Vol. 00, No. 00, April 2010, 1–23

RESEARCH ARTICLE

A Fast Particle Swarm Algorithm For Solving Smooth and Non-smooth Economic Dispatch Problems

Leticia Cecilia Cagnina, Susana Cecilia Esquivel[1]
and Carlos A. Coello Coello[2]*
[1]*LIDIC (Research Group). Universidad Nacional de San Luis*
Ej. de Los Andes 950. (D5700HHW) San Luis, ARGENTINA
[2]*CINVESTAV-IPN (Evolutionary Computation Group)*
Departamento de Computación, Av. IPN No. 2508
Col. San Pedro Zacatenco, México D.F. 07360, MEXICO
Emails: {`lcagnina,esquivel`}`@unsl.edu.ar` *and* `ccoello@cs.cinvestav.mx`

(*v3.7 released September 2008*)

The Economic Dispatch (ED) problem is a common task in the operational planning of a power system, which requires to be optimized. Such optimization includes two goals: (1) the minimization of the total schedule cost and (2) the minimization of the response time. This paper presents a fast Particle Swarm Optimization (PSO) approach for solving the ED problem. The proposed approach is applied to five case studies, having different size and complexity. Three of them have a smooth fuel cost, whereas the other two have non-smooth load functions, with valve-points. The experimental results indicate that the proposed approach provides good results on the test problems adopted, while requiring a lower computational time than other approaches taken from the specialized literature.

Keywords: Power Systems; Particle Swarm Optimization; Economic Load Dispatch; Smooth Function; Valve-point Loading.

---

*Corresponding author

ISSN: 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215YYxxxxxxxx
http://www.informaworld.com

1. Introduction

The Economic Dispath (ED) problem is widely used and constitutes a very important optimization task that must be undertaken on a daily basis in the operation of power systems. The schedules that solve ED represent the generating outputs for each on-line unit. When a certain time horizon is considered, dynamic constraints for the generators are taken into account and the problem is called *dynamic ED*. Otherwise, when the allocation of the outputs for each committed unit is based on the static behavior of the generators, the problem is called *static ED*. Thus, the dynamic ED problem is an extension of the static one, but this paper, will focus only on the latter (i.e., the static ED problem).

The main goal when dealing with Economic Dispatch tasks is to find the optimal schedule that produces the desired power outputs for all the generating units, and minimizes the total fuel cost ($), while satisfying some constraints. But another goal associated to the previous one, is to obtain the desired schedule as fast as possible.

A variety of traditional optimization approaches have been proposed to solve the ED problem, such as piecewise quadratic cost functions (Lin and Viviani 1984), the lambda iteration method (Bakirtzis *et al.* 1994), quadratic programming (Fan and Zhang 1998) and gradient methods (Naka *et al.* 2001). Other traditional methods such as dynamic programming (Wood and Wollenberg 1984), Lagrangian relaxation (Hindi and Ghani 1991) and linear programming (Song and Yu 1997), have been adopted, too. In recent years, the use of stochastic search techniques has become popular in this area. Researchers have proposed the use of neural networks (Park *et al.* 1993), genetic algorithms (Li *et al.* 1997), evolutionary programming (Park *et al.* 1998), simulated annealing (Ongsakul and Ruangpayoongsak 2001), tabu search (Lin *et al.* 2002) and particle swarm optimization (Victoire and Jeyakumar 2005), among other soft computing techniques.

Although particle swarm optimization (PSO) has been adopted before to solve the ED problem, this paper presents a novel approach which is not only effective, but also faster than other previously proposed soft computing techniques.

The remainder of the paper is organized as follows. Section 2 describes the ED problem. A revision of the most representative previous related work is presented in Section 3. Section 4 describes the proposed PSO for solving ED problems. The experimental setup and the comparison of results with respect to other approaches are presented in Section 5. Finally, the conclusions and some possible paths for future works are presented in Section 6.

2. Problem Statement

In power system operation, the problem of scheduling generating units to meet electricity demands, while considering operational constraints, is called the *Unit Commitment* (UC) problem (Wood and Wollenberg 1984).

A subproblem of the Unit Commitment problem is the Economic Dispatch (ED) problem, in which the optimum scheduling for the generating units during a certain period of time, has to be determined. That schedule has to minimize the total cost of production and involves the satisfaction of equality constraints (power balance) and inequality constraints (operating limits). The mathematical model of the Economic Dispatch problem is defined as follows (Wood and Wollenberg 1984):

$$\text{minimize TC} = \sum_{i=1}^{N} F_i(P_i) \tag{1}$$

subject to:

$$\sum_{i=1}^{N} P_i = P_D \qquad \text{(Power Balance Constraint)} \tag{2}$$

and

$$Pmin_i \leq P_i \leq Pmax_i \qquad \text{(Operating Limit Constraints)} \tag{3}$$

where TC is the total cost of production, F_i is the fuel cost of the $i-th$ generator, P_i the power output of the $i-th$ generator, N is the number of generators, P_D is the power demand required, $Pmin_i$ and $Pmax_i$ are the minimum and maximum power output of the $i-th$ generator, respectively.
In (Swarup and Yamashiro 2002) more constraints are added to the classical ED problem: the ramp rate limits (equation (6)) and the prohibited operating zones (equation (7)). The ramp rates constraints restrict the operating range of all on-line units. The prohibited zones constraints restrict the operation of the units due to steam valve operating or vibrations in shaft bearing. Also, the total system generation includes the transmission network loss (P_L). Then, in (Swarup and Yamashiro 2002) equation (2) is replaced with equation (4).

$$\sum_{i=1}^{N} P_i = P_D + P_L \qquad \text{(Power Balance with Transmission Loss)} \tag{4}$$

The P_L value is calculated with a function (equation (5)) of unit power outputs that uses a loss coefficients matrix B, a vector B0 and a value B00 (Gaing 2003).

$$P_L = \sum_{i=1}^{N}\sum_{j=1}^{N} P_i \mathrm{B}_{ij} P_j + \sum_{i=1}^{N} \mathrm{B0}_i P_i + \mathrm{B00} \tag{5}$$

$$\max(Pmin_j, P_j^0 - DR_j) \leq P_j \leq \min(Pmax_j, P_j^0 + UR_j) \qquad \text{(Ramp Rate Limits)} \tag{6}$$

where P_j^0 is the previous output power of unit j (in MW) and, UR_j and DR_j are the up-ramp and down-ramp limits of the unit j (in MW/h), respectively.

$$\left.\begin{array}{l} Pmin_j \leq P_j \leq P^l_{j,1} \\ P^u_{j,k-1} \leq P_j \leq P^l_{j,k} \\ P^u_{j,nj} \leq P_j \leq Pmax_j \\ \text{with} \qquad k = 2, 3, \ldots, nj \\ \text{and} \qquad j = 1, \ldots, N \end{array}\right\} \quad \text{(Prohibited Operating Zones)} \qquad (7)$$

where $P^l_{j,k}$ and $P^u_{j,k}$ are the lower and upper bound of the prohibited zone k corresponding to the unit j, and nj is the number of prohibited zones of the jth unit.

Depending on the characteristics of the function that defines the fuel cost F, it can be categorized as smooth or non-smooth. Each of them defines a different type of problem, as is explained next:

a. **ED problem with a smooth cost function**

It is the most simplified cost function for the ED problem. It can be represented as a single quadratic function (Wood and Wollenberg 1984):

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i \qquad (8)$$

with a_i, b_i and c_i, being the fuel consumption cost coefficients of the $i-th$ generator.

b. **ED problem with a non-smooth cost function**

Some valve-points effects are commonly observed in this problem and, therefore, the objective function includes multiple non differentiable points. This feature makes the problem more difficult to solve, and the task of reaching the global minimum with guaranteed convergence still remains unsolved (Sinha *et al.* 2003). Compared with the smooth cost function, this one has very different input-output curves, and sinusoidal functions are added (Walters and Sheble 1993):

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |e_i sin(f_i(Pmin_i - P_i))| \qquad (9)$$

with e_i and f_i being the fuel cost coefficients of the $i-th$ unit with valve-point effects.

Figure 1 illustrates examples of a smooth function and a non-smooth function with valve-points loading.

In this paper, both types of functions will be considered: smooth with 3, 6 and 15 generating units (cases A, D and E, respectively), and non-smooth with 3 and 40 generating units (cases B and C, respectively).

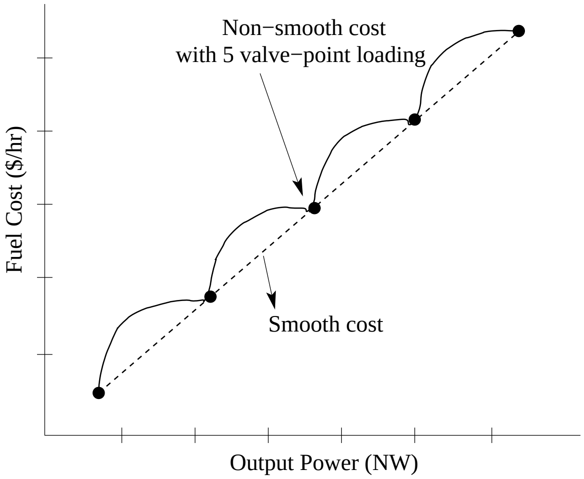

Figure 1. Examples of smooth and non-smooth functions.

3. Previous Related Work

Park et al. (1993) presented a method based on the Hopfield Neural Network to solve the economic power dispatch problem. The Hopfield model has been successfully applied to the solution of several difficult optimization problems, but it frequently requires a large number of neurons to represent a large numerical value. The authors proposed a variant of this method, which uses a single neuron to represent a large value, named, for short, MHNN. The algorithm was tested in three cases, from which only one of them is considered for the purposes of this paper: case A. For that test case, the algorithm obtained a total cost slightly better than the best known solution for the problem under study, but the total power demanded did not reach the 850 MW required by the specifications of the problem. The conclusion was that MHNN was simpler than other methods and that the results achieved were very close to those obtained using numerical methods.

Walters and Sheble (1993) proposed a Genetic Algorithm (GA) for the ED problem. The proposed method incorporates payoff information of candidate solutions in the evaluation process. That allows the use of any type of unit cost curve (i.e., any sort of data characteristics (smooth or not)) to be optimized. The authors performed four independent runs to obtain the final results of the test case B adopted here. The solutions found were very close to the optimum, but an additional initial effort was required for fine-tuning the parameters of the algorithm, particularly the scaling parameter selection, and this could evidently become a major obstacle when solving more difficult problems.

Yang et al. (1996) proposed the use of Evolutionary Programming (EP) in the nonlinear and discontinuous economic dispatch problem. They justified the use of EP over other methods such as Lagrange multipliers, dynamic programming, simulated annealing and genetic algorithms. They used the test case B adopted here, obtaining good results in a reasonably short computational time.

Gaing (2003) proposed a classical Particle Swarm Optimization algorithm to solve the ED problem. The author defined an evaluation function combining the ED cost function with the power balance constraint. The algorithm (named here as PSO-Gaing) penalized with a large positive constant all infeasible individuals. PSO-Gaing was tested with case D and E adopted here. The results obtained were compared with respect to those generated by a Genetic Algorithm and, for both cases, PSO-Gaing outperformed such approach.

Sinha et al. (2003) studied several methods to solve the ED problem, based on Evolutionary Programming (EP) techniques. Several modifications to the original EP method were proposed in order to improve its robustness and speed. In order to achieve that, they relied on two main features: a scaled cost and an empirical learning rate. The EP variants that resulted from this study were: Classical Evolutionary Programming with Gaussian mutation (CEP), Fast Evolutionary Programming with Cauchy mutation (FEP), Improved Fast Evolutionary Programming with the best between Gaussian and Cauchy mutations (IFEP) and, Fast Evolutionary Programming with a weighted mean of the Gaussian and Cauchy mutations (MFEP). The performance of the approaches was tested with cases B and C presented here. They concluded that FEP worked better than the other algorithms when using small problems (for instance, case B). For case C, which is a more difficult problem, IFEP offered the best performance.

Liu and Cai (2005) introduced a reduced complexity Taguchi Method (TM) based on orthogonal arrays. The cost function was minimized recursively satisfying the constraints of the problem. In the TM, a limited amount of computation is required to optimize the objective function. The authors tested the algorithm with cases B and C, presented here. The results found were satisfactory and they concluded that TM was less sensitive to its initial parameter values, and that the results were better than those obtained with other algorithms.

Park et al. (2005) proposed a Modified Particle Swarm Optimization (MPSO) algorithm to solve the economic dispatch problem with non-smooth cost functions. The modification was a particular treatment of the constraints of the ED problem when each individual has moved to another place in the search space. The authors adopt a dynamic search-space reduction strategy to speed up convergence. This strategy is adopted when no improvement is detected after a certain number of iterations. Such strategy is based on the distance between the best position of the swarm and the boundaries defined by the inequality constraints. The MPSO algorithm was tested with cases B and C presented here. MPSO reached the optimum in case B and values close to the optimum for case C (these values were better than those produced by other heuristics).

Sriyanyong et al. (2007) hybridized a Particle Swarm Optimization (PSO) algorithm with a modified heuristic search approach (the approach was called IPSO), to solve the ED problem. The PSO algorithm was adopted to find global solutions and the modified heuristic was used to handle the constraints (both equalities and inequalities) of the problem. The heuristic search method adopted by the authors was originally introduced in (Park *et al.* 2005) but they introduced some changes to enhance its performance. A simulation was made with three of the cases reported here: cases A, B and C. The results indicated that the proposed approach could reach the global optimum with a reasonably low computational time, for the first two cases. For case C, the result was good, but not optimal. However, such result was achieved at a reduced computational time. This led the authors to conclude that the proposed approach was more powerful than the other techniques with respect to which it was compared.

Subramani et al. (2008) modified a PSO (modPSO) algorithm to solve the ED problem with non-smooth cost functions. They enhanced the search process by using randomly chosen particles from the population to update the velocity of the particles with a decreasing probability. The algorithm was validated using two examples from which only one is considered here: Case A. ModPSO found a total generation cost almost as good as that obtained by the other approaches with respect to which it was compared. The authors concluded that their ModPSO is a promising alternative for solving the ED problem with valve-points and multi-fuel effects.

Khamsawang et al. (2009) proposed a differential evolution (DE) algorithm for the ED problem. A regenerating population mechanism was introduced to the standard DE algorithm, in order to improve its ability to escape from local optima. The authors adopted the Case B considered in this paper. The authors concluded that the changes introduced allowed DE to improve its performance in the ED problem solved.

Sun et al. (2009) modified a quantum-behaved particle swarm optimization algorithm by adding to it a differential mutation operator to enhance its global search ability. The proposed algorithm (QPSO-DM) was validated using cases D and E adopted here, and the results were compared with respect to those generated by a genetic algorithm, a differential evolution approach and a PSO method. The authors concluded that QPSO-DM produced high quality solutions, and that its behavior was robust while also exhibiting good convergence properties with respect to those of the algorithms adopted in the study.

4. A Fast PSO Algorithm for the ED Problem

Here, a fast Particle Swarm Optimization Algorithm to solve ED problems (fast-CPSO, for short) is proposed, which inherits some features from the classical PSO model (Eberhart and Kennedy 1995). The proposed approach also incorporates new features which aim to improve its performance, namely, a bi-population and a shake mechanism.

4.1. *The standard PSO algorithm*

The PSO algorithm operates on a population (called *swarm*) of individuals (called *particles*) which encode the possible solutions to the problem to be solved. Such particles consist of vectors of real numbers that represent velocities and positions. The size of each vector determines the *dimension* of each particle. The algorithm iterates searching for better solutions and stores the best position found so far either by the entire population (for the "global best" or *gbest* model) or within a certain neighborhood defined by an interconnection topology (for the "local best" or *lbest* model). The best value reached by each particle (called "personal best" or *pbest*) is also stored. The particles evolve using two update formulas, one for the particle's velocity and another for its position, in the following way:

$$v_{id} = \mathcal{X}(v_{id} + c_1 r_1 (p_{id} - par_{id}) + c_2 r_2 (pg_d - par_{id})) \tag{10}$$

$$par_{id} = par_{id} + v_{id} \tag{11}$$

where v_{id} is the velocity of the particle i at the dimension d, $\mathcal{X}$ is the constriction factor (Clerc and Kennedy 2002) which aims to balance the global exploration and the local exploitation of the swarm, c_1 is the personal learning factor, and c_2 the social learning factor. r_1 and r_2 are two random numbers within the range [0, 1], which are used to introduce a stochastic value to determine how much of each factor is added. p_{id} is the best position reached by the particle i and pg_d is the best position reached by any particle in the entire swarm (*gbest* model). par_{id} is the value of the particle i at dimension d.

4.2. *The fast-CPSO approach*

Next, the modifications introduced in the proposed approach to the standard PSO model are described.

4.2.1. *Updating velocity and particle vectors*

The equation for updating the particles was modified. Instead of using equation (11) in all the iterations, it is selected with a probability of 0.925 (this value was empirically derived). The rest of the time the equation (12) is selected, which is based on Kennedy's proposal (Kennedy 2003). In this case, the position of each particle is randomly chosen from a Gaussian distribution with the mean selected as the average between the best position recorded for the particle and the best in the swarm. The standard deviation is the difference between these two values.

$$par_i = N\left(\frac{p_i + pg}{2}, |p_i - pg|\right) \tag{12}$$

where par_i is the particle to be updated, N is the Gaussian random generator, p_i and pg are the best position reached by the ith particle and the best position reached by any particle in the swarm, respectively. The probability of selection adopted was empirically found to be the best, after performing a series of experiments on the test functions adopted.

4.2.2. *Handling Constraints*

From the wide variety of constraint-handling techniques currently available for evolutionary algorithms (Coello Coello 2002), a very simple scheme is adopted here. The constraint-handling method follows the rule: "a feasible particle is preferred over an infeasible one". Particles are compared in pairs. When one of them is feasible and the other one is infeasible, then the feasible particle is chosen. When the two particles compared are infeasible, the one closer to the feasible region is chosen. In order to do that, the algorithm stores the largest violation obtained for each constraint at each generation. When an individual is found to be infeasible, the sum of its constraints violations (this value is normalized with respect to the largest violation stored so far) is the one considered as its distance to the feasible region. This constraint-handling scheme is used when the *pbest* and *gbest* particles are chosen.

4.2.3. *A bi-population into the swarm*

In the proposed approach, a bi-population scheme is introduced. The idea of adopting several swarms rather than only one as the population of a PSO algorithm is not new, since several authors have used similar schemes (Liang and Suganthan 2005, Blackwell and Branke 2006, Daneshyari 2006, Zhao *et al.* 2008, Trojanowski 2008). In most of the previous works, the authors adopted several small and dynamic swarms which are frequently regrouped at different stages during the search. Also, in all of these previous cases, the swarms exchange information. Here, this concept of using more than one swarm is adopted in a different way.

In the proposed fast-CPSO algorithm, the idea is maintain more than one group of particles exploring the search space at the same time. The aim of this is to reduce the possibility of getting trapped in local optima. In the algorithm the entire population is split into only two static subpopulations, each of which is independently evolved.

Unlike the works previously indicated, in the proposed approach, no information is shared between the two swarms. Also, the size of the swarms throughout the search does not change.

A question that naturally arises is why not to adopt more than two subpopulations. The reason is purely pragmatic: since the number of particles used in the population of the algorithm is small, it would be inappropriate to adopt more than two populations. All the features stated before for the entire population remain applicable, but in this case, they are applied not to a single population, but to each subpopulation. When the iterative process finishes, the best particle from both subpopulations is reported as the final output.

4.2.4. *Shake Mechanism*

A *shake mechanism* is introduced, which is adopted to overcome potential stagnation problems that can arise when exploring the vicinity of the global optimum (Cagnina *et al.* 2006). In order to implement such a shake mechanism, some particles are moved to a different place in the search space. Although this can be done by guiding a particle to a random direction, it is undesirable that the particles move away too much (we just want to shake them a little). So, a particle with a good solution is selected as a reference: a randomly chosen *pbest* particle (pb_{SELd}). Thus, equation (13) is used to move a particle i:

$$v_{id} = \mathcal{X} v_{id} + c_1 r_1 (pb_{SELd}) \tag{13}$$

where v_{id} is the dth-position of the velocity vector, $\mathcal{X}$ is the constriction factor, c_1 is the personal learning factor multiplied by r_1, which is a random number within the range [0, 1]. pb_{SELd} is the dth-position of a (randomly chosen) selected *pbest* vector.

This mechanism is executed when the percentage of infeasible individuals is higher than 10% (this value was empirically derived), with a 50% probability over all the individuals in the swarm, at each iteration. It is worth noticing that it is not convenient to keep populations in which all the solutions are feasible, since infeasible solutions play an important role when trying to solve problems with active constraints, since they allow us to explore the boundary between the feasible and infeasible regions. Figure 2 illustrates this mechanism.

4.3. ***Pseudocode of the proposed fast-CPSO***

Figure 3 shows the pseudocode of the proposed algorithm. At the beginning of the search, the vectors of position and velocity of each particle in both subpopulations are randomly initialized (lines 2 to 5). After evaluating the particles (line 6) and obtaining the best values: *pbest* and *gbest* (line 7), the subpopulations begin to evolve. During the evolutionary process, new values of *pbest* and *gbest* are chosen and both, the velocity and the position of each particle, are updated (lines 8 to 20). At line 21, a *bounding* mechanism is applied, to control that all the dimensions in all particles are within the allowable bounds. Then, the percentage of infeasible individuals in both subpopulations is calculated and the shake mechanism is applied if the required conditions are fulfilled (lines 22 to 25). After that, the particles are evaluated and new "best" values are recorded (lines 26 and 27). All the process is repeated until the stop condition is reached. Finally, the best value reached by any subpopulation is taken and compared. The best of them is

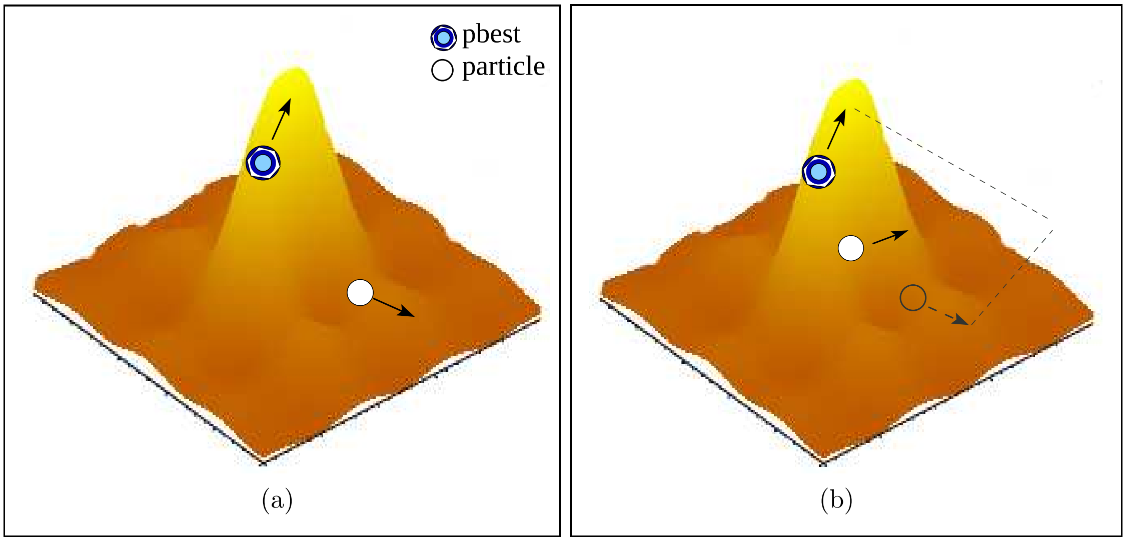

Figure 2. (a) The *pbest* particle close to the optimum and a particle with a stagnation problem. (b) The same particle, after applying the Shake mechanism, is now close to *pbest*.

returned (lines 29 and 30).

5. Experimental Study

For validating the proposed approach, its performance is assessed using five different problems. The first, called **case A**, is a traditional Economic Dispatch problem (Wood and Wollenberg 1984) with a 3-unit system, a smooth function and, two constraints: power balance (equation (2)) and operating limits (equation (3)). **Case B**, is a non-smooth standard 3-unit system, considering a valve point loading function (Walters and Sheble 1993) with power balance (equation (2)) and operating limits (equation (3)) constraints. **Case C**, consists of 40 generating units adapted from (Chen and Chang 1995), with some modifications to incorporate a valve point loading function (Sinha *et al.* 2003), and constrained by power balance (equation (2)) and operating limits (equation (3)). **Case D** is a system with 6 thermal units, 26 buses, 46 transmission lines (Yoshida *et al.* 2000), a smooth cost function and, constraints defined by equations (4), (6) and (7). **Case E** is a system with 15 thermal units (Lee and Breipohl 1993), a smooth cost function and constraints defined by equations (4), (6) and (7). The complete data for the five systems is shown in the Appendix.

The experiments were executed in a personal computer having an AMD Athlon processor running at 1.24GHz with 512MB of RAM. This data is relevant because the computational time required to reach each of the results reported next, is almost as important as the total cost itself. The underlying assumption is that the response time is crucial in these problems, particularly when they are as difficult as those depicted in cases C and E.

Next, a short explanation of the parameters settings adopted by the fast-CPSO is presented, as well as each of the case studies under consideration. The results obtained by the fast-CPSO and their comparison with respect to those obtained by other algorithms,

```
fast-CPSO:
Swarm Initialization
  Initialize subpop1
  Initialize velocity for subpop1
  Initialize subpop2
  Initialize velocity for subpop2
  Evaluate fitness for each subpop
  Record pbest and gbest for each subpop
  DO
    FOR each subpop DO
        FOR i=1 TO numberOfparticles DO
            FOR j=1 TO numberOfdimensions DO
                Update $vel_{ij}$
                IF probability>$(0.075)$
                   Update $part_{ij}$ with eq.(11)
                ELSE
                   Gaussian update with eq.(12)
                END
            END
        END
    END
    Bounding particles
    Calculating % infeasible
    IF % infeasible > 10%
       shake-mechanism
    END
    Evaluate fitness($part_i$)
    Record pbest and gbest
  WHILE($current\_cycle < max\_cycle$)
  result=BEST($best\_subpop1$,$best\_subpop2$)
  RETURN(result)
```

Figure 3. Pseudocode of the proposed fast-CPSO.

is also shown.

5.1. *Parameters Settings*

In order to avoid premature convergence, the constriction factor $\mathcal{X}$ was adopted to regulate the velocity and to maintain a proper behavior of the PSO algorithm during the search (Clerc and Kennedy 2002). For that sake, the relationship between the constriction factor $\mathcal{X}$ and the learning factors c_1 and c_2 was studied. After performing such a study, only one of the learning factors was set and the same value was assigned to the other one, because the aim was to express an equal preference for both learning factors (c_1 and c_2). Based on the recommendations from (Kennedy and Eberhart 2001), the values of these learning factors were set within the range [1.4, 1.9]. Thus, $c_1 = c_2 = 1.8$

were adopted. The constriction factor was set to $\mathcal{X} = c_1 - 1.0$ in all cases, so that it could vary within the range [0.4, 0.9] as suggested in (Clerc and Kennedy 2002). Since $c_1 = 1.8$, then $\mathcal{X} = 0.8$.

In order to select the probability distribution to be adopted for updating the particles, a parameter which defines the probability of selecting one distribution over the other was used. In this case, an empirical study to analyze the impact of the probability distribution on the performance of the proposed approach was performed. It was found that the use of a Normal distribution was beneficial in most cases. As a consequence of this study, the probability of selecting a Normal distribution was set in 92.5%. Consequently, the probability of adopting a Gaussian distribution is 7.5%.

5.2. *Test Case A: 3-generator system with a smooth function*

The population size and the number of cycles were set according to the total number of evaluations spent by (Sriyanyong *et al.* 2007) (i.e., 3,000), in order to allow a fair comparison of results. The size of the swarm was set to 10 particles and the maximum number of iterations was set to 300. Only 4 independent runs were performed to obtain the best value to be compared with that reported in (Sriyanyong *et al.* 2007). This is the same number of runs performed by (Sriyanyong *et al.* 2007). However, 10 independent runs were also done because performing less runs does not provide information of statistical significance to reach conclusions about the performance of a stochastic algorithm. In both cases (when using 4 and 10 independent runs) the proposed algorithm reached the global optimum with a standard deviation of zero. The demand of power is 850 MW and the global solution for this system (Wood and Wollenberg 1984) has a total cost of \$8,194.35612. In Table 1, a comparison of results with respect to different methods is presented: a Numerical Method (Wood and Wollenberg 1984) (NM), a Modified Hopfield Neural Network (Park *et al.* 1993) (MHNN), Improved Evolutionary Programming (Park *et al.* 1998) (IEP), three Particle Swarm Optimization algorithms: Modified Particle Swarm Optimization (MPSO) (Park *et al.* 2005), another PSO variant called ModPSO (Subramani and Rajeswari 2008), and an Integrated Particle Swarm Optimization (IPSO) algorithm (Sriyanyong *et al.* 2007). Column 1 names the methods evaluated. Column 2 shows the values for the three generating units obtained by each method. Column 3 shows the average time spent by each approach in finishing a complete run (expressed in seconds), and column 4 indicates the total power generated and, at the end, the total cost (in \$) of the solution obtained by each algorithm. Note that "N/A" stands for "Not Available".

All the methods compared (except for MHNN and ModPSO) were able to reach the total power demanded with the minimum cost, although there is another factor that can be considered to be as important as the cost: the computational time required to reach a good solution, which can be critical in real-world applications. As can be observed in Table 1, only IPSO and the fast-PSO can be compared in a fair manner, since they were both run in computers with similar characteristics. The proposed approach turned out to be several times faster than IPSO in test case A.

In Figure 4, the evolution curve for test case A is presented. The vertical axis shows the distance between the best values reached (the average) and the global optimum (or best known value) over the 10 runs, and the horizontal axis shows the number of iterations performed. As can be observed in Figure 4, the average error is extremely low at the beginning of the search process and decays to zero at approximately generation 150, which is when the algorithm has reached the optimum value.

Table 1. Comparison of the **best** values obtained by the proposed fast-CPSO and other approaches for test case A.

Method	U1	U2	U3	Exec. Time	Power(MW)	Total Cost($)
NM (Wood and Wollenberg 1984)	393.170	334.604	122.226	N/A	850	8,194.3561
MHNN (Park *et al.* 1993)	393.800	333.100	122.300	60[a]	849.2	8,187.0000
IEP (Park *et al.* 1998)	393.170	334.603	122.227	N/A	850	8,194.3561
MPSO (Park *et al.* 2005)	393.170	334.604	122.226	N/A	850	8,194.3561
IPSO (Sriyanyong *et al.* 2007)	393.170	334.604	122.226	0.42[b]	850	8,194.3561
ModPSO (Subramani and Rajeswari 2008)	392.361	334.985	122.654	N/A	850	8,194.4000
fast-CPSO	393.170	334.604	122.226	**0.01**[c]	850	8,194.3561

[a] Simulation time using an IBM PC-386.

[b] Simulation time using a Pentium IV processor running at 3GHz and with 512MB of RAM.

[c] Simulation time using an AMD Athlon processor running at 1.24GHz with 512MB of RAM.

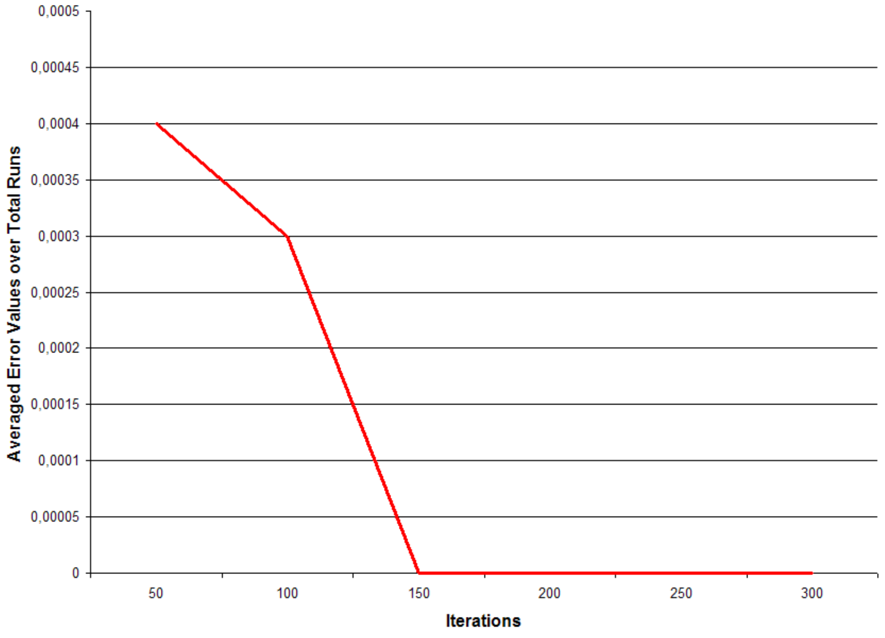

Figure 4. Evolution curve of fast-CPSO for test case A.

5.3. *Test Case B: 3-generator system with a valve-point loading function*

As in the previous case, the population size and the number of cycles was set according to the total number of evaluations reported by (Sriyanyong *et al.* 2007) (6,000 in this case). The size of the swarm was set to 20 particles and a maximum number of iterations of 300 was adopted. Again, for case B, 4 independent runs were performed to obtain the best value to be compared with respect to the one reported in (Sriyanyong *et al.* 2007). However, as before, fast-CPSO also performed 10 independent runs in order to have results with a better statistical significance. For both cases, the proposed approach reached

Table 2. Comparison of the **best** values obtained by the proposed fast-CPSO and other approaches for test case B.

Method	U1 | U2 | U3	Exec. Time	Power(MW)	Total Cost($)
GA (Walters and Sheble 1993)	300.00 | 400.00 | 150.00	16[a]	850	8,237.60
EP (Yang *et al.* 1996)	300.26 | 400.00 | 149.74	0.09[b]	850	8,234.07
IEP (Park *et al.* 1998)	300.23 | 400.00 | 149.77	N/A	850	8,234.09
TM (Liu and Cai 2005)	300.27 | 400.00 | 149.73	N/A	850	8,234.07
MPSO (Park *et al.* 2005)	300.27 | 400.00 | 149.73	N/A	850	8,234.07
IPSO (Sriyanyong *et al.* 2007)	300.27 | 400.00 | 149.73	0.5[c]	850	8,234.07
DE (Khamsawang and Jiriwibhakorn 2009)	300.27 | 400.00 | 149.73	0.30[d]	850	8,234.07
fast-CPSO	300.27 | 400.00 | 149.73	**0.02**[e]	850	8,234.07

[a] Simulation time using an IBM Model.

[b] Simulation time using a PC-486.

[c] Simulation time using a Pentium IV processor running at 3GHz with 512MB of RAM.

[d] Simulation time using an Intel Core 2 Duo 3.0GHz with 4GB of RAM.

[e] Simulation time using an AMD Athlon processor running at 1.24GHz with 512MB of RAM.

the global optimum of $8,234.07 (Yang *et al.* 1996). The demand of power is 850 MW. In Table 2, it can be observed a comparison with different methods, that is, a Genetic Algorithm (Walters and Sheble 1993) (GA), basic Evolutionary Programming (EP) (Yang *et al.* 1996), Improved Evolutionary Programming (IEP) (Park *et al.* 1998), the Taguchi Method (TM) (Liu and Cai 2005), Modified Particle Swarm Optimization (MPSO) (Park *et al.* 2005), an Integrated Particle Swarm Optimization (IPSO) (Sriyanyong *et al.* 2007) and a Differential Evolution (DE) approach (Khamsawang and Jiriwibhakorn 2009). The meanings of the columns are the same as in case A.

In this case, all the methods evaluated (except for the GA) reached the total power demanded with the minimum cost (or a value very close to it). As in the previous example, by observing Table 2, it should be clear that only IPSO and the fast-CPSO can be compared in terms of their computational time, because they were both run in similar computers. The fast-CPSO was again faster than IPSO, but it is worth noting that in (Yang *et al.* 1996), a very good execution time (0.09 seconds) was also reported. This value is better than the one reported in (Sriyanyong *et al.* 2007), considering the characteristics of the computers used in both cases.

In Figure 5, the evolution curve for test case B is presented. As can be observed, the average error is low at the beginning of the search process and decays to zero at the last iterations of the algorithm.

5.4. *Test Case C: 40-generator system with a valve-point loading function*

This is a more difficult problem and, therefore, more evaluations are required for the optimization process. Again, the number of evaluations adopted here is the reported in (Sriyanyong *et al.* 2007) for this problem (i.e., 90,000). The size of the swarm was set to 60 particles and the maximum number of generations was set to 1,500. In this case, a total of 20 independent runs were performed. The best known solution for this problem is $122,190.63, and was reported by (Sriyanyong *et al.* 2007). However, because of the non-smoothness of the function, a better solution could exist. The power demand

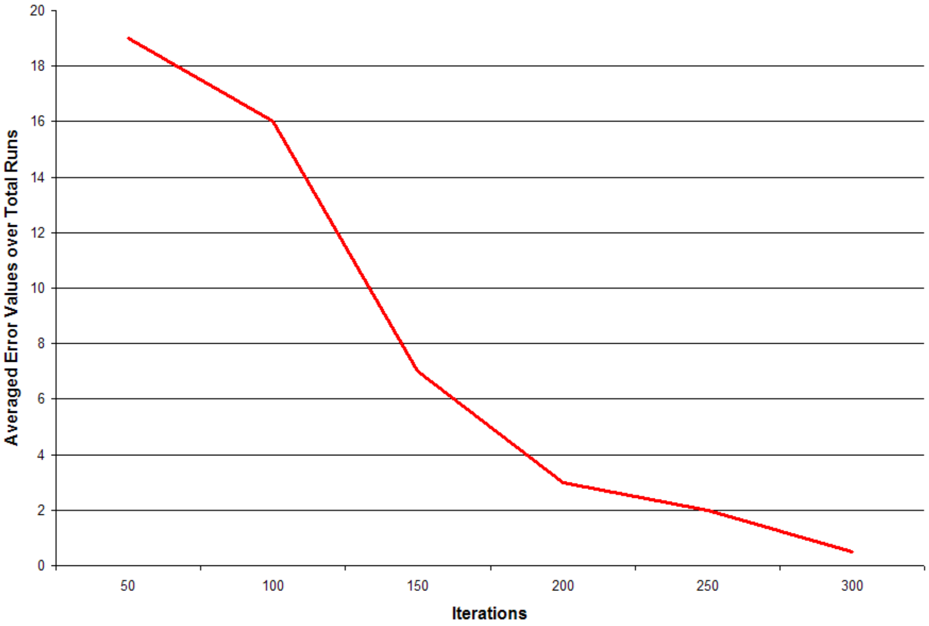

Figure 5. Evolution curve of fast-CPSO for test case B.

is 10,500 MW. In Table 3, it can be observed a comparison of results among the proposed approach and several other methods: Classical Evolutionary Programming (CEP) (Sinha *et al.* 2003), Fast Evolutionary Programming (FEP) (Sinha *et al.* 2003), Modified Fast Evolutionary Programming (MFEP) (Sinha *et al.* 2003), Improved FEP (IFEP) (Sinha *et al.* 2003), the Taguchi Method (TM) (Liu and Cai 2005), Modified Particle Swarm Optimization (MPSO) (Park *et al.* 2005) and Integrated Particle Swarm Optimization (IPSO) (Sriyanyong *et al.* 2007). The column 1 of Table 3 shows the methods evaluated. Column 2 shows the mean cost which is the average over the 20 runs executed. Column 3 presents the average time for a single run (in seconds) and the last column indicates the minimum cost achieved ($).

All the algorithms compared were able to reach the total power demanded. The minimum total costs vary a little, but the best value was obtained by the fast-CPSO. Regarding the computational cost, it is fair to compare the proposed approach only with respect to the algorithm reported in (Sriyanyong *et al.* 2007), since that is the only one that was run in a similar computer. As before, the proposed algorithm had a lower computational time than IPSO, while also obtaining the lowest cost among all the approaches under comparison.

In Table 4, the values obtained for each unit together with their corresponding costs, are shown.

The evolution curve for test case C is shown in Figure 6. In that figure, it can be observed that the average error is high at the beginning of the search process and decays as the search progresses. Towards the end of the execution, the error is of approximately 900.

Table 3. Comparison of the **best** values obtained by the proposed fast-CPSO and other approaches for test case C.

Method	Mean Cost($)	Exec. Time	Min. Total Cost($)
CEP (Sinha *et al.* 2003)	124,793.48	$1,956.96^{a}$	123,488.29
FEP (Sinha *et al.* 2003)	124,119.37	$1,039.16^{a}$	122,679.71
MFEP (Sinha *et al.* 2003)	123,489.74	$2,196.19^{a}$	122,647.57
IFEP (Sinha *et al.* 2003)	123,382.00	$1,167.35^{a}$	122,624.35
TM (Liu and Cai 2005)	123,078.21	94.28^{b}	122,477.78
MPSO (Park *et al.* 2005)	N/A	N/A	122,252.26
IPSO (Sriyanyong *et al.* 2007)	122,304.70	14.56^{c}	122,190.63
fast-CPSO	122,937.22	**3.36**^{d}	**122,102.00**

[a] Simulation time using a Pentium II processor running at 350MHz with 128MB RAM.

[b] No information is available about the hardware platform adopted.

[c] Simulation time using a Pentium IV processor running at 3GHz with 512MB of RAM.

[d] Simulation time using an AMD Athlon processor running at 1.24GHz with 512MB of RAM.

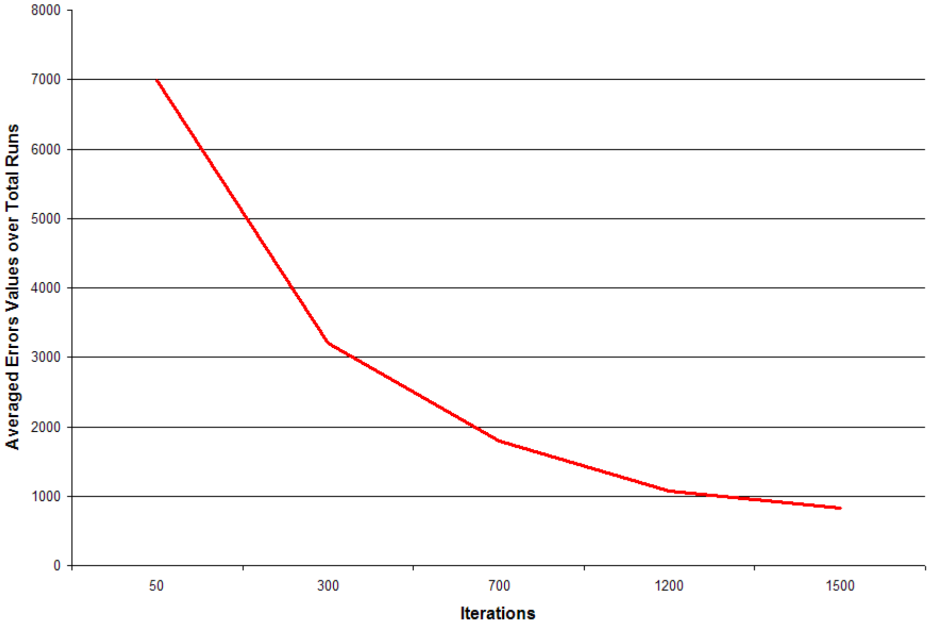

Figure 6. Evolution curve of fast-CPSO for test case C.

5.5. *Test Case D: 6-generator system with a smooth function*

The population size and the number of cycles were set in this case according to the total number of evaluations spent by (Sun *et al.* 2009) (i.e., 20,000), in order to allow a fair comparison of results. The size of the swarm was set to 100 particles and the maximum number of iterations was set to 200. 100 independent runs were performed to obtain

Table 4. Best results obtained by the fast-CPSO for test case C.

Unit	Power(MW)	Cost($)
1	111.350975	934.278441
2	113.000646	961.688227
3	101.183869	1,263.79287
4	179.653785	2,143.382531
5	88.748757	722.245746
6	139.838162	1,595.964151
7	259.070572	2,612.100278
8	283.384682	2,778.150071
9	284.45058	2,798.012528
10	203.433921	3,608.966312
11	169.630792	2,978.605141
12	96.759801	1,969.618297
13	305.480028	5,134.852133
14	303.767971	5,147.507701
15	392.294478	6,428.195382
16	306.205148	5,211.597378
17	491.469276	5,343.858246
18	489.208354	5,288.728751
19	512.657763	5,570.800164
20	509.974661	5,540.03484
21	524.266162	5,091.271833
22	525.414979	5,114.52581
23	532.626636	5,243.911085
24	525.047777	5,092.822089
25	522.866776	5,275.295573
26	528.35423	5,378.909508
27	10.247098	1,146.237657
28	11.710313	1,181.331381
29	11.875318	1,185.419513
30	88.762214	722.468979
31	189.635109	1,642.52283
32	161.965586	1,327.358785
33	189.49853	1,641.953942
34	166.970272	1,623.248355
35	198.070091	2,024.866572
36	168.632795	1,605.096452
37	92.831048	1,020.540419
38	89.770777	970.193241
39	106.543766	1,195.934737
40	513.346297	5,585.715229
MW & $:	**10,500**	**122,102.003178**

the best value to be compared with respect to that reported in (Sun *et al.* 2009). The demand of power is 1,263 MW and the global solution for this system is the one reported by (Sun *et al.* 2009) with a total cost of $15,442.3758. In Table 5, a comparison of results with respect to two PSO methods is presented: PSO-Gaing (Gaing 2003) and QPSO-DM (Sun *et al.* 2009). Column 1 names the methods evaluated. Column 2 shows the minimum cost (in $) obtained. Column 3 shows the average cost over the total number of runs performed (100 for QPSO-DM and fast-CPSO, and 50 for PSO-Gaing). Column 4 indicates the maximum cost (in $) obtained by the methods, and, at the end, the time spent by the algorithms in a single run (in seconds). All the algorithms reached the total power demanded (1,263 MW) with feasible solutions (all the constraints are satisfied). Note that "N/A" stands for "Not Available".

Table 5. Comparison of the **best** values obtained by the proposed fast-CPSO and other approaches for test case D.

Method	Min. Cost($)	Average Cost($)	Max. Cost($)	Exec. Time
PSO-Gaing (Gaing 2003)	15,450.00	15,454.00	15,492.00	14.89[a]
QPSO-DM (Sun *et al.* 2009)	15,442.37	15,445.59	15,450.42	N/A
fast-CPSO	**15,432.40**	15,467.46	15,474.05	**0.17**[b]

[a] Simulation time using a Pentium III processor running at 550MHz with 256MB RAM.

[b] Simulation time using an AMD Athlon processor running at 1.24GHz with 512MB of RAM.

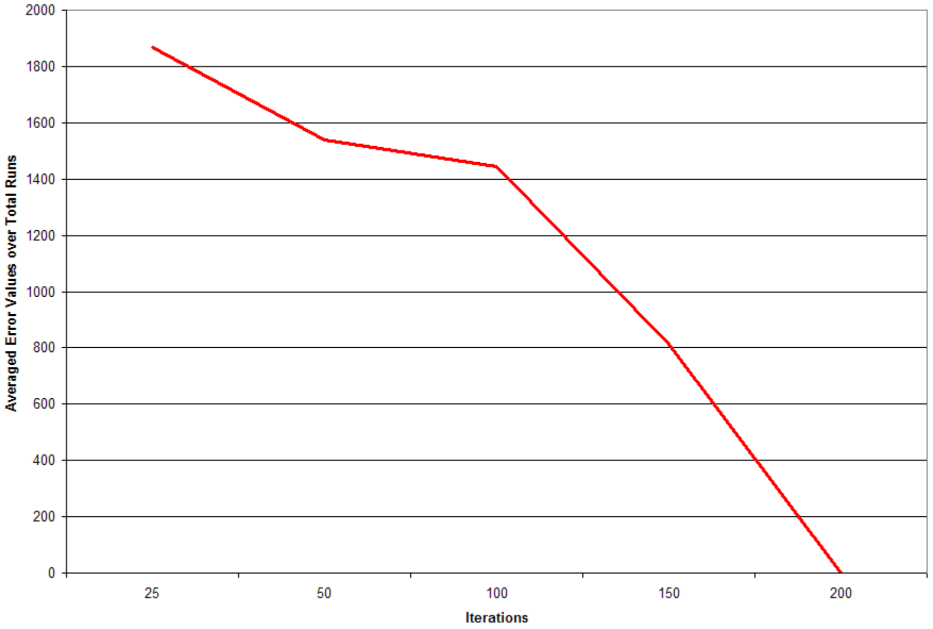

Figure 7. Evolution curve of fast-CPSO for test case D.

As can be observed in Table 5, the minimum cost is reached by fast-CPSO and, although the computational times required to obtain the best solutions are not comparable (between PSO-Gaing and fast-CPSO), the proposed approach is fast (it requires an average of 0.17 seconds per run).

Figure 7 shows the evolution curve for case D. In this case, the average error is relatively high until iteration 100, and then decays progressively during the last iterations.

In Table 6, the power generated by each unit can be observed, as well as the total power output (in MW), the power lost (in MW) and the generation cost (in $) for case D.

Table 6. Best results obtained by the fast-CPSO for test case D.

Unit	Power(MW)
1	453.173875
2	181.657861
3	251.289981
4	129.456347
5	167.002450
6	91.947002
Power output (MW):	1,274.52
Power lost (MW):	11.52
Generation cost ($):	15,432.40

Table 7. Comparison of the **best** values obtained by the proposed fast-CPSO and other approaches for test case E.

Method	Min. Cost($)	Average Cost($)	Max. Cost($)	Exec. Time
PSO-Gaing (Gaing 2003)	32,858.00	33,039.00	33,331.00	26.59[a]
QPSO-DM (Sun *et al.* 2009)	**32,652.57**	32,695.95	32,767.58	N/A
fast-CPSO	33,076	33,102.63	33,140.59	**1.52**[b]

[a] Simulation time using a Pentium III processor running at 550MHz with 256MB RAM.

[b] Simulation time using an AMD Athlon processor running at 1.24GHz with 512MB of RAM.

5.6. *Test Case E: 15-generator system with a smooth function*

The population size and the number of cycles were set in this case according to the total number of evaluations spent by (Sun *et al.* 2009) (i.e., 20,000), in order to allow a fair comparison of results. The size of the swarm was set to 100 particles and the maximum number of iterations was set to 200. 100 independent runs were performed to obtain the best value to be compared with respect to that reported in (Sun *et al.* 2009). The demand of power is 2,630 MW and the global solution for this system is the one reported by (Sun *et al.* 2009) with a total cost of $32,652.57. Table 7 shows a comparison of results with respect to two PSO approaches: PSO-Gaing (Gaing 2003) and QPSO-DM (Sun *et al.* 2009). Column 1 names the methods evaluated. Column 2 shows the minimum cost (in $) obtained. Column 3 shows the average cost over the total number of runs performed (100 for QPSO-DM and fast-CPSO, and 50 for PSO-Gaing). Column 4 indicates the maximum cost (in $) obtained by the methods, and, at the end, the time spent by the algorithms in a single run (in seconds). All the algorithms reached the total power demanded (2,630 MW). QPSO-DM and fast-CPSO obtained feasible solutions (all the constraints are satisfied) but the solution reached by PSO-Gaing reported in (Gaing 2003) does not satisfy the ramp rate limits constraint (in units 2 and 5). Note that “N/A” stands for “Not Available”.

As can be observed in Table 7, the minimum cost is reached by QPSO-DM. The solution obtained by fast-CPSO is better than the one obtained by PSO-Gaing, but the latter performed only 50 runs. The computational times required to obtain the best solutions are not comparable (between PSO-Gaing and fast-CPSO), but the proposed approach is fast (it required an average of only 1.52 seconds per run).

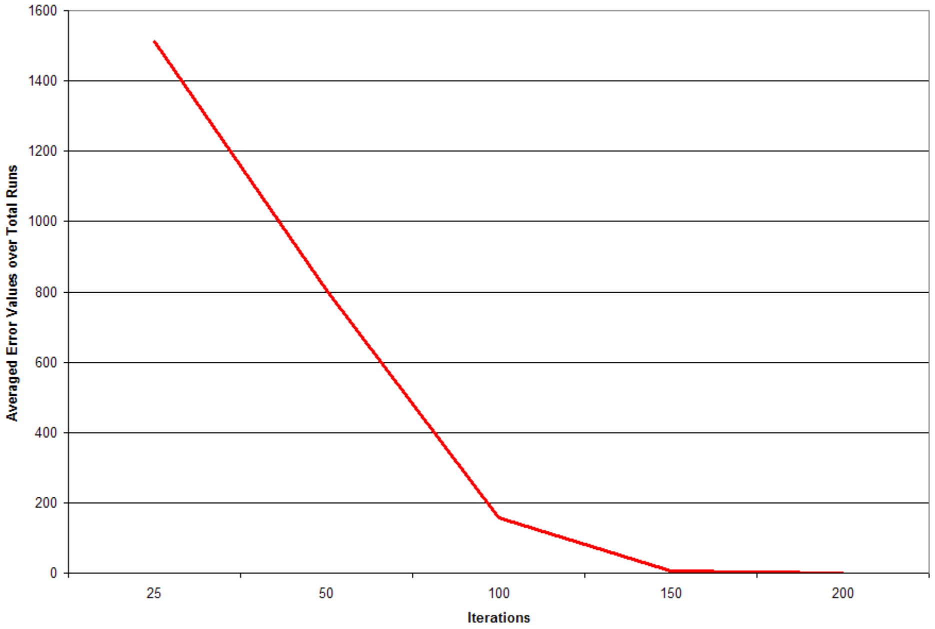

Figure 8. Evolution curve of fast-CPSO for test case E.

Figure 8 shows the evolution curve for case E. In this case, the average error decays quickly until iteration 100, then it decays more slowly lower until iteration 150 and then it quickly reaches zero.

In Table 8 the power generated by each unit can be observed, as well as the total power output (in MW), the power lost (in MW) and the generation cost (in $) for case E.

6. Conclusions and Future Work

A PSO-based approach for solving Economic Dispatch problems, with smooth and non-smooth cost functions, was presented. The approach introduces relatively simple changes to a traditional PSO algorithm, aiming to provide better diversity maintenance and a better exploration in constrained search spaces. The proposed approach was validated using five cases of the ED problem having 3, 6, 15 and 40 generating units. The performance of the proposed PSO was compared with respect to the one reported by other approaches that have solved the same problems, including a variety of evolutionary algorithms as well as mathematical programming techniques. The results were found to be very encouraging, and the execution time spent by the approach was found to be very low when compared to another approach that was implemented in a similar hardware platform.

As part of the future work to be developed, it could be interesting to design mechanisms that can improve the robustness of the proposed approach (i.e., to reduce its variability of results over several independent runs), particularly when dealing with difficult problems. In addition, it would be interesting to test the approach with a dynamic version of the ED problem.

Table 8. Best results obtained by the fast-CPSO for test case E.

Unit	Power(MW)
1	427.514201
2	345.130698
3	106.351539
4	108.941275
5	151.266812
6	455.590142
7	420.770185
8	148.432523
9	110.197987
10	142.444791
11	64.636988
12	70.180234
13	53.298569
14	25.820784
15	36.560917
Power output (MW):	2,667.13
Power lost (MW):	37.13
Generation cost ($):	33,076.12

Acknowledgments

The first and second authors gracefully acknowledge the continuous support from ANPCyT and the Universidad Nacional de San Luis. The third author acknowledges support from CONACYT project No. 103570.

References

Bakirtzis, A., Petridis, V., and Kazarlis, S., 1994. Genetic algorithm solution to the economic Dispatch problem. *Proc. Inst. Elect. Eng., Gener. Transm., Distrib.*, 141 (4), 377–382.

Blackwell, T. and Branke, J., 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10 (4), 459–472.

Cagnina, L.C., Esquivel, S.C., and Coello Coello, C.A., 2006. A Particle Swarm Optimizer for Constrained Numerical Optimization. *In*: T.P. Runarsson, H.G. Beyer, E. Burke, J.J. Merelo-Guervós, L.D. Whitley and X. Yao, eds. *Parallel Problem Solving from Nature (PPSN IX). 9th International Conference*, September. Lecture Notes in Computer Science Vol. 4193 Reykjavik, Iceland: Springer, 910–919.

Chen, P.H. and Chang, H.C., 1995. Large-scale economic dispatch by genetic algorithm. *IEEE Trans. Power Syst.*, 10, 1919–1926.

Clerc, M. and Kennedy, J., 2002. The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6 (1), 58–73.

Coello Coello, C.A., 2002. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191 (11-12), 1245–1287.

Daneshyari, G.G.Y.M., 2006. Diversity-based Information Exchange among Multiple Swarms in Particle Swarm Optimization. *In*: *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, July. Vancouver, BC, Canada: IEEE, 1686–1693.

Eberhart, R. and Kennedy, J., 1995. A new optimizer using particle swarm theory. *In*: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS'95*, October. Nagoya, Japan: IEEE Press, 39–43.

Fan, J.Y. and Zhang, L., 1998. Real-time economic Dispatch with line flow and emission constraints using quadratic programming. *IEEE Trans. Power Syst.*, 13 (2), 320–325.

Gaing, Z., 2003. Particle Swarm Optimization to Solving the Economic Dispatch Considering the Generator Constraints. *IEEE Transactions on Power Systems*, 18 (3), 1187–1195.

Hindi, K.S. and Ghani, M.R.A., 1991. Dynamic economic dispatch for large scale power systems: a Lagrangian relaxation approach. *Journal of Electrical Power & Energy Systems*, 13, 51–56.

Kennedy, J. and Eberhart, R., 2001. *Swarm Intelligence*. Morgan Kaufmann Publishers.

Kennedy, J., 2003. Bare Bones Particle Swarms. *In*: *Proceedings of the IEEE 2003 Swarm Intelligence Symposium (SIS 2003)*, April. Indianapolis, Indiana, USA: IEEE Press, 80–87.

Khamsawang, S. and Jiriwibhakorn, S., 2009. Solving the Economic Dispatch Problem by Using Differential Evolution. *International Journal of Electrical Power and Energy Systems Engineering*, 2 (2), 121–125.

Lee, F.N. and Breipohl, A.M., 1993. Reserve constrained economic dispatch with prohibited operating zones. *IEEE Transactions on Power Systems*, 8, 246–254.

Li, F., Morgan, R., and Williams, D., 1997. Hybrid genetic approaches to ramping rate constrained Dynamic economic dispatch. *Electric Power Systems Research*, 43, 97–103.

Liang, J.J. and Suganthan, P.N., 2005. Dynamic multi-swarm particle swarm optimizer. *In*: *Proceedings of the IEEE 2005 Swarm Intelligence Symposium (SIS 2005)*, June. IEEE Press, 124–129.

Lin, C.E. and Viviani, G.L., 1984. Hierarchical economic Dispatch for piecewise quadratic cost functions. *IEEE Trans. Power App. Syst.*, PAS-103 (6), 1170–1175.

Lin, W.M., Cheng, F.S., and Tsay, M.T., 2002. An improved Tabu search for economic Dispatch with multiple minima. *IEEE Trans. Power Syst.*, 17, 108–112.

Liu, D. and Cai, Y., 2005. Taguchi method for solving the Economic dispatch problem with nonsmooth cost functions. *IEEE Trans. Power Syst.*, 20, 2006–2014.

Naka, S., *et al.*, 2001. Practical distribution state estimation using hybrid particle swarm optimization. *Proc. IEEE Power Engineering Society Winter Meeting*, 2, 815–820.

Ongsakul, W. and Ruangpayoongsak, N., 2001. Constrained dynamic economic dispatch by simulated annealing/genetic algorithms. *In*: *IEEE Power Engineering International Conference on Power Industry Computer Applications (PICA 2001)*, May. IEEE Press, 207–212.

Park, J.B., *et al.*, 2005. A particle swarm optimization for Economic dispatch with nonsmooth cost functions. *IEEE Trans. Power Syst.*, 20, 34–42.

Park, J.H., *et al.*, 1993. Economic load dispatch for piecewise quadratic cost function using Hopfield neural network. *IEEE Trans. Power Syst.*, 8, 1030–1038.

Park, Y.M., Won, J.R., and Park, J.B., 1998. A new approach to economic load dispatch based on improved evolutionar programming. *Eng. Intell. Syst. Elect. Eng. Commu.*, 6, 103–110.

Sinha, N., Chakrabarti, R., and Chattopadhyay, P.K., 2003. Evolutionary Programming Techniques for Economic Load Dispatch. *IEEE Transactions on Evolutionary Computation*, 7 (1), 83–94.

Song, Y.H. and Yu, I.K., 1997. Dynamic load dispatch with voltage security and environmental constraints. *Electric Power Systems Research*, 43, 53–60.

Sriyanyong, P., Song, Y.H., and Turner, P.J., 2007. Vol. 49 of *Studies in Computational Intelligence*, Particle Swarm Optimisation for Operational Planning: Unit Commitment and Economic Dispatch. *In*: *Evolutionary Scheduling.* Springer.

Subramani, S.S. and Rajeswari, P.R., 2008. A Modified Particle Swarm Optimization for Economic Dispatch Problems with Non-Smooth Cost Functions. *International Journal of Soft Computing*, 3 (4), 326–332 ISSN: 1816-9503.

Sun, J., *et al.*, 2009. Solving the economic dispatch problem with a modified quantum-behaved particle swarm optimization method. *Energy Conversion and Management*, 50, 2967–2975.

Swarup, K.S. and Yamashiro, S., 2002. Unit Commitment Solution Methodology Using Genetic Algorithm. *IEEE Transactions on Power Systems*, 17 (1), 87–91.

Trojanowski, K., 2008. Multi-Swarm That Learns. *In*: *Intelligent Information Systems 2008*, July. Vancouver, BC, Canada: IEEE, 121–130.

Victoire, T.A.A. and Jeyakumar, A.E., 2005. Reserve Constrained Dynamic Dispatch of Units with valve-point effects. *IEEE Trans. Power Syst.*, 20, 1273–1282.

Walters, D.C. and Sheble, G.B., 1993. Genetic algorithm solution of economic dispatch with valve point loading. *IEEE Trans. Power Syst.*, 8, 1325–1332.

Wood, A.J. and Wollenberg, B.F., 1984. *Power Generation, Operation and Control.* 2 Ed. New York: John Wiley.

Yang, H.T., Yang, P.C., and Huang, C.L., 1996. Evolutionary Programming based Economic dispatch for units with non-smooth fuel cost functions. *IEEE Trans. Power Syst.*, 11, 112–118.

Yoshida, H., *et al.*, 2000. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 15, 1232–1239.

Zhao, S.Z., *et al.*, 2008. Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization. *In*: *2008 IEEE Congress on Evolutionary Computation (CEC'2008)*, June. Hong Kong: IEEE Press, 3845–3852.

Appendix A. Data for Economic Dispatch Problem

In Table A1 the value for 3 units, minimum and maximum power, and coefficients of cost (a, b and c values), correspond to test case A (smooth cost function). The data in Table A2 and Table A3 are the same values that in the previous case, but other values for the coefficients of the non-smooth function are added (e and f), and correspond to test cases B and C, respectively.

Tables A4 and A5 show the value for the units, minimum and maximum power, coefficients of cost (a, b and c values), the previous output power for each unit (P_i^0), the up-ramp and down-ramp limits (UR_i and DR_i), and the prohibited zones corresponding to test cases D and E. The B loss coefficients matrix, B0 vector and B00 value for cases D and E are provided in Tables A6, A7, A8 and A9.

Table A1. Data for **test case A** - 3 generating units with smooth function (Wood and Wollenberg 1984).

Unit	Pmin(MW)	Pmax(MW)	*a*	*b*	*c*
1	150	600	0.001562	7.92	561
2	100	400	0.001940	7.85	310
3	50	200	0.004820	7.97	78

Table A2. Data for **test case B** - 3 generating units with non-smooth function (Walters and Sheble 1993).

Unit	Pmin(MW)	Pmax(MW)	*a*	*b*	*c*	*e*	*f*
1	100	600	0.001562	7.92	561	300	0.0315
2	100	400	0.001940	7.85	310	200	0.042
3	50	200	0.004820	7.97	78	150	0.063

Table A3. Data for **test case C** - 40 generating units with non-smooth function (Sinha *et al.* 2003).

Unit	Pmin(MW)	Pmax(MW)	a	b	c	e	f
1	36	114	0.00690	6.73	94.705	100	0.084
2	36	114	0.00690	6.73	94.705	100	0.084
3	60	120	0.02028	7.07	309.54	100	0.084
4	80	190	0.00942	8.18	369.03	150	0.063
5	47	97	0.0114	5.35	148.89	120	0.077
6	68	140	0.01142	8.05	222.33	100	0.084
7	110	300	0.00357	8.03	287.71	200	0.042
8	135	300	0.00492	6.99	391.98	200	0.042
9	135	300	0.00573	6.60	455.76	200	0.042
10	130	300	0.00605	12.9	722.82	200	0.042
11	94	375	0.00515	12.9	635.20	200	0.042
12	94	375	0.00569	12.8	654.69	200	0.042
13	125	500	0.00421	12.5	913.40	300	0.035
14	125	500	0.00752	8.84	1760.40	300	0.035
15	125	500	0.00708	9.15	1728.30	300	0.035
16	125	500	0.00708	9.15	1728.30	300	0.035
17	220	500	0.00313	7.97	647.85	300	0.035
18	220	500	0.00313	7.95	649.69	300	0.035
19	242	550	0.00313	7.97	647.83	300	0.035
20	242	550	0.00313	7.97	647.81	300	0.035
21	254	550	0.00298	6.63	785.96	300	0.035
22	254	550	0.00298	6.63	785.96	300	0.035
23	254	550	0.00284	6.66	794.53	300	0.035
24	254	550	0.00284	6.66	794.53	300	0.035
25	254	550	0.00277	7.10	801.32	300	0.035
26	254	550	0.00277	7.10	801.32	300	0.035
27	10	150	0.52124	3.33	1055.10	120	0.077
28	10	150	0.52124	3.33	1055.10	120	0.077
29	10	150	0.52124	3.33	1055.10	120	0.077
30	47	97	0.01140	5.35	148.89	120	0.077
31	60	190	0.00160	6.43	222.92	150	0.063
32	60	190	0.00160	6.43	222.92	150	0.063
33	60	190	0.00160	6.43	222.92	150	0.063
34	90	200	0.0001	8.95	107.87	200	0.042
35	90	200	0.0001	8.62	116.58	200	0.042
36	90	200	0.0001	8.62	116.58	200	0.042
37	25	110	0.0161	5.88	307.45	80	0.098
38	25	110	0.0161	5.88	307.45	80	0.098
39	25	110	0.0161	5.88	307.45	80	0.098
40	242	550	0.00313	7.97	647.83	300	0.035

Table A4. Data for **test case D** - 6 thermal units with 26 buses and 46 transmission lines (Yoshida *et al.* 2000).

Unit	Pmin(MW)	Pmax(MW)	a	b	c	P_i^0	UR_i	DR_i	Prohibited Zones
1	100	500	0.0070	7.0	240	440	80	120	[210,240][350,380]
2	50	200	0.0095	10.0	200	170	50	90	[90,110][140,160]
3	80	300	0.0090	8.5	220	200	65	100	[150,170][210,240]
4	50	150	0.0090	11.0	200	150	50	90	[80,90][110,120]
5	50	200	0.0080	10.5	220	190	50	90	[90,110][140,150]
6	50	120	0.0075	12.0	190	110	50	90	[75,85][100,105]

Table A5. Data for **test case E** - 15 thermal units system (Lee and Breipohl 1993).

Unit	Pmin	Pmax	a	b	c	P_i^0	UR_i	DR_i	Prohibited zones
1	150	455	0.000299	10.1	671	400	80	120	-
2	150	455	0.000183	10.2	574	300	80	120	[185,225][305,335][420,450]
3	20	130	0.001126	8.8	374	105	130	130	-
4	20	130	0.001126	8.8	374	100	130	130	-
5	150	470	0.000205	10.4	461	90	80	120	[180,200][305,335][390,420]
6	135	460	0.000301	10.1	630	400	80	120	[230,255][365,395][430,455]
7	135	465	0.000364	9.8	548	350	80	120	-
8	60	300	0.000338	11.2	227	95	65	100	-
9	25	162	0.000807	11.2	173	105	60	100	-
10	25	160	0.001203	10.7	175	110	60	100	-
11	20	80	0.003586	10.2	186	60	80	80	-
12	20	80	0.005513	9.9	230	40	80	80	[30,40][55,65]
13	25	85	0.000371	13.1	225	30	80	80	-
14	15	55	0.001929	12.1	309	20	55	55	-
15	15	55	0.004447	12.4	323	20	55	55	-

Table A6. The B loss coefficients matrix (Gaing 2003) for **test case D**.

0.0017	0.0012	0.0007	-0.0001	-0.0005	-0.0002
0.0012	0.0014	0.0009	0.0001	-0.0006	-0.0001
0.0007	0.0009	0.0031	0	-0.0010	-0.0006
-0.0001	0.0001	0	0.0024	-0.0006	-0.0008
-0.0005	-0.0006	-0.0010	-0.0006	0.0129	-0.0002
-0.0002	-0.0001	-0.0006	-0.0008	-0.0002	0.0150

Table A7. The B0 loss coefficients vector and B00 value (Gaing 2003) for **test case D**.

B0:	-0.0003908	-0.0001297	0.0007047	0.0000591	0.0002161	-0.0006635
B00:	0.0056					

Table A8. The B loss coefficients matrix (Gaing 2003) for **test case E**.

0.0014	0.0012	0.0007	-0.0001	-0.0003	-0.0001	-0.0001	-0.0001	-0.0003	-0.0005	-0.0003	-0.0002	0.0004	0.0003	-0.0001
0.0012	0.0015	0.0013	0	-0.0005	-0.0002	0	0.0001	-0.0002	-0.0004	-0.0004	0	0.0004	0.0010	-0.0002
0.0007	0.0013	0.0076	-0.0001	-0.0013	-0.0009	-0.0001	0	-0.0008	-0.0012	-0.0017	0	-0.0026	0.0111	-0.0028
-0.0001	0	-0.0001	0.0034	-0.0007	-0.0004	0.0011	0.0050	0.0029	0.0032	-0.0011	0	0.0001	0.0001	-0.0026
-0.0003	-0.0005	-0.0013	-0.0007	0.0090	0.0014	-0.0003	-0.0012	-0.0010	-0.0013	0.0007	-0.0002	-0.0002	-0.0024	-0.0003
-0.0001	-0.0002	-0.0009	-0.0004	0.0014	0.0016	0	-0.0006	-0.0005	-0.0008	0.0011	-0.0001	-0.0002	-0.0017	0.0003
-0.0001	0	-0.0001	0.0011	-0.0003	0	0.0015	0.0017	0.0015	0.0009	-0.0005	0.0007	0	-0.0002	-0.0008
-0.0001	0.0001	0	0.0050	-0.0012	-0.0006	0.0017	0.0168	0.0082	0.0079	-0.0023	-0.0036	0.0001	0.0005	-0.0078
-0.0003	-0.0002	-0.0008	0.0029	-0.0010	-0.0005	0.0015	0.0082	0.0129	0.0116	-0.0021	-0.0025	0.0007	-0.0012	-0.0072
-0.0005	-0.0004	-0.0012	0.0032	-0.0013	-0.0008	0.0009	0.0079	0.0116	0.0200	-0.0027	-0.0034	0.0009	-0.0011	-0.0088
-0.0003	-0.0004	-0.0017	-0.0011	0.0007	0.0011	-0.0005	-0.0023	-0.0021	-0.0027	0.0140	0.0001	0.0004	-0.0038	0.0168
-0.0002	0	0	0	-0.0002	-0.0001	0.0007	-0.0036	-0.0025	-0.0034	0.0001	0.0054	-0.0001	-0.0004	0.0028
0.0004	0.0004	-0.0026	0.0001	-0.0002	-0.0002	0	0.0001	0.0007	0.0009	0.0004	-0.0001	0.0103	-0.0101	0.0028
0.0003	0.0010	0.0111	0.0001	-0.0024	-0.0017	-0.0002	0.0005	-0.0012	-0.0011	-0.0038	-0.0004	-0.0101	0.0578	-0.0094
-0.0001	-0.0002	-0.0028	-0.0026	-0.0003	0.0003	-0.0008	-0.0078	-0.0072	-0.0088	0.0168	0.0028	0.0028	-0.0094	0.1283

Table A9. The B0 loss coefficients vector and B00 value (Gaing 2003) for **test case E**.

B0:	-0.0001	-0.0002	0.0028	-0.0001	0.0001	-0.0003	-0.0002	-0.0002	0.0006	0.0039	-0.0017	0	-0.0032	0.0067	-0.0064
B00:	0.0055														