Applied Soft Computing 11 (2011) 2805–2817

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Non-smooth economic dispatch computation by fuzzy and self adaptive particle swarm optimization

Taher Niknam [a,*], Hasan Doagou Mojarrad [b], Hamed Zeinoddini Meymand [a]

[a] Department of Electrical and Electronics Engineering, Shiraz University of Technology, Shiraz, P.O. 71555-313, Iran
[b] Electronic and Electrical Engineering Department, Shiraz University of Technology, Shiraz, Iran

ARTICLE INFO

Article history:
Received 9 January 2010
Received in revised form
10 September 2010
Accepted 16 November 2010
Available online 24 November 2010

Keywords:
Economic dispatch
New adaptive particle swarm optimization (NAPSO)
Mutation operator
Multi-fuel effects
Self-adaptive parameter control

ABSTRACT

Economic dispatch (ED) problem is a nonlinear and non-smooth optimization problem when valve-point effects, multi-fuel effects and prohibited operating zones (POZs) have been considered. This paper presents an efficient evolutionary method for a constrained ED problem using the new adaptive particle swarm optimization (NAPSO) algorithm. The original PSO has difficulties in premature convergence, performance and the diversity loss in optimization process as well as appropriate tuning of its parameters. In the proposed algorithm, to improve the global searching capability and prevent the convergence to local minima, a new mutation is integrated with adaptive particle swarm optimization (APSO). In APSO, the inertia weight is tuned by using fuzzy IF/THEN rules and the cognitive and the social parameters are self-adaptively adjusted. The proposed NAPSO algorithm is validated on test systems consisting of 6, 10, 15, 40 and 80 generators with the objective functions possessing prohibited zones, multi-fuel effects and valve-point loading effects. The research results reveal the effectiveness and applicability of the proposed algorithm to the practical ED problem.

1. Introduction

As power demand increases and since the fuel cost of the power generation is exorbitant, reducing the operation costs of power systems becomes an important topic. The economic dispatch (ED) optimization problem is one of the fundamental issues in power systems to obtain optimal benefits and to control the committed generators output such that the total fuel cost is minimized while satisfying the power demand and all other operating constraints [1].

There are different methods to find the rate of optimum production for each power generation unit. Previous solutions for ED problems have been applied through various mathematical programming methods and optimization techniques, such as the lambda-iteration method [2], the base point and participation factors method [3], the gradient method [4], and Newton method [2]. However, the methods include an essential assumption: the incremental costs of the generators are monotonically increasing and piecewise-linear functions are used [2]. Unfortunately, this assumption may lead to infeasible practical use due to nonlinear characteristics in real generators. A dynamic programming (DP) method had been used for solving the ED problem with valve-point loading effect [5–7]. The DP technique decomposes a multi-stage decision problem into a sequence of single stage decision problems. However, the DP method may cause the dimensions of the ED problem to become extremely large, thus requiring large computational efforts.

With the development of computer science and technology, evolutionary algorithms have been successfully used to solve the ED problem, such as the genetic algorithm [8,9], particle swarm optimization [10,11], simulated annealing and tabu search [12].

When compared with conventional techniques, modern heuristic optimization techniques have been paid much more attention by many researchers due to their ability to find an almost global optimal solution for ED problems with operating constraints.

One of the evolutionary algorithms that have shown great potential and good perspective for the solution of various optimization problems [13–16] is PSO. The PSO algorithm, first introduced by Kennedy and Eberhart in 1995, was developed through simulation of a simplified social system, such as social behavior of birds flocking or fish schooling [17]. PSO has been found to be robust in solving continuous nonlinear optimization problems. Recently, PSO has been successfully employed to solve the ED problem while considering generator constraints [18] and non-smooth cost constraints [19]. However, the performance of the original PSO greatly depends on its parameters, such as inertia weight, cognitive and the social parameters, and it often suffers from the problem of being trapped in local optima. Eiben et al. [20] describes two ways

* Corresponding author. Tel.: +98 711 7264121; fax: +98 711 7353502.
E-mail addresses: niknam@sutech.ac.ir, taher_nik@yahoo.com (T. Niknam), hasan_doagou@yahoo.com (H.D. Mojarrad), h.zeinaddini@gmail.com (H.Z. Meymand).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.11.010

of defining the parameter values: adaptive parameter control and self-adaptive parameter control. In the former, the parameter values change according to a heuristic rule that takes feedback from the current search state, while in the latter, the parameters of the meta-heuristic are incorporated into the representation of the solution. Thus, the parameter values evolve together with the solutions of the population.

In this paper, an adaptive parameter control is used for inertia weight by using a fuzzy logical controller and the cognitive and the social parameters are self-adaptively evaluated. Also, in order to avoid trapping in local optima, this paper presents a new mutation operator to explore the search space much more efficiently.

This paper considers three types of non-convex ED problems, namely, ED with prohibited operating zones (EDPO), ED with combined valve-point loading effects and prohibited operating zones (EDVLPO) and ED with combined multi-fuel effects and valve-point loading effects (EDMFVL). The prohibited operating zones in the input–output performance curve for a typical thermal unit can be due to robustness in the shaft bearings caused by the operation of steam valves or to faults in machines themselves or in the associated auxiliary equipment such as boilers, feed pumps [21,22]. In practice, when adjusting the generation output of a unit, one must avoid operating in prohibited zones. The presence of prohibited zones for individual generators leads to a disjoint solution space. In addition, power plants usually have multiple valves that are used to control the output power of the unit. When steam admission valves in thermal units are first opened, a sudden increase in power loss is observed. This leads to ripples in the cost function, which is known as the valve point loading.

Moreover, many generating units, specifically those which are supplied with multi-fuel source lead to the problem of determining the most economic fuel to burn [23]. Consequently, multi-fuel units, prohibited operating zones and valve loading effects should be considered to solve a realistic ED problem, which makes it difficult to find the optimum solution.

In order to evaluate the performance of the proposed algorithm, two case studies including 6 and 15 generating units are used for EDPO, two typical systems consisting of 40 and 80 thermal units are used for EDVLPO and EDMFVL with 10 generators, where both multiple fuels and valve-point loading effect are considered. The results obtained through the improved NAPSO algorithm are analyzed and compared with those reported in the recent literature.

The main contribution of the paper is to present a new evolutionary algorithm to solve non-smooth and non-convex economic dispatch problem.

The rest of the paper is structured as follows: Section 2 presents the mathematical formulation of ED problem as a constrained optimization problem. Section 3 contains a brief overview of original PSO along with proposed NAPSO algorithm. Section 4 describes NAPSO algorithm for solving ED problem with the proposed constraint handling mechanism. Section 5 reports the different experimental results compared to the other methodologies. Finally, Section 6 concludes this paper.

2. Mathematical model of the ED problem

The ED problem is a nonlinear optimization problem. The objective function and the constraints are mathematically modeled as follows.

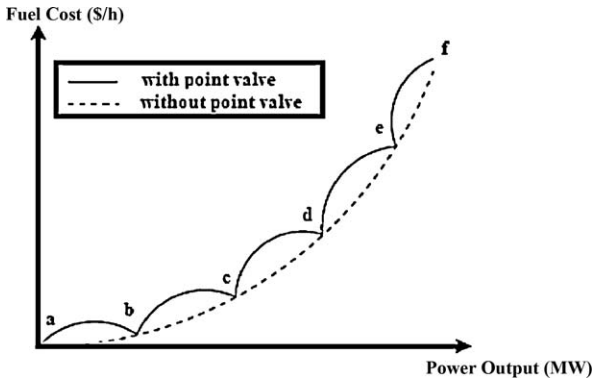

Fig. 1. Cost function of a generator with and without valve-points.

2.1. Objective function of ED problem without the valve effect

The objective function corresponding to the production cost can be approximated to be a quadratic function of the active power outputs of the generating units. The cost function curve of a thermal generator is shown in Fig. 1. The dash line represents the curve approximated by the quadratic function given as follows [24]:

$$\text{Min} \quad C(X) = \sum_{j=1}^{N_g} F_j(P_j) \tag{1}$$

$$F_j(P_j) = a_j P_j^2 + b_j P_j + c_j$$
$$X = [P_1, P_2, \ldots, P_{N_g}] \tag{2}$$

where $C(X)$ and $F_j(P_j)$ are total generation cost and Fuel cost function of generator j, respectively; a_j, b_j, c_j are fuel cost coefficients of generator j, P_j is electrical output power of generator j, N_g is the number of generators, and, X is the control variable vector.

2.2. Objective function of ED problem with the valve effect

Considering the practical operating conditions of each unit, the economic dispatch problem is a non-convex optimization problem. The non-smooth cost function considering multiple valve effects is shown in Fig. 1. The function has a quadratic input–output curve superposed with a nonlinear sinusoid function, which models valve-point characteristics [25–27]:

$$F_j(P_j) = a_j P_j^2 + b_j P_j + c_j + |e_j \sin(f_j(P_{j,\min} - P_j))| \tag{3}$$

where e_j and f_j are non-smooth fuel cost coefficients of generator j and $P_{j,\min}$ is minimum power generation limit of generator j.

2.3. Objective function of ED with multiple fuels and valve effect

Usually there are many generating units supplied with multiple fuels. The cost function of these units, unlike the conventional units cost function, should be presented with a few piecewise functions reflecting the effects of fuel type changes. The fuel cost function for *jth* unit reflecting the multiple fuel option and valve-point loading is as follows [18]:

$$F_j(P_j) = \begin{cases} a_{j,1}P_{j,1}^2 + b_{j,1}P_{j,1} + c_{j,1} + |e_{j,1}\ \sin(f_{j,1}(P_{j,\min} - P_j))| & \text{if} \quad P_{j,\min} \le P_j \le P_{j,1}; \quad \text{Fuel Type '1'} \\ a_{j,2}P_{j,2}^2 + b_{j,2}P_{j,2} + c_{j,2} + |e_{j,1}\ \sin(f_{j,1}(P_{j,\min} - P_j))| & \text{if} \quad P_{j,1} \le P_j \le P_{j,2}; \quad \text{Fuel Type '2'} \\ \ldots & \\ a_{j,n}P_{j,n}^2 + b_{j,n}P_{j,n} + c_{j,n} + |e_{j,n}\ \sin(f_{j,n}(P_{j,\min} - P_j))| & \text{if} \quad P_{j,n-1} \le P_j \le P_{j,\max}; \quad \text{Fuel Type '}n\text{'} \end{cases} \tag{4}$$

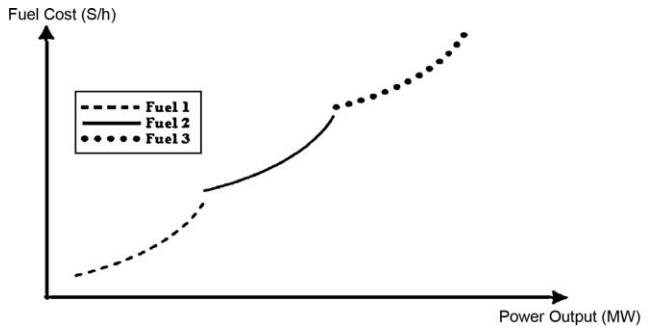

Fig. 2. Cost function of a thermal generating unit supplied with multiple fuels.

where, $a_{j,n}, b_{j,n}, c_{j,n}, e_{j,n}, f_{j,n}$ are fuel cost coefficients of the generator '*j*' for the fuel type *n* and $P_{j,\max}$ is maximum power generation limit of the generator '*j*'. The piece quadratic cost function considering multi-fuel is shown in Fig. 2.

2.4. Constraints

2.4.1. Power balance constraint

The power balance constraint is expressed as:

$$\sum_{j=1}^{N_g} P_j - P_L - P_D = 0 \tag{5}$$

where P_D is the total load of consumers and P_L is the total transmission network losses, which is expressed as a quadratic function of the generators power outputs. P_L can be approximated as:

$$P_L = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P_i B_{ij} P_j + \sum_{i=1}^{N_g} B_{0i} P_i + B_{00} \tag{6}$$

where B_{ij} is the *ij*th element of the loss coefficient square matrix, B_{i0} is the *i*th element of the loss coefficient vector, and B_{00} is the loss coefficient constant.

2.4.2. Power output constraints

The electrical output power of each unit must be between its minimum and maximum values.

$$P_{j,\min} \le P_j \le P_{j,\max}. \tag{7}$$

2.4.3. Prohibited operating zone constraints

Each generator has its generation capacity limitation, which cannot be exceeded. Moreover, a typical thermal unit may have a steam valve in operation, or a vibration in a shaft bearing, which may result in interference and causes a discontinuity in input–output performance curve, called prohibited zones, as shown in Fig. 3. Therefore, in practical operation, adjusting the generation output of a unit must be done regarding all capacity limits so that it prevents the unit operation in prohibited zones. The feasible operating zones of a unit can be described as follows [22]:

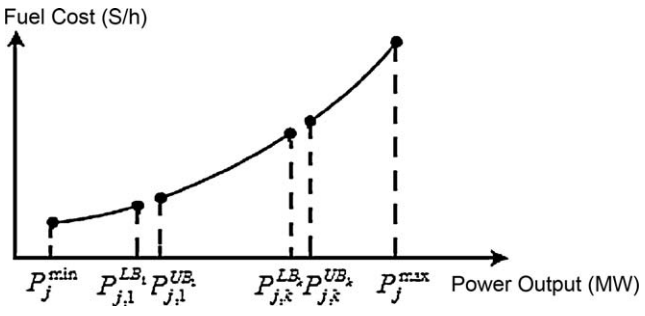

Fig. 3. Cost function of a generator with prohibited operating zones.

$$P_j \in \begin{cases} P_j^{\min} \le P_j \le P_j^{LB_1} \\ P_j^{UB_{K-1}} \le P_j \le P_j^{LB_K} \quad j = 1, 2, \ldots, N_g \\ P_j^{UB_K} \le P_j \le P_j^{\max} \end{cases} \tag{8}$$

where $P_j^{LB_K}$ and $P_j^{UB_K}$ are lower and upper bounds of the *k*th prohibited zone of unit *j*, respectively; *k* is the index of prohibited zones.

3. NAPSO

3.1. Original PSO

Particle swarm optimization (PSO) is one of the evolutionary optimization algorithms developed by Eberhart and Kennedy. This algorithm was inspired by the movement of a flock of birds searching for food. Compared to other evolutionary techniques, the major advantages of PSO are: simple concept, easy implementation, robustness to control parameters, minimal storage requirement, and pointing global and local minimum due to higher exploration abilities.

PSO is initialized with a random population of solutions – particles – in an N-dimensional space. The *j*th particle is represented by $X_j = [x_{j1}, x_{j2}, \ldots x_{jN_g}]$, and then the particles are replaced to get close to the best position and also find the global minimum point. Each particle adjusts its position according to its own experience, and the experience of neighboring particles, making use of the best position encountered by itself and its neighbors. The swarm direction of a particle is defined by the set of particles neighboring the particle and its history experience. Its position and velocity are updated according to the following formula:

$$V_j^{t+1} = \omega * V_j^t + c_1 * rand(.) * (P_{best,j} - X_j^t) + c_2 * rand(.) * (G_{best} - X_j^t) \tag{9}$$

$$X_{swarm,j}^{t+1} = X_{swarm,j}^t + V_j^{t+1} \quad j = 1, 2, 3, \ldots, N_{swarm} \tag{10}$$

The first term on the right side of Eq. (9) is the momentum part of the particle. The inertia weight ω represents the degree of the momentum of the particles. The velocity of the *j*th particle is represented as $V_j^t = [v_{j,1}^t, v_{j,2}^t, \ldots, v_{j,N_g}^t]$. The second part is the 'cognition' part, which represents the independent thinking of the particle itself. The best position of the *j*th particle so far is stored and represented as $P_{best,j} = [P_{best,j1}, P_{best,j2}, \ldots, P_{best,j,N_g}]$. The third part is the 'social' part, which represents the society behavior of the population. All the P_{best} are evaluated by using a fitness function, which changes for different problems. The best particle among all P_{best} is represented as G_{best}.

Acceleration parameters c_1 and c_2 are two positive constants, called the cognitive and social parameters, respectively. *rand*(.) represents a function that creates random numbers between 0 and 1; two independent random numbers are included in Eq. (9), which is used to maintain the diversity of the population. *j* is the index of each particle and *t* is the current iteration number.

3.2. Adaptive PSO

There are three tuning parameters ω, c_1 and c_2 that greatly influence the PSO algorithm performance. The inertia weight ω is used to control the impact of the previous history of velocities on the current velocity. Suitable selection of the inertia weight ω can provide a balance between global and local optimum points [28].

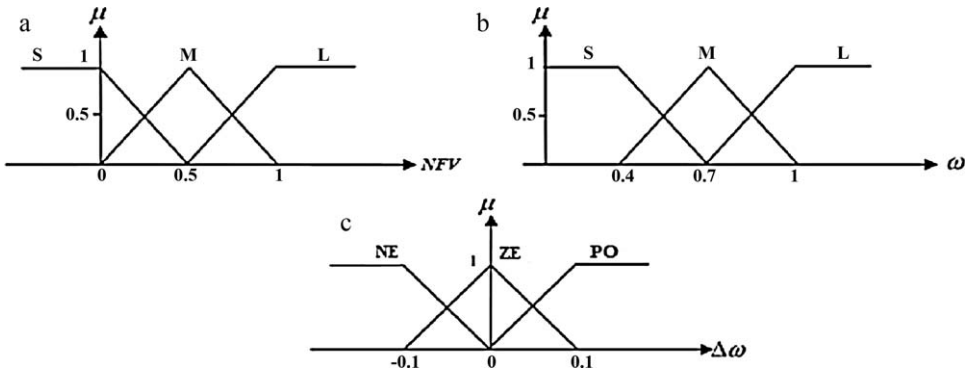

Fig. 4. Membership functions of inputs and outputs (a) NFV, (b) ω, (c) $\Delta\omega$.

The acceleration factors c_1 and c_2 determines the influence of partial best $P_{best,j}$ and global best G_{best}.

Since c_1 expresses how much the particle trusts its own past experience, it is called cognitive parameter. On the other hand, c_2 expresses how much it trusts the swarm, and is called social parameter.

Also, on each dimension, particle velocities are restricted to minimum and maximum velocities, which are user-defined parameters,

$$V_{j,\min} \le V_j^{t+1} \le V_{j,\max} \tag{11}$$

To control excessive roaming of particles outside the search space, usually, $V_{j,\min}$ is taken as $-V_{j,\max}$.

If $V_{j,\max}$ is too high, particles may fly past good solutions. If $V_{j,\max}$ is too small, particles may not explore sufficiently beyond local solutions. In many experiences with PSO, $V_{j,\max}$ was often set at 10–20% of the dynamic range on each dimension.

The performance of this type of evolutionary algorithms is heavily dependent on the setting of control parameters. Proper selection of the control parameters is very important for the success of the algorithm. Eiben et al. [20] describe two ways of modifying the parameter values: adaptive parameter control and self-adaptive parameter control. In adaptive parameter control, the parameter values change according to a heuristic rule that takes feedback from the current search state. The information used for the current state is usually the current iteration of the search, the performance of operators and/or the diversity of the population. For example, adaptive parameter control is used for inertia control in [29,30] by linearly decreasing inertia over the generations, and in [31] by using a fuzzy logical controller. In self-adaptive parameter control, the parameters of the meta-heuristic are incorporated into the representation of the solution. Thus, the parameter values evolve together with the solutions of the population.

In this paper, in addition to using the fuzzy-adaptive parameter control for the inertia proposed by Eberhart and Kennedy [29], we investigate the other parameters of the evolution variables, in order to self-adaptively control them.

3.2.1. Fuzzy adaptive parameter control

The inertia weight (ω) determines the influence of particle's previous velocity over the velocity in the next iteration. Suitable selection of the inertia weight provides a balance between global exploration, local exploration, and exploitation, which results in less iteration on average to find a sufficiently optimal solution. Several approaches have been applied to handle the inertia weight during the progression of the optimization process. Constant inertia weight, linearly decreasing inertia weight and random inertia weight are a few examples. ω often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight ω is set according to the following equation:

$$\omega = \omega_{\max} - \left(\frac{\omega_{\max} - \omega_{\min}}{\mathrm{Iter}_{\max}}\right) * \mathrm{Iter} \tag{12}$$

where $\omega_{\max}$ is the initial weight, $\omega_{\min}$ is the final weight, $\mathrm{Iter}_{\max}$ is the maximum iteration number, and Iter is the current iteration number [31–34].

In this paper, the inertia weight of PSO is dynamically adjusted using fuzzy IF/THEN rules.

The fuzzy system consists of four principal components: fuzzification, fuzzy rules, fuzzy reasoning and defuzzification, which are described in the following subsections.

3.2.1.1. Fuzzification. Among a set of membership functions, left-triangle, triangle and right-triangle membership functions are used for every input and output variables as illustrated in Fig. 4. All the memberships of input variables are represented in three linguistic values: S, M and L which stand for Small, Medium and Large, respectively in Fig. 4a and b and Table 1. Output variable is represented in three fuzzy sets of linguistic values; NE (negative), ZE (zero), and PO (positive) with associated membership functions, as shown in Fig. 4c [35].

3.2.1.2. Fuzzy rules. The Mamdani-type fuzzy rules are used to formulate the conditional statements that comprise fuzzy rule base. For example:

if (NFV is S) and (ω is M) THEN ($\Delta\omega$ is NE)

Normalized fitness value (NFV) is an input variable between 0 and 1. $\Delta\omega$ changes between −0.1 and 0.1.

The fuzzy rules in Table 1 are used to choose the inertia weight correction ($\Delta\omega$). Each rule represents a mapping from the input space to the output space.

3.2.1.3. Fuzzy reasoning. The fuzzy inference strategy is used to map the inputs to the outputs. Mamdani's fuzzy inference method

Table 1
Fuzzy rules for inertia weight correction.

	IF		THEN
Rule number	*NFV*	ω	$\Delta\omega$
1	S	S	ZE
2	S	M	NE
3	S	L	NE
4	M	S	PE
5	M	M	ZE
6	M	L	NE
7	L	S	PE
8	L	M	ZE
9	L	L	NE

is used in this paper. The AND operator is typically used to combine the membership values for each fired rule to generate the membership values for the fuzzy sets of output variables in the consequent part of the rule. Since there may be several rules fired in the rule sets, for some fuzzy sets of the output variables there may be different membership values obtained from different fired rules.

To obtain a better inertia weight under the fuzzy environment, the variables selected as inputs to the fuzzy system are the current best performance evaluation and current inertia weight, whereas the output variable is the change in the inertia weight. The normalized fitness value NFV is used as an input variable between 0 and 1, and is defined as:

$$\mathrm{NFV} = \frac{(\mathrm{FV} - \mathrm{FV}_{\min})}{(\mathrm{FV}_{\max} - \mathrm{FV}_{\min})} \tag{13}$$

The calculated value of FV may be used from Eq. (13) at the first iteration as $\mathrm{FV}_{\min}$ for the next iterations. In Eq. (13) $\mathrm{FV}_{\max}$ is a very large value, which is greater than any acceptable feasible solution. Typical inertia weight value is $0.4 \le \omega \le 0.9$. Both positive and negative corrections limits are required for the inertia weight. Therefore, a range of −0.1 to 0.1 has been chosen for the inertia weight correction.

$$\omega^{t+1} = \omega^t + \Delta\omega \tag{14}$$

ω^t is the value of ω at the tth iteration and is initially taken 0.9. To obtain a deterministic control action, a defuzzification strategy is required. It will be illustrated at a later point.

3.2.1.4. Defuzzification. The fuzzy inference system produces a fuzzy output and in situation of working in real world, it is necessary to have a crisp output. Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representative value. The crisp value represents a fuzzy set the best. In general, there are different methods for defuzzifying a fuzzy set. In this paper, we use centroid defuzzification strategy. When the fuzzy output set A with membership function $a(y)$ is generated by the fuzzy inference system, the output crisp value is calculated as follows:

$$\bar{y} = \frac{\int y a(y) dy}{\int a(y) dy} \tag{15}$$

The idea behind centroid defuzzification is to consider the output membership function as a linear mass distribution and then calculate the center gravity of the mass as a representation point for mass distribution.

3.2.2. Self-adaptive parameter control

In this section, a self-adaptive manipulation of c_1, c_2 and $V_{\max}$ is considered to avoid the cumbersome task of first localizing and then fine-tuning these three parameters.

Suitable fine tuning of cognitive and social parameters c_1 and c_2 in Eq. (9) may result in faster convergence of the algorithm and alleviate the risk of settling in one of the local minima. An extensive study of these acceleration parameters is given for standard PSO by Kennedy [36]. As default values, $c_1 = c_2 = 2$ were proposed, but next experimental results indicated that alternative configurations, depending on the application, may produce superior performance. Recent work reports that it might be even better to choose a larger cognitive parameter, c_1, than a social parameter, c_2, with constraint $c_1 + c_2 \le 4$ [37]. Arumugam and Rao [30] suggest that the acceleration coefficients should neither set to a constant value nor set as a linearly decreasing time varying function. Instead, they are defined as a function of local best and global best values of the fitness function of a minimization problem.

In this paper, the acceleration coefficients and the clamping velocity are neither set to a constant value, like in original PSO, nor set as a time varying function, like in adaptive PSO variants. Instead, they are incorporated into their own optimization problem, as explained below.

Each particle will be allowed to self-adaptively set its own parameters by using the same process used by PSO given by Eqs. (9) and (10). To this end, these three parameters are considered as three new variables that are incorporated with position vectors X_j. In general, if N_g is the dimension of the problem and p is the number of self adapting parameters, the new position vector for particle j will be:

$$X_j^{New} = [x_{j,1}, x_{j,2}, \ldots, x_{j,N_g}, x_{j,N_g+1}, \ldots, x_{j,N_g+p}] \tag{16}$$

It is clear that the first N_g variables correspond to the real position vector of the particle in the search space, while the last p accounts for its own acceleration constants and velocity limit.

Obviously, these new variables do not enter the fitness function but are manipulated using the same mixed individual-social learning paradigm as the one used in PSO. Also, V_j and $P_{best,j}$, which represent the velocity and the best position so far for particle j, respectively, increase their dimension as follows:

$$V_j^{New} = [v_{j,1}, v_{j,2}, \ldots, v_{j,N_g+1}, \ldots, v_{j,N_g+p}] \tag{17}$$

$$P_{best,j}^{New} = [P_{best,1}, P_{best,2}, \ldots P_{best,N_g}, P_{best,N_g+1}, \ldots, P_{best,N_g+p}] \tag{18}$$

This way, by using Eqs. (9) and (10), each particle will be additionally endowed with the ability of adjusting its parameters by aiming at both the parameters it had when it got its best position in the past and the parameters of the leader, which managed to bring this best particle to its privileged position. As a consequence, particles use their cognition of individual thinking and social cooperation not only to improve their positions, but also to improve the way they do it by accommodating themselves to the best known conditions, namely, their conditions when taking the best so far position and the leader's conditions.

3.2.3. A new mutation for NAPSO

In order to improve the convergence velocity and accuracy of the PSO algorithm, this paper proposes a new mutation operator in standard particle swarm optimization method. Premature convergence may occur under different situations:

(a) The population has converged to a local optimum of the objective function.
(b) The population has lost its diversity.
(c) The search algorithm has proceeded slowly or has not proceeded at all.

Mutation is a powerful strategy to diversify the PSO population and improve the PSO's performance in preventing premature convergence to local minima [28]. In each step, the algorithm mutates vectors by selecting four vectors m_1, m_2, m_3, m_4 from initial population as $m_1 \neq m_2 \neq m_3 \neq m_4 \neq j$ in order to cover the entire searching region uniformly.

For each target vector $X_j^t (j = 1, 2, \ldots, N_{swarm})$, a mutant vector $X_{mut,j}^t$ is generated as:

$$\begin{aligned} X_{mut,j}^t &= X_{m_1}^t + rand(.) * (X_{m_2}^t - X_{m_3}^t) + rand(.) * (G_{best} - X_{m4}^t) \\ X_{mut,j}^t &= [x_{mut,1}^t, x_{mut,2}^t, \ldots, x_{mut,N_g}^t] \end{aligned} \tag{19}$$

The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector $X_{new,j}^t = [x_{new,1}^t, x_{new,2}^t, \ldots, x_{new,N_g}^t]$, that is

$$x_{new,z}^t = \begin{cases} x_{mut,z}^t & \text{if } (rand1(.) \le rand2(.)) \text{ or } z = randperm(.) \\ x_{swarm,z}^t & \text{otherwise} \end{cases} \tag{20}$$

where, *randperm*(.) is a randomly chosen index from 1,2,. . .,N_g, which ensures that $X^t_{\text{new},j}$ gets at least one parameter from $X^t_{\text{mut},j}$. *rand*1(.) and *rand*2(.) are two random numbers between 0 and 1. To decide whether or not it should become a member of the next generation, the trial vector $X^t_{new,j}$ is compared to the target vector $X^t_{swarm,j}$ using the cost function.

$$X^{t+1}_{swarm,j} = \begin{cases} X^t_{new,j} & \text{if} \quad C(X^t_{new,j}) \le C(X^t_{swarm,j}) \\ X^t_{swarm,j} & \text{otherwise} \end{cases} \tag{21}$$

If the generation cost of the trial vector $C(X^t_{new,j})$ is better than that of the target vector i.e., $C(X^t_{swarm,j})$, the target vector is replaced with the trial vector in the next generation.

4. Implementation of the NAPSO algorithm for ED problem

In this section, the process to solve ED problems with constraints using NAPSO methods was developed to get efficiently a high quality solution in practical power system operation. The NAPSO methods were mainly used to find the optimal power generation of each unit that was submitted to operate at the specific period, thus minimizing the total generation cost.

To apply the proposed NAPSO algorithm in solving ED problem, the following steps should be repeated:

Step 1: The input data including cost coefficients of the generators, output generator constraints, and transmission loss matrix coefficients, limits of output power and self-adaptive parameters, and loads should be read.
Step 2: An initial population, X_j, and initial velocity V_j, which must meet constraints, are generated randomly as follows:

$$\text{population} = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_{swarm} \end{bmatrix} \tag{22}$$

$$X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,N_g}, x_{i,N_g+1}, \dots, x_{i,N_g+p}];$$

$$x_{i,j} = rand(.) * (x_{j,\max} - x_{j,\min}) + x_{j,\min};$$

$$j = 1, 2, 3, \dots, (N_g + p); \quad i = 1, 2, 3, \dots, N_{swarm}; \quad p = 3$$

$$\text{velocity} = \begin{bmatrix} V_1 \\ V_2 \\ \dots \\ V_{N_{swarm}} \end{bmatrix} \tag{23}$$

$$V_i = [v_{i,1}, v_{i,2}, \dots, v_{i,N_g}, v_{i,N_g+1}, \dots, v_{i,N_g+p}];$$

$$v_{i,j} = 0.1 * (rand(.) * (v_{j,\max} - v_{j,\min}) + v_{j,\min});$$

$$j = 1, 2, 3, \dots, (N_g + p); i = 1, 2, 3, \dots, N_{swarm}; \quad p = 3$$

where, $x_{j,\max}$ and $x_{j,\min}$ are the maximum and minimum values of the jth control variable, respectively. $v_{j,\max}$ and $v_{j,\min}$ are the maximum and minimum velocity values of the jth control variable, respectively.

Step 3: Handling the generating capacity constraint

The individuals of the algorithm are created randomly. Therefore, it is very important to satisfy the real power balance equality constraints (5). We have used a dedicated method for handling the constraints. The constraints handling procedure is shown in Fig. 5 [38].

The algorithm ensures that the generated random elements are not selected repeatedly.

Step 4: Check and modify the constraints (7) to ensure the feasibility of all potential solutions.
Step 5: Conflicts handling in constraints:

If ever, the maximum or minimum limits of the generation of a unit, as given by Eq. (8), lie in the prohibited zone for the generator, then some modifications are to be made in the upper and lower limits for the generator constraints in order to avoid the conflicts. To overcome the unit's operating in prohibited operating zones, a delimitation point, P^d_j, is introduced for the kth prohibited zone of unit j.

The delimitation point divides the prohibited zone into two subzones, i.e., the left and right prohibited sub-zones with respect to the point. When a unit operates into one of its prohibited zones, the concept of this strategy is to force the unit either to move toward the lower bound of that zone from the left sub-zone or to move toward the upper bound of that zone from the right sub-zone.

To attain the purpose of moving the unit validly, a delimitation ratio, λ, is defined first as follows:

$$\lambda = \frac{(P^d_j - P^{LB_k}_j)}{(P^{UB_K}_j - P^{LB_K}_j)} \tag{24}$$

where $P^{UB_K}_j$ and $P^{LB_K}_j$ are the upper and lower bounds of the kth prohibited zone of unit j, respectively. If the ratio, λ is smaller than 0.5, the delimitation point is closer to the lower bound, $P^{LB_K}_j$. On the other hand, if the ratio is greater than 0.5, the point is closer to the upper bounds, $P^{UB_K}_j$.

Step 6: For each individual X_j ; j = 1, 2, ..., N_g of the population, the transmission loss P_L is calculated using Eq. (6).
Step 7: The objective function is evaluated for each individual (only the first N_g variables appear in the fitness function)
Step 8: Select $P_{best,j}$ and G_{best}

The real power output of all generators is represented in additional self-adaptive parameters as the position of the particle in the swarm. The best position of a particle is $P_{best,j}$, and the best position out of all the P_{best} is taken as G_{best}.

Step 9: Updated the NAPSO parameters

In this algorithm, the suitable selection of inertia weight ω is updated by the fuzzy rules described in Section 3.2.1.

The other parameters tuning by self-adaptive parameters that described in Section 3.2.2.

Step 10: Update velocity and position

To modify the position of each individual, it is necessary to calculate the velocity of each individual in the next stage which is obtained from (10). The position of each individual is modified by (10).

Step 11: Check the velocity component constraint according to the limits from the following conditions:

$$v^{t+1}_{i,j} = \begin{cases} v_{j,\max} & \text{if} \quad v^{t+1}_{i,j} \ge v_{j,\max} \\ v_{j,\min} & \text{if} \quad v^{t+1}_{i,j} \le v_{j,\min} \end{cases} \tag{25}$$

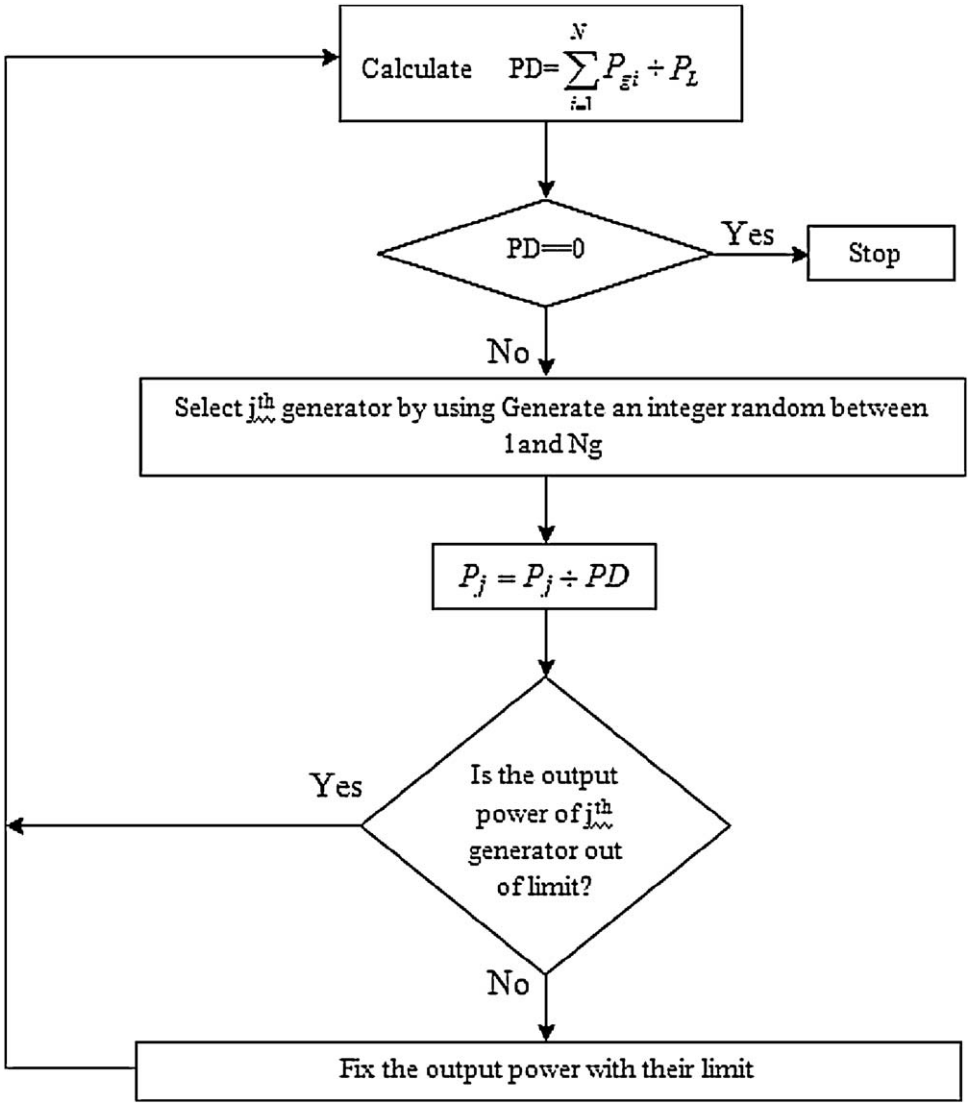

Fig. 5. Flowchart of handling the generating capacity constraints of the NAPSO algorithm.

where $v_{j,\min} = -v_{j,\max} = 0.5 * x_{j,\max}$.

Step 12: Position modification considering constraints

Since the resulting position of an individual is not always guaranteed to satisfy the equality and inequality constraints due to over/under velocity, the modified position of an individual is adjusted. Equality constraint is checked by flowchart as described in Step 3. If any element of an individual violates its inequality constraint then the position of the individual is fixed to its maximum/minimum operating point. Therefore, this can be formulated as:

$$x_{i,j}^{t+1} = \begin{cases} x_{i,j}^{t} & \text{if} \quad P_{j,\min} < x_{i,j}^{t} < P_{j,\max} \\ P_{j,\min} & \text{if} \quad x_{i,j}^{t} < P_{j,\min} \\ P_{j,\max} & \text{if} \quad x_{i,j}^{t} > P_{j,\max} \end{cases} \tag{26}$$

Step 13: Check and modify the constraints according to the method mentioned in Step 5.
Step 14: Mutation

Mutation is an operation that adds a differential vector to a population vector of individuals. The new individuals are generated by mutation process as described in Section 3.2.3. If any element of an individual violates its constraints (7) and (8) then the position of the individual is modified as:

$$X_{mut,j}^{t} = \begin{cases} P_{j,\min} & \text{if} \quad X_{mut,j}^{t} < P_{j,\min} \\ P_{j,\max} & \text{if} \quad X_{mut,j}^{t} > P_{j,\max} \end{cases} \tag{27}$$

Step 15: Update $P_{best,j}$ and G_{best}

The objective function of each particle is now evaluated according to the updated position and mutation. $P_{best,j}$ of each individual at iteration $k+1$ is updated as:

$$P_{best,j}^{t+1} = \begin{cases} X_{i,j}^{t+1} & \text{if} \quad C(P_{best,j}^{t+1}) > C(X_{swarm,j}^{t}) \\ X_{mut,j}^{t} & \text{if} \quad C(P_{best,j}^{t+1}) > C(X_{mut,j}^{t}) \\ P_{best,j}^{t} & \text{if} \quad C(P_{best,j}^{t+1}) < (C(X_{swarm,j}^{t}) \, and \, C(X_{mut,j}^{t})) \end{cases} \tag{28}$$

where C is the total generation cost. Also, G_{best}, at iteration $t+1$ is set as the best evaluated position among $P_{best,j}^{t+1}$.

Step 16: If the current iteration number reaches the predetermined maximum iteration number, the search process is stopped, otherwise go to step 9.

5. Simulation results

In this section, to assess the efficiency of the proposed NAPSO approaches, five case studies (6, 15, 40, 80 and 10 generators) of ED problems are performed in which the objective functions are valve-point loading, multi-fuel effects and prohibited operating zones in the power system operation. Transmission losses are employed to demonstrate were taken into account.

The proposed algorithm has been implemented in MATLAB7 and executed on Pentium-IV, 3 GHz personal computer with 2GB RAM to solve the ED problems.

The setup for the algorithm is as follows:

Number of population = $5 \times N_g$. Maximum iteration number are 100, 100, 500, 500 and 100 for case studies 1, 2, 3, 4 and 5, respectively.

In the following, the results are presented.

5.1. ED with prohibited operating zones (EDPO)

5.1.1. Case study 1

The system contains six thermal units. The system parameters including fuel cost coefficients and generator capacities are listed in [39]. The load demand is 1263 MW. The prohibited zones are given in Table 2.

The B matrix of the transmission loss coefficient (with a 100 MVA base capacity), is as follows [39]:

$$B_{ij} = \begin{bmatrix} 0.0017 & 0.0012 & 0.0007 & -0.0001 & -0.0005 & -0.0002 \\ 0.0012 & 0.0014 & 0.0009 & 0.0001 & -0.0006 & -0.0001 \\ 0.0007 & 0.0009 & 0.0031 & 0.0000 & -0.0010 & -0.0006 \\ -0.0001 & 0.0001 & 0.0000 & 0.0024 & -0.0006 & -0.0008 \\ -0.0005 & -0.0006 & -0.001 & -0.0006 & 0.0129 & -0.0002 \\ -0.0002 & -0.0001 & -0.0006 & -0.0008 & -0.0002 & 0.015 \end{bmatrix}$$

$$B_{i0} = 0.001 * \begin{bmatrix} -0.3908 & -0.1297 & 0.7047 & 0.0591 & 0.2161 & -0.6635 \end{bmatrix}$$

$$B_{00} = 0.056$$

The optimum dispatch of generators and fuel cost by the proposed method is tabulated in Table 3 for two states: one state is with considering power loss and other state is without considering power loss. The computational results of the best, average, and the worst fuel costs among the 50 runs of solutions which satisfy the system constraints are listed in Table 4.

Table 2
Prohibited zones of units for test case study 1.

Generator	Zone 1 (MW)	Zone 2 (MW)
1	[210,240]	[350,380]
2	[90,110]	[140,160]
3	[150,170]	[210,240]
4	[80,90]	[110,120]
5	[90,110]	[140,150]
6	[75,85]	[100,105]

Table 3
Dispatch result of the proposed algorithm for case study 1.

Generator	Output power (MW)	Output power (MW)
Pg1	446.4232	446.3697
Pg2	172.608	171.0092
Pg3	262.6183	263.8431
Pg4	142.7752	124.9542
Pg5	164.665	171.8235
Pg6	86.323	85
Ploss (MW)	12.4131	–
Pload (MW)	1263	1263
Cost ($/h)	15,443.7656645	15,275.9485

Table 4
Comparison of fuel costs for 6 generators for power demand of 1263 MW.

Evolution method	Best cost ($)	Worst cost ($)	Average cost ($)
NAPSO	15,443.765664	15,443.765683	15,443.765671
FAPSO	15,445.244	15,451.63	15,448.052
PSO	15,450	15,492	15,454
GA [27]	15,459	15,524	15,469
DE [27]	15,449.766	15,449.874	15,449.777
NPSO-LRS [40]	15,450	15,452	15,450.5
SPSO [41]	15,466.63	–	–
PC_PSO [41]	15,453.09	–	–
SOH-PSO [41]	15,446.02	–	–
TS [42]	15,454.89	15,498.05	15,472.56
MTS [42]	15,450.06	15,453.64	15,451.17
SA [42]	15,461.1	15,545.5	15,488.98

For comparison, the obtained results from several recently published ED solution methods are also represented in Table 4. It is noted that the mentioned results of all other methods reported in this paper, except the conventional PSO and FAPSO have been directly quoted from their respective references. For the conventional PSO and FAPSO, we have implemented its algorithm according to the explanations of Section 4 and tested it on the first test system. It can be concluded that the solution obtained from the proposed method is better than other techniques in the literature. Table 4 discloses that NAPSO has the higher probability of attaining quality solution.

The convergence characteristics of 6 units test system with NAPSO, FAPSO and PSO algorithms for economic dispatch are shown in Fig. 6. The convergence characteristics are drawn by plotting the minimum fitness value from the combined population across iteration index. From Fig. 6 it is observed that the cost function value converges smoothly to the optimum value without any abrupt oscillations, thus ensuring convergence reliability of the proposed NAPSO algorithm.

Fig. 7 displays the distribution of total fuel cost of the results obtained by the proposed algorithm, NAPSO and FAPSO for 50 different runs. As shown in Fig. 7, the NAPSO algorithm has the most stable and minimum deviation. Also, we can see that with the help of proposed methods the chance of getting stuck in local minima is reduced which result in a better performance.

The presented results in this section indicate performance, robustness of the proposed NAPSO to solve the realistic ED problems, which confirm the validity of the developed approach.

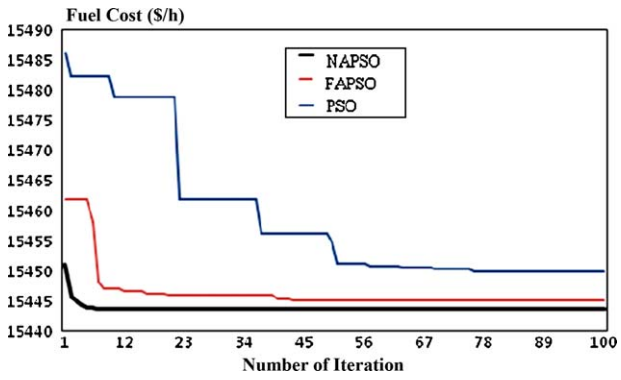

Fig. 6. Convergence behaviors of NAPSO, FAPSO and PSO algorithms for a load demand of 1263 MW.

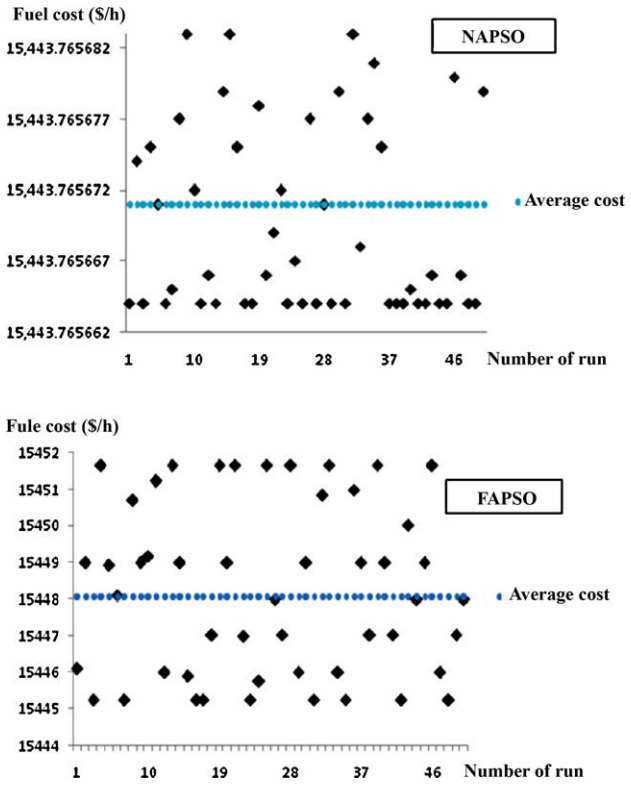

Fig. 7. Distribution of generation cost of case study 1 for NAPSO and FAPSO (50 trials).

Table 5
Prohibited zones of units for test case study 2.

Generator	Zone 1 (MW)	Zone 2 (MW)	Zone 3 (MW)
2	[185,225]	[305,335]	[420,450]
5	[180,200]	[305,335]	[390,420]
6	[230,255]	[365,395]	[430,455]
12	[30,40]	[55,65]	–

5.1.2. Case study 2

The load demand is 2630 MW and the system contains 15 thermal units whose characteristics and B matrix loss formula are given in [39], which four units have prohibited operation zones, which, Table 5 lists the prohibited zones of units 2, 5, 6, and 12. These zones result in three disjoint feasible sub-regions for each of units 2, 5, 6, and two for unit 12.

The results of the best, average and the worst fuel costs from this case study are listed in Table 6. For validity, these results are also compared with the FAPSO, PSO, GA, ESO, DE, SPSO, PC_PSO, SOH-PSO, MTS, SA, SCA, CSO and the APSO method. It can be evidently seen from Table 6 that the proposed technique provides better results than the other reported minimum costs using some of other methods. So, the proposed method is more robust and practically applicable to real systems.

Table 6
Comparison of fuel costs for 15 generators for power demand of 2630 MW.

Evolution method	Best cost ($)	Worst cost ($)	Average cost ($)
NAPSO	32,548.585876	32,548.5904	32,548.5869
FAPSO	32,659.794	32,676.07	32,663.19
PSO	32,858	33,031	32,989
GA [27]	33,063.54	33,337	33,228
ESO [27]	32,640.86	32,710	32,620
DE [27]	32,588.865	32,641.419	32,609.851
SPSO [41]	32,798.69	–	–
PC_PSO [41]	32,775.36	–	–
SOH-PSO [41]	32,751.39	32,945	32,878
MTS [43]	32,716.87	32,796.13	32,767.4
SA [43]	32,786.4	33,028.95	32,869.51
SCA [43]	32,867.025	33,381.0607	33,138.3020
APSO [44]	32,742.77	–	32,976.6812
CSO [43]	32,588.9189	32,796.7792	32,679.8775

Table 7
Optimum dispatch result of the proposed algorithm for power demand of 2630 MW.

Generator	Output power (MW)	Output power (MW)
Pg1	454.9999	455
Pg2	454.9999	455
Pg3	130	130
Pg4	130	130
Pg5	234.2005	271.5806
Pg6	460	460
Pg7	464.9999	465
Pg8	60	60
Pg9	25	25
Pg10	30.9939	25
Pg11	76.7014	43.4193
Pg12	79.9999	55
Pg13	25	25
Pg14	15	15
Pg15	15	15
Ploss(MW)	26.8959	–
Pload(MW)	2630	2630
Cost	32,548.585876	32,256.754239

Table 7 shows the best dispatch of generators, power loss and cost among all solutions, which is found by the proposed algorithm.

Fig. 8 shows the convergence characteristics of the NAPSO, FAPSO, and PSO for power demand of 2630 MW. The figure demonstrates that the proposed algorithm converges to the global optimal solution after 10 iterations while the FAPSO and PSO algorithms converge to local solutions after 55 and 60 iterations, respectively.

By studying the results of Fig. 8, it is clear that the new proposed algorithm is more efficient, faster and giving a cheaper total generating cost than the other algorithms. In other words, the new proposed algorithm is capable of giving a more optimum solution.

5.1.3. Investigation of the solution quality on the proposed method

Since the performance of original PSO depends on its parameters such as inertia weight and two acceleration coefficients (i.e., c_1 and c_2), it is important to determine the suitable values of parameters. To successfully implement the proposed algorithm, some parameters must be determined in advance based on a reasonable framework. In the proposed algorithm, an adaptive parameter control is used for inertia weight by using a fuzzy logical controller and the cognitive and the social parameters are self-adaptively evaluated.

5.1.3.1. Effect of self-adaptive control. Tables 8 and 9 show the best, average and worst solutions obtained by NAPSO for ω varying from 0.4 to 0.9 with step size 0.1. As seen in Tables 8 and 9, NAPSO has

Table 8
Effect of fuzzy rule on proposed algorithm.

Parameter	Total generation cost ($/h)		
ω	Best	Average	Worst
0.4	15,444.025702	15,446.765722	15,449.765784
0.5	15,444.065693	15,447.765719	15,450.765763
0.6	15,444.765714	15,445.765751	15,448.765806
0.7	15,444.165672	15,447.765738	15,450.765862
0.8	15,444.205670	15,446.765778	15,448.766230
0.9	15,444.165693	15,447.765875	15,449.766229

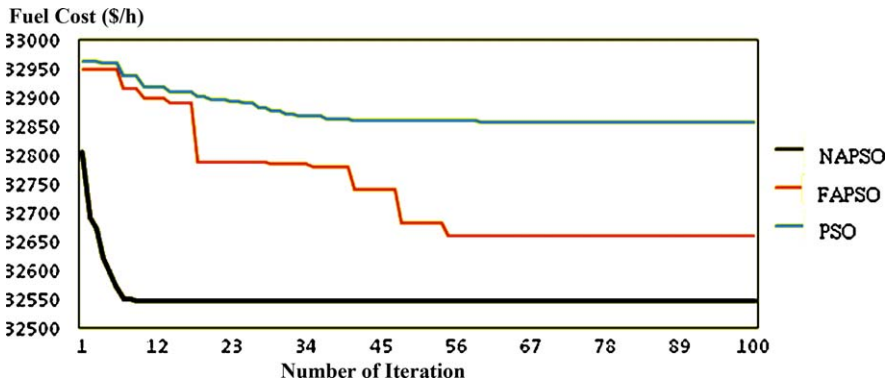

Fig. 8. Convergence behaviors of NAPSO, FAPSO and PSO algorithms for case study 2.

Table 9
Effect of fuzzy rule on proposed algorithm.

Parameter	Total generation cost ($/h)		
ω	Best	Average	Worst
0.4	32,548.585910	32,608.58704	32,621.594306
0.5	32,548.585990	32,591.59666	32,613.656186
0.6	32,548.585919	32,598.59107	32,607.616722
0.7	32,548.586168	32,630.60011	32,642.642348
0.8	32,548.585923	32,633.60208	32,651.665468
0.9	32,548.586350	32,612.62368	32,634.666441

provided almost the same solution for different combinations of parameter ω. According to the tables, NAPSO is robust for solving ED problems with non-smooth cost functions while both of parameters c_1 and c_2 are tuned self-adaptively.

5.1.3.2. Comparison of NAPSO with other NAPSO strategies. In order to show the traits of proposed algorithm, we consider NAPSO in 4 cases for 6 and 15 units.

Case 1: the proposed NAPSO
Case 2: NAPSO without considering proposed mutation
Case 3: NAPSO without considering fuzzy ω
Case 4: NAPSO without considering self-adaptive c_1 and c_2

To show and comprise cases of ascendency, Tables 10 and 11 have been provided. Due to the stochastic nature of the proposed algorithm, a series of 20 independent runs have been carried out. From the data presented in the tables, it can be inferred that the mutation used in NAPSO is very effective for reaching the optimum solution and it does not reach the best solution obtained by NAPSO algorithm. Also, by comparing the result obtained from NAPSO and the proposed approaches for $c_1 = 2$ and $c_2 = 2$, it is found that proposed algorithm is very dependent on proper tuning of these parameters.

5.2. ED with combined valve-point loading effects and prohibited operating zones (EDVLPO)

5.2.1. Case study 3

In order to demonstrate the efficiency and the scalability of the proposed approach, the proposed method is tested on 40 units system without transmission loss. This case study comprises 40 units, 5 of which exhibit prohibited zones. The test data for the system of 40 generating units have been obtained from [45] in which the fuel cost functions also take into account the valve-point loading. Total load demand of the system is 10,500 (MW) and all of generators should satisfy this load demand economically.

To demonstrate the applicability of the proposed NAPSO when encountering prohibited operating zones, this case study investigates two prohibited zones (POZ1 and POZ2). Generator constraints of prohibited zone from number 10 to 14 units are listed in Tables 12 and 13. These prohibited zones result in four disjoint feasible sub-regions for each of the units.

Table 10
Comparison of NAPSO on different cases.

Methods	Total generation cost ($/h)		
	Best	Average	Worst
NAPSO	15,443.765664	15,443.765671	15,443.765683
NAPSO without mutation	15,449.1392	15,451.0352	15,452.9164
NAPSO without Fuzzy ω[a]	15,444.765693	15,446.765719	15,449.765763
NAPSO without self-adaptive c_1, c_2 ($c_1 = c_2 = 2$)	15,449.082	15,450.327	15,452.783

[a] $\omega = \frac{(\omega_{\min} - \omega_{\max})}{\text{Iter}_{\max}} \times \text{Iter} + \omega_{\max}$.

Table 11
Comparison of NAPSO on different cases.

Methods	Total generation cost ($/h)		
	Best	Average	Worst
NAPSO	32,548.585876	32,548.5869	32,548.5904
NAPSO without mutation	32,601.173	32,632.502	32,649.861
NAPSO without Fuzzy ω[a]	32,588.585910	32,598.58704	32,628.594306
NAPSO without self-adaptive c_1, c_2 ($c_1 = c_2 = 2$)	32,601.704	32,623.819	32,656.827

[a] $\omega = \frac{(\omega_{\min} - \omega_{\max})}{\text{Iter}_{\max}} \times \text{Iter} + \omega_{\max}$.

Table 12
Prohibited zones of units for test case study 3.

Generator	Zone 1 (MW)	Zone 2 (MW)	Zone 3 (MW)
10	[130,150]	[200,230]	[270,299]
11	[100,140]	[230,280]	[300,350]
12	[100,140]	[230,280]	[300,350]
13	[150,200]	[250,300]	[400,450]
14	[200,250]	[300,350]	[450,490]

Table 13
Prohibited zones of units for test case study 3.

Generator	Zone 1 (MW)	Zone 2 (MW)	Zone 3 (MW)
10	[130,150]	[200,230]	[270,299]
11	[100,140]	[230,280]	[300,350]
12	[100,140]	[230,280]	[300,350]
13	[150,220]	[250,300]	[400,450]
14	[200,250]	[300,400]	[450,490]

Table 14
Comparison of fuel costs for 40 generators for power demand of 10,500 MW.

Methods	Total generation cost ($/h)		
	Best	Average	Worst
NAPSO (POZ1)	121,412.6102	121,412.7373	121,412.9109
FAPSO (POZ1)	121,719.73	121,783.093	121,842.51
PSO (POZ1)	123,861.45	124,162.4819	124,663.047
NAPSO (POZ2)	121,491.0662	121,491.2756	121,491.5261
FAPSO (POZ2)	122,261.3706	122,471.0751	122,597.5196
PSO (POZ2)	124,875.8523	125,162.7011	125,368.9204

Since this is a larger system with more nonlinear elements, it has more local minima. Therefore, it is more difficult to find the global solution.

We could not find a research work that considers the mentioned characteristics of the ED problem, which are present in the case study 3. Therefore, we have implemented the conventional PSO and FAPSO according to the algorithm of Section 4 and tested it on the third test system. The computational results of the best, average, and the worst fuel costs are listed in Table 14. The results presented in Table 14 shows that the worst result (max cost) obtained by the proposed algorithm is better than the best result (min cost) found by other method. Not only the best result but also the worst result of the proposed algorithm is less costly than the best result of other methods, which indicates the effectiveness of the proposed algorithm to solve the ED problems with valve-point effect and prohibited zones.

Table 15 shows the simulation results obtained by NAPSO for the best solution for POZ1, POZ2 and without prohibited zones.

Table 16
Comparison of fuel costs for 80 generators for power demand of 21,000 MW.

Methods	Pload	Best cost ($/h)
NAPSO (EDVLPOZ1)	21,000	242,844.1172
FAPSO (EDVLPOZ1)	21,000	244,273.5429
PSO (EDVLPOZ1)	21,000	249,248.3751

5.2.2. Case study 4

To demonstrate the applicability of the proposed NAPSO for large-scale test systems, simulations are preformed on systems, which have 80 generators whose fuel cost function is calculated by taking account the effect of valve-point loading. The large-scale system has been created by expanding the 40 generator EDVLPOZ1 case study 3. Table 16 shows the best results obtained among all solutions, which are found by the proposed algorithm, FAPSO and PSO for 20 different runs.

5.3. ED with combined multi-fuel and valve-point loading effects (EDMFVL)

5.3.1. Case study 5

In order to demonstrate the efficiency of the proposed approach on practical ED problem, the proposed method is tested on a 10-unit system where both multiple fuels and valve-point loading effect are considered. The input data is given in [46]. The total load demand is 2700 MW and not considering transmission line losses.

Table 17 shows the best dispatch of generators, load demand, fuel type, and cost among all the solutions, which are found by the proposed algorithm. For comparison, the computational results from several recently published ED solution methods are also represented in Table 18.

Table 15
Optimum dispatch result of the proposed algorithm for power demand of 10,500 MW.

Generator	Output power[a]	Output power[b]	Output power[c]	Generator	Output power[a]	Output power[b]	Output power[c]
Pg1	110.8018	110.7997	110.7998	Pg21	523.2793	523.2793	523.2793
Pg2	110.8000	110.8004	110.7998	Pg22	523.2806	523.2793	523. 2793
Pg3	97.3999	97.3971	97.3999	Pg23	523. 2793	523. 2793	523. 2793
Pg4	179.7331	179.7331	179.7331	Pg24	523. 2793	523. 2793	523. 2793
Pg5	87.7997	87.8009	91.4370	Pg25	523. 2793	523. 2793	523. 2793
Pg6	140	139.9999	139.9999	Pg26	523. 2793	523. 2793	523. 2793
Pg7	259.5996	259.5996	259.5996	Pg27	10	10	10
Pg8	284.5996	284.5961	284.5996	Pg28	10	10	10
Pg9	284.5996	284.5997	284.5996	Pg29	10	10	10
Pg10	130	130	130	Pg30	87.79989	87.7998	87.7999
Pg11	94	94	168.79998	Pg31	190	189.9999	190
Pg12	94	94	94	Pg32	190	189.9999	190
Pg13	214.7597	214.7597	125	Pg33	190	189.9999	189.9999
Pg14	394.2793	394.27937	400	Pg34	164.8014	164.8003	164.7998
Pg15	394.2793	394.27937	394.2793	Pg35	194.3927	194.4019	199.9999
Pg16	394.2793	394.27937	394.2793	Pg36	200	199.9999	199.9999
Pg17	489.2793	489.27937	489.2793	Pg37	110	109.9999	109.9999
Pg18	489.2793	489.27937	489.2793	Pg38	110	109.9999	109.9999
Pg19	511.2793	511.27937	511.2793	Pg39	110	109.9999	109.9999
Pg20	511.2793	511.27937	511.2793	Pg40	511.2793	511.2793	511.2793
Fuel cost ($/h)	[a]121,412.57		[b]121,412.6102		[c]121,491.0662		

[a] Without POZ.
[b] With POZ1.
[c] With POZ2.

Table 17
Dispatch result of the proposed algorithm for power demand of 2700 MW.

Generator	Output power (MW)	Fuel type
Pg1	219.066	2
Pg2	211.1629	1
Pg3	279.6578	1
Pg4	239.4176	3
Pg5	280.0966	1
Pg6	239.5266	3
Pg7	287.7378	1
Pg8	240.0895	3
Pg9	428.1732	3
Pg10	275.072	1
Load (MW)	2700	
Cost ($/h)	623.62170	

Table 18
Comparison of fuel costs for 10 generators for case study 5.

Evolution method	Best cost ($)	Worst cost ($)	Average cost ($)
NAPSO	623.62170	623.67543	623.6335
FAPSO	624.2189	624.2951	624.2782
IGA-MU [47]	624.5178	630.8705	627.6087
PSO-LRS [47]	624.2297	628.3214	625.7887
NPSO [47]	624.1624	627.4237	625.2180
NPSO-LRS [47]	624.1273	626.9981	624.9985
DSPSO–TSA [48]	623.8375	623.9001	623. 8625
PSO [48]	624.3045	625.9252	624.5054
GA [48]	624.5050	624.8169	624.7419
TSA [48]	624.3078	624.8285	635.0623
ARCGA [47]	623.8281	623.8550	623.8431

Simulation results reveal that the NAPSO algorithm can give an optimal generation cost more consistently than any other algorithms considered due to the extra diversification capability provided by the proposed mutation operator and appropriate tuning of internal parameters of NAPSO.

5.4. CPU time

Execution time complexity of each optimization method is very important for its application to real systems.

As for CPU time, it may not be directly comparable between the methods due to various computers and programming languages used.

In order to show the computational efficiency of the proposed algorithm, its average CPU time of the 50 trial runs for the five test systems are shown in Table 19 and compared with that of the conventional PSO and FAPSO. All tests of this paper are carried out on a common personal computer using the MATLAB7 software package. Table 19 shows less computation times of the proposed NAPSO than the other methods.

In other words, the new proposed algorithm is capable of giving a more optimum solution with less computational time and provides a general idea that the NAPSO can be utilized without any restriction on practical networks.

Table 19
CPU time of PSO, FAPSO and proposed algorithm for all of case studies.

Methods	Mean time (s)				
	Case study 1	Case study 2	Case study 3	Case study 4	Case study 5
NAPSO	3.2	4.8	12.7	23	2.8
FAPSO	8.7	9.6	19.6	37.22	5.9
PSO	11.3	15.4	35.87	68	10.7

5.5. Advantages of NAPSO over Other EAs

Having observed the comparison results given in Tables 4, 6, 14, 16 and 18, the NAPSO algorithm is shown to be more efficient than all other EAs in finding optimal solutions and can always converge to a better evaluation value.

In the proposed method, in order to alleviate the risk of settling in one of the local minima and to gain a faster convergence, an adaptive parameter control is used for inertia weight (ω) by using a fuzzy logical controller and the cognitive and the social parameters (c_1 and c_2) are self-adaptively evaluated.

The performance of NPSO is better than the other methods due to the extra diversification capability provided by the proposed mutation. Although the equality and inequality constraints are satisfied by constraint handling strategy (see Section 4), the proposed method reduces the computational burden and is superior to most of available economic dispatch techniques.

Hence, the NAPSO is suggested as a powerful optimization tool for non-convex ED problems

6. Conclusion

This paper proposes a new evolutionary algorithm based on the combination of Adaptive PSO and mutation operator, called new adaptive PSO (NAPSO), to solve ED problems with non-smooth/non-convex cost functions.

In APSO, acceleration factors are coevolved with the particles during optimization process and the inertia weight is tuned based on a fuzzy system. Five case studies have been employed to illustrate the applicability of the proposed method. Numerical results show that the proposed algorithm has advantages over other algorithms in superior robustness, less computational efforts, comparable performance with mathematical programming approach, and applicability to large-scale and practical systems.

References

[1] D. Liu, Y. Cai, Taguchi method for solving the economic dispatch problem with nonsmooth cost functions, IEEE Trans. Power Syst. 20 (4) (2005) 2006–2014.
[2] A.J. Wood, B.F. Wollenberg, Power Generation Operation and Control, John Wiley & Sons, New York, 1984.
[3] C.L. Chen, C.L. Wang, Branch-and-bound scheduling for thermal generating units, IEEE Trans. Energy Convers. 8 (2) (1993) 184–189.
[4] K.Y. Lee, et al., Fuel cost minimize at ion for both real- and reactive power dispatches, IEE Proc. Part. C, Gener. Transm. Distr. 131 (3) (1984) 85–93.
[5] Z.X. Liang, J.D. Glover, A zoom feature for a dynamic programming solution to economic dispatch including transmission losses, IEEE Trans. Power Syst. 7 (2) (1992) 544–550.
[6] A. Rong, H. Hakonen, R. Lahdelma, A variant of the dynamic programming algorithm for unit commitment of combined heat and power systems, Eur. J. Oper. Res. 190 (2008) 741–755.
[7] S.S. Kumar, V. Palanisamy, A dynamic programming based fast computation Hopfield neural network for unit commitment and economic dispatch, Electr. Power Syst. Res. 77 (2007) 917–925.
[8] D.C. Walters, G.B. Sheble, Genetic algorithm solution of economic dispatch with valve point loading, IEEE Trans. Power Syst. 8 (3) (1993) 1325–1328.
[9] P.H. Chen, H.C. Chang, Large-scale economic dispatch by genetic algorithm, IEEE Trans. Power Syst. 10 (4) (1995) 1919–1926.
[10] J.B. Park, K.S. Lee, J.R. Shin, K.Y. Lee, A particle swarm optimization for economic dispatch with nonsmooth cost function, IEEE Trans. Power Syst. 20 (1) (2005) 34–42.
[11] Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Trans. Power Syst. 18 (3) (2003) 1187–1195.
[12] W.M. Lin, F.S. Cheng, M.T. Tsay, Nonconvex economic dispatch by integrated artificial intelligence, IEEE Trans. Power Syst. 16 (2) (2001) 307–311.
[13] Y.X. Jin, H.Z. Cheng, J.Y. Yan, L. Zhang, New discrete method for particle swarm optimization and its application in transmission network expansion planning, Electr. Power Syst. Res. 77 (3–4) (2007) 227–233.
[14] C.J. Liao, C.T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (10) (2007) 3099–3111.
[15] J. Izquierdo, I. Montalvo, R. Pérez, V.S. Fuertes, Design optimization of wastewater collection networks by PSO, Comput. Math. Appl. 56 (3) (2008) 777–784.

[16] S. Janson, D. Merkle, M. Middendorf, Molecular docking with multi-objective particle swarm optimization, Appl. Soft Comput. 8 (1) (2008) 666–675.
[17] J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE Int. Conf. Neural Netw. 4 (1995) 1942–1948.
[18] Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Trans. Power Syst. 18 (3) (2003) 661–669.
[19] J.B. Park, K.S. Lee, J.R. Shin, K.Y. Lee, A particle swarm optimization for economic dispatch with non-smooth cost functions, IEEE Trans. Power Syst. 20 (3) (2005) 34–39.
[20] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141.
[21] P.H. Chen, H.C. Chang, Large-scale economic dispatch by genetic algorithm, IEEE Trans. Power Syst. 10 (1995) 1919–1926.
[22] S.O. Orero, M.R. Irving, Economic dispatch of generators with prohibited operating zones: A genetic algorithm approach, IEE Proc. Part C, Gener. Transm. Distr. 143 (6) (1996) 529–533.
[23] C.E. Lin, G.L. Viviani, Hierarchical economic dispatch for piecewise quadratic cost functions, IEEE Trans. Power Appl. Syst. 103 (6) (1984) 1170–1175.
[24] S.Y. Lim, M. Montakhab, H. Nouri, Economic dispatch of power system using particle swarm optimization with constriction factor, Int. J. Innov. Energy Syst. Power 4 (2) (2009) 29–34.
[25] D.K. He, F.L. Wang, Z.Z. Mao, Hybrid genetic algorithm for economic dispatch with valve-point effect, Electr. Power Syst. Res. 78 (4) (2008) 626–633.
[26] P. Subbaraj, R. Rengaraj, S. Salivahanan, Enhancement of combined heat and power economic dispatch using self adaptive real-coded genetic algorithm, Appl. Energy 86 (6) (2009) 915–921.
[27] N. Noman, H. Iba, Differential evolution for economic load dispatch problems, Electr. Power Syst. Res. 78 (8) (2008) 1322–1331.
[28] T. Niknam, A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem, Appl. Energy 87 (1) (2009) 327–339.
[29] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Piscataway, NJ, IEEE Service Center, 1995, pp. 39–43.
[30] M.S. Arumugam, M.V.C. Rao, On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems, Appl. Soft Comput. 8 (1) (2008) 324–336.
[31] T. Niknam, H. Doagou Mojarrad, M. Nayeripour, A new fuzzy adaptive particle swarm optimization for non-smooth economic dispatch, Energy 35 (4) (2010) 1764–1778.
[32] T. Niknam, M. Nayeripour, J. Olamaei, A. Arefi, An efficient hybrid evolutionary optimization algorithm for daily Volt/Var control at distribution system including DGs, Int. Rev. Electr. Eng. 3 (3) (2008) 1–11.
[33] J. Olamaei, T. Niknam, G. Gharehpetian, Application of particle swarm optimization for distribution feeder reconfiguration considering distributed generators, Appl. Math. Comput. J. 200 (1-2) (2008) 575–586.
[34] T. Niknam, B. Amiri, An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis, Appl. Soft Comput. 10 (1) (2010) 183–197.
[35] P. Bajpai, S.N. Singh, Fuzzy adaptive particle swarm optimization for bidding strategy in uniform price spot market, IEEE Trans. Power Syst. 22 (4) (2007) 2152–2160.
[36] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of Congress Evolutionary Computation 3, IEEE Press, vol. 3, issue no. 3, 1999, pp. 1931–1938.
[37] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[38] X. Yuan, A. Su, Y. Yuan, H. Nie, L. Wang, An improved PSO for dynamic load dispatch of generators with valve-point effects, Energy 34 (1) (2009) 67–74.
[39] C. Jiejin, M. Xiaoqian, L. Lixiang, P. Haipeng, Chaotic particle swarm optimization for economic dispatch considering the generator constraints, Energy Convers. Manage. 48 (2007) 645–653.
[40] A. Selvakumar, K. Thanushkodi, A new particle swarm optimization solution to nonconvex economic dispatch problems, IEEE Trans. Power Syst. 22 (1) (2007) 42–51.
[41] K.T. Chaturvedi, M. Pandit, L. Srivastava, Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch, IEEE Trans. Power Syst. 23 (3) (2008) 1079–1087.
[42] S. Pothiya, I. Ngamroo, W. Kongprawechnon, Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints, Energy Convers. Manage. 49 (4) (2008) 506–516.
[43] A. Selvakumar, K. Thanushkodi, Optimization using civilized swarm: solution to economic dispatch with multiple minima, Electr. Power Syst. Res. 79 (1) (2009) 8–16.
[44] B.K. Panigrahi, V.R. Pandi, S. Das, Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Convers. Manage. 49 (6) (2008) 1407–1415.
[45] N. Sinha, R. Chakrabarti, P.K. Chattopadhyay, Evolutionary programming techniques for economic load dispatch, IEEE Trans. Evol. Comput. 7 (1) (2003) 83–94.
[46] C.L. Chiang, Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels, IEEE Trans. Power Syst. 20 (4) (2005) 1690–1699.
[47] N. Amjady, H. Nasiri-Rad, 'Solution of nonconvex and nonsmooth economic dispatch by a new adaptive real coded genetic algorithm, Expert Syst. Appl. 37 (2010) 5239–5245.
[48] S. Khamsawang, S. Jiriwibhakorn, DSPSO–TSA for economic dispatch problem with nonsmooth and noncontinuous cost functions, Energy Convers. Manage. 51 (2) (2010) 365–375.