolutionary Particle Swarm Optimization", *Proceedings of the Portuguese Physics Conference FISICA 2002*, Évora, Portugal, Sep 2002.

[4] V. Miranda and N. Fonseca, "EPSO - Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems ", Proceedings of IEEE/PES Transmission and Distribution Conference and Exhibition 2002: Asia Pacific, vol.2, pp.745-750, October, 2002

[5] A. Mendonça, N. Fonseca, J. P. Lopes and V. Miranda, "Robust Tuning of Power System Stabilizers Using Evolutionary PSO", presented at ISAP 2003 – Intelligent Systems Applications to Power Systems Conference, Lemnos, Greece, September 2003

[6] J.J. Xu and Z.H. Xin, "An Extended Particle Swarm Optimizer", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium IPDPS'05 – workshop 6, Denver, Colorado, USA, April 2005

[7] G.C. Chen and J.S.Yu, "Enhanced Particle Swarm Optimization and its Application in Soft Sensor" (in Chinese), Control And Decision, vol 20, no. 4, pag. 377-381, WangFa Data (ChinaInfo) and Chinese Electronic Periodical Services CEPS, April 2005

[8] G. Yang and R. Zhang, "An Emotional Particle Swarm Algorithm", in Advances in Natural Computation: Proceedings of the First International Conference ICNC 2005, part III, pag. 553, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2005

[9] H.-P. Schwefel, "Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit", Technical report, Technical University Berlin, 1974

[10] D.B.Fogel, "Evolving Artificial Intelligence", Ph.D. Thesis, University of California, San Diego, 1992

[11] H.-G. Beyer, "Toward a Theory of Evolution Strategies: Self-Adaptation", in *Evolutionary Computation*, vol. 3, no. 3, pp. 311-347, 1996

[12] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948, Perth, Australia, 1995

[13] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", *Proceedings of the 1999 Congress of Evolutionary Computation*, vol. 3, 1951-1957, Washington D.C., USA, IEEE Press, 1999

[14] J. Kennedy, "The Particle Swarm: Social Adaptation of Knowledge", *Proceedings of the 1997 International Conference on Evolutionary Computation*, pp. 303-308, IEEE Press, 1997

[15] R. Storn and K. Price, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", International Computer Science Institute – *Technical Report* TR-95-012, March 1995

[16] S. Kitagawa and Y. Fukuyama, "Comparison of particle swarm optimizations for optimal operational planning of energy plants", Proceedings of the Swarm Intelligence Symposium 2005, pag. 155-161 June 2005

[17] H. Mori and Y. Komatsu, "A Hybrid Method of Optimal Data Mining and Artificial Neural Network for Voltage Stability Assessment", Proceedings of IEEE St. Petersburg PowerTech, Russia, June 2005

[18] N. W. Oo and V. Miranda, "Multi-energy Retail Market Simulation with Intelligent Agents", Proceedings of IEEE St. Petersburg Power Tech, St. Petersburg, Russia, Jun 2005

[19] V. Miranda and N. W. Oo, "Evolutionary Algorithms and Evolutionary Particle Swarms (EPSO) in Modeling Evolving Energy Retailers", *Proceedings of PSCC - 15th Power Systems Computation Conference*, Liège, Belgium, Aug 2005

[20] N. W. Oo and V. Miranda, "Evolving Agents in a Market Simulation Platform – A Test for Distinct Meta-Heuristics", Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems ISAP 2005, pag. 482–487, Arlington (VA), USA, Nov. 2005

X. Biography

Vladimiro Miranda received his Licenciado, Ph.D. and Agregado degrees from the Faculty of Engineering of the University of Porto, Portugal (FEUP) in 1977, 1982 and 1991, all in Electrical Engineering. In 1981 he joined FEUP and currently holds the position of Professor Catedrático. He is also currently Director of INESC Porto. He has authored many papers and been responsible for many projects in areas related with the application of Computational Intelligence to Power Systems.

Naing Win Oo is a researcher at the Power Systems Unit of INESC Porto – Institute of Engineering in Systems and Computers of Porto, Portugal. He obtained his M.Eng. degree in Power System Engineering from AIT - Asian Institute of Technology, Thailand, in 1998. Presently, he is in the PhD program in the University of Porto and developing his thesis work at INESC Porto, and his interests include energy markets, power system planning and control, evolutionary computation and artificial intelligence applications to power systems.