1

# Two Population-Based Heuristic Search Algorithms and Their Applications

Weirong Chen, Chaohua Dai and Yongkang Zheng
*Southwest JiaotongUniversity*
*China*

## 1. Introduction

Search is one of the most frequently used problem solving methods in artificial intelligence (AI) [1], and search methods are gaining interest with the increase in activities related to modeling complex systems [2, 3]. Since most practical applications involve objective functions which cannot be expressed in explicit mathematical forms and their derivatives cannot be easily computed, a better choice for these applications may be the direct search methods as defined below: *A direct search method for numerical optimization is any algorithm that depends on the objective function only through ranking a countable set of function values*. Direct search methods do not compute or approximate values of derivatives and remain popular because of their simplicity, flexibility, and reliability [4]. Among the direct search methods, hill climbing methods often suffer from local minima, ridges and plateaus. Hence, random restarts in search process can be used and are often helpful. However, high-dimensional continuous spaces are big places in which it is easy to get lost for random search. Resultantly, augmenting hill climbing with memory is applied and turns out to be effective [5]. In addition, for many real-world problems, an exhaustive search for solutions is not a practical proposition. It is common then to resort to some kind of heuristic approach as defined below: *heuristic search algorithm for tackling optimization problems is any algorithm that applies a heuristic to search through promising solutions in order to find a good solution.* This heuristic search allows the bypass of the "combinatorial explosion" problem [6]. Those techniques discussed above are all classified into heuristics involved with random move, population, memory and probability model [7]. Some of the best-known heuristic search methods are genetic algorithm (GA), tabu search and simulated annealing, etc.. A standard GA has two drawbacks: premature convergence and lack of good local search ability [8]. In order to overcome these disadvantages of GA in numerical optimization problems, differential evolution (DE) algorithm has been introduced by Storn and Price [9].
In the past 20 years, swarm intelligence computation [10] has been attracting more and more attention of researchers, and has a special connection with the evolution strategy and the genetic algorithm [11]. Swarm intelligence is an algorithm or a device and illumined by the social behavior of gregarious insects and other animals, which is designed for solving distributed problems. There is no central controller directing the behavior of the swarm; rather, these systems are self-organizing. This means that the complex and constructive collective behavior emerges from the individuals (agents) who follow some simple rules and

communicate with each other and their environments. Swarms offer several advantages over traditional systems based on deliberative agents and central control: specifically robustness, flexibility, scalability, adaptability, and suitability for analysis. Since 1990's, two typical swarm intelligence algorithms have emerged. One is the particle swarm optimization (PSO) [12], and the other is the ant colony optimization (ACO) [13].
In this chapter, two recently proposed swarm intelligence algorithms are introduced. They are seeker optimization algorithm (SOA) [3, 14-19] and stochastic focusing search (SFS) [20, 21], respectively.

## 2. Seeker Optimization Algorithm (SOA) and its applications

### 2.1 Seeker Optimization Algorithm (SOA) [3, 14-19]

Human beings are the highest-ranking animals in nature. Optimization tasks are often encountered in many areas of human life [6], and the search for a solution to a problem is one of the basic behaviors to all mankind [22]. The algorithm herein just focuses on human behaviors, especially human searching behaviors, to be simulated for real-parameter optimization. Hence, the seeker optimization algorithm can also be named as human team optimization (HTO) algorithm or human team search (HTS) algorithm. In the SOA, optimization process is treated as a search of optimal solution by a seeker population.

#### 2.1.1 Human searching behaviors

Seeker optimization algorithm (SOA) models the human searching behaviors based on their memory, experience, uncertainty reasoning and communication with each other. The algorithm operates on a set of solutions called seeker population (i.e., swarm), and the individual of this population are called seeker (i.e., agent). The SOA herein involves the following four human behaviours.

*A. Uncertainty Reasoning behaviours*

In the continuous objective function space, there often exists a neighborhood region close to the extremum point. In this region, the function values of the variables are proportional to their distances from the extremum point. It may be assumed that better points are likely to be found in the neighborhood of families of good points. In this case, search should be intensified in regions containing good solutions through focusing search [2]. Hence, it is believed that one may find the near optimal solutions in a narrower neighborhood of the point with lower objective function value and find them in a wider neighborhood of the point with higher function value.
"Uncertainty" is considered as a situational property of phenomena [23], and precise quantitative analyses of the behavior of humanistic systems are not likely to have much relevance to the real-world societal, political, economic, and other type of problems. Fuzzy systems arose from the desire to describe complex systems with linguistic descriptions, and a set of fuzzy control rules is a linguistic model of human control actions directly based on a human thinking about the operation. Indeed, the pervasiveness of fuzziness in human thought processes suggests that it is this fuzzy logic that plays a basic role in what may well be one of the most important facets of human thinking [24]. According to the discussions on the above human focusing search, the uncertainty reasoning of human search could be described by natural linguistic variables and a simple fuzzy rule as "If {*objective function value* is *small*} (i.e., *condition* part), Then {*step length* is *short*} (i.e., *action* part)". The

understanding and linguistic description of the human search make a fuzzy system a good candidate for simulating human searching behaviors.

*B. Egotistic Behavior*

Swarms (i.e., seeker population here) are a class of entities found in nature which specialize in mutual cooperation among them in executing their routine needs and roles [25]. There are two extreme types of co-operative behavior. One, *egotistic,* is entirely pro-self and another, *altruistic,* is entirely pro-group [26]. Every person, as a single sophisticated agent, is uniformly egotistic, believing that he should go toward his personal best position $\vec{p}_{i,best}$ through cognitive learning [27].

*C. Altruistic Behavior*

The altruistic behavior means that the swarms co-operate explicitly, communicate with each other and adjust their behaviors in response to others to achieve the desired goal. Hence, the individuals exhibit entirely pro-group behavior through social learning and simultaneously move to the neighborhood's historical best position or the neighborhood's current best position. As a result, the move expresses a self-organized aggregation behavior of swarms [28]. The aggregation is one of the fundamental self-organization behaviors of swarms in nature and is observed in organisms ranging from unicellular organisms to social insects and mammals [29]. The positive feedback of self-organized aggregation behaviors usually takes the form of attraction toward a given signal source [28]. For a "black-box" problem in which the ideal global minimum value is unknown, the neighborhood's historical best position or the neighborhood's current best position is used as the only attraction signal source for the self-organized aggregation behavior.

*C. Pro-Activeness Behavior*

Agents (i.e., seekers here) enjoy the properties of pro-activeness: agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative [30]. Furthermore, future behavior can be predicted and guided by past behavior [31]. As a result, the seekers may be pro-active to change their search directions and exhibit goal-directed behaviors according to the response to his past behaviors.

### 2.1.2 Implementation of Seeker Optimization Algorithm

Seeker optimization algorithm (SOA) operates on a search population of $s$ $D$-dimensional position vectors, which encode the potential solutions to the optimization problem at hand. The position vectors are represented as $\vec{x}_i = [x_{i1}, \cdots, x_{ij}, \cdots, x_{iD}]$, $i$=1, 2, $\cdots$, $s$, where $x_{ij}$ is the $j$th element of $\vec{x}_i$ and $s$ is the population size. Assume that the optimization problems to be solved are minimization problems.

The main steps of SOA are shown as Fig. 1. In order to add a social component for social sharing of information, a neighborhood is defined for each seeker. In the present studies, the population is randomly divided into three subpopulations (all the subpopulations have the same size), and all the seekers in the same subpopulation constitute a neighborhood. A search direction $\vec{d}_i(t) = [d_{i1}, \cdots, d_{iD}]$ and a step length vector $\vec{\alpha}_i(t) = [\alpha_{i1}, \cdots, \alpha_{iD}]$ are computed (see Section 1.1.3 and 1.1.4) for the $i$th seeker at time step $t$, where $\alpha_{ij}(t) \geq 0$, $d_{ij}(t) \in$ {-1,0,1}, $i$=1,2, $\cdots$,$s$; $j$=1,2, $\cdots$,$D$. When $d_{ij}(t) = 1$, it means that the $i$-th seeker goes towards the positive direction of the coordinate axis on the dimension $j$; when $d_{ij}(t) = -1$, the seeker goes towards the negative direction; when $d_{ij}(t)=0,$ the seeker stays at the current position on the corresponding dimension. Then, the *j*th element of the *i*th seeker's position is updated by:

$$x_{ij}(t+1)=x_{ij}(t)+\alpha_{ij}(t)d_{ij}(t) \tag{1}$$

Since the subpopulations are searching using their own information, they are easy to converge to a local optimum. To avoid this situation, an inter-subpopulation learning strategy is used, i.e., the worst two positions of each subpopulation are combined with the best position of each of the other two subpopulations by the following binomial crossover operator:

$$x_{k_n j,\text{worst}}=\begin{cases} x_{lj,\text{best}} & \text{if } R_j \le 0.5 \\ x_{k_n j,\text{worst}} & \text{else} \end{cases} \tag{2}$$

where $R_j$ is a uniformly random real number within [0,1], $x_{k_n j,\text{worst}}$ is denoted as the *j*th element of the *n*th worst position in the *k*th subpopulation, $x_{lj,\text{best}}$ is the *j*th element of the best position in the *l*th subpopulation, the indices *k*, *n*, *l* are constrained by the combination $(k,n,l)\in\{(1,1,2), (1,2,3), (2,1,1), (2,2,3), (3,1,1), (3,2,2)\}$, and $j=1,\cdots,D$. In this way, the good information obtained by each subpopulation is exchanged among the subpopulations and then the diversity of the population is increased.

### 2.1.3 Search direction

The gradient has played an important role in the history of search methods [32]. The search space may be viewed as a gradient field [33], and a so-called empirical gradient (EG) can be determined by evaluating the response to the position change especially when the objective function is not be available in a differentiable form at all [5]. Then, the seekers can follow an EG to guide their search. Since the search directions in the SOA does not involve the magnitudes of the EGs, a search direction can be determined only by the signum function of a better position minus a worse position. For example, an empirical search direction $\vec{d}=sign(\vec{x}'-\vec{x}'')$ when $\vec{x}'$ is better than $\vec{x}''$, where the function $sign(\cdot)$ is a signum function on each element of the input vector. In the SOA, every seeker *i* ($i=1,2,\cdots,s$) selects his search direction based on several EGs by evaluating the current or historical positions of himself or his neighbors. They are detailed as follows.

According to the egotistic behavior mentioned above, an EG from $\vec{x}_i(t)$ to $\vec{p}_{i,best}(t)$ can be involved for the *i*th seeker at time step *t*. Hence, each seeker *i* is associated with an empirical direction called as egotistic direction $\vec{d}_{i,ego}(t)=[d_{i1,ego},d_{i2,ego},\cdots,d_{iD,ego}]$:

$$\vec{d}_{i,ego}(t)=sign(\vec{p}_{i,best}(t)-\vec{x}_i(t)) \tag{3}$$

On the other hand, based on the altruistic behavior, each seeker *i* is associated with two optional altruistic direction, i.e., $\vec{d}_{i,alt_1}(t)$ and $\vec{d}_{i,alt_2}(t)$:

$$\vec{d}_{i,alt_1}(t)=sign(\vec{g}_{i,best}(t)-\vec{x}_i(t)) \tag{4}$$

$$\vec{d}_{i,alt_2}(t)=sign(\vec{l}_{i,best}(t)-\vec{x}_i(t)) \tag{5}$$

where $\vec{g}_{i,best}(t)$ represents the neighborhood's historical best position up to the time step *t*, $\vec{l}_{i,best}(t)$ represents the neighborhood's current best position. Here, the neighborhood is the one to which the *i*th seeker belongs.

Moreover, according to the pro-activeness behavior, each seeker *i* is associated with an empirical direction called as pro-activeness direction $\vec{d}_{i,pro}(t)$:

$$\vec{d}_{i,pro}(t) = sign(\vec{x}_i(t_1) - \vec{x}_i(t_2)) \tag{6}$$

where $t_1, t_2 \in \{t, t-1, t-2\}$, $\vec{x}_i(t_1)$ and $\vec{x}_i(t_2)$ are the best one and the worst one in the set $\{\vec{x}_i(t), \vec{x}_i(t-1), \vec{x}_i(t-2)\}$ respectively.

According to human rational judgment, the actual search direction of the *i*th seeker, $\vec{d}_i(t) = [d_{i1}, d_{i2}, \cdots, d_{iD}]$, is based on a compromise among the aforementioned four empirical directions, i.e., $\vec{d}_{i,ego}(t)$, $\vec{d}_{i,alt_1}(t)$, $\vec{d}_{i,alt_2}(t)$ and $\vec{d}_{i,pro}(t)$. In this study, the *j*th element of $\vec{d}_i(t)$ is selected applying the following proportional selection rule (shown as Fig. 2):

$$d_{ij} = \begin{cases} 0 & \text{if } r_j \le p_j^{(0)} \\ 1 & \text{if } p_j^{(0)} < r_j \le p_j^{(0)} + p_j^{(1)} \\ -1 & \text{if } p_j^{(0)} + p_j^{(1)} < r_j \le 1 \end{cases} \tag{7}$$

where *i*=1,2,⋯,*s*, *j*=1,2,⋯,*D*, $r_j$ is a uniform random number in [0,1], $p_j^{(m)}$ $(m \in \{0,1,-1\})$ is defined as follows: In the set $\{d_{ij,ego}, d_{ij,alt_1}, d_{ij,alt_2}, d_{ij,pro}\}$ which is composed of the *j*th elements of $\vec{d}_{i,ego}(t)$, $\vec{d}_{i,alt_1}(t)$, $\vec{d}_{i,alt_2}(t)$ and $\vec{d}_{i,pro}(t)$, let $num^{(1)}$ be the number of "1", $num^{(-1)}$ be the number of "-1", and $num^{(0)}$ be the number of "0", then $p_j^{(1)} = \frac{num^{(1)}}{4}, p_j^{(-1)} = \frac{num^{(-1)}}{4}$, $p_j^{(0)} = \frac{num^{(0)}}{4}$. For example, if $d_{ij,ego} = 1, d_{ij,alt_1} = -1$, $d_{ij,alt_2} = -1, d_{ij,pro} = 0$, then $num^{(1)}$ =1, $num^{(-1)}$=2, and $num^{(0)}$=1. So, $p_j^{(1)} = \frac{1}{4}, p_j^{(-1)} = \frac{2}{4}, p_j^{(0)} = \frac{1}{4}$.

### 2.1.4 Step length

In the SOA, only one fuzzy rule is used to determine the step length, namely, "If {*objective function value* is *small*} (i.e., *condition* part), Then {*step length* is *short*} (i.e., *action* part)". Different optimization problems often have different ranges of fitness values. To design a fuzzy system to be applicable to a wide range of optimization problems, the fitness values of all the seekers are descendingly sorted and turned into the sequence numbers from 1 to *s* as the inputs of fuzzy reasoning. The linear membership function is used in the conditional part (fuzzification) since the universe of discourse is a given set of numbers, i.e., {1,2,⋯,*s*}. The expression is presented as (8).

$$\mu_i = \mu_{\max} - \frac{s - I_i}{s-1}(\mu_{\max} - \mu_{\min}) \tag{8}$$

where $I_i$ is the sequence number of $\vec{x}_i(t)$ after sorting the fitness values, $\mu_{\max}$ is the maximum membership degree value which is assigned by the user and equal to or a little less than 1.0. Generally, $\mu_{\max}$ is set at 0.95.

In the action part (defuzzification), the Gaussian membership function $\mu(\alpha_{ij}) = e^{-\alpha_{ij}^2/(2\delta_j^2)}$ $(i = 1,\cdots,s; j = 1,\cdots,D)$ is used for the *j*th element of the *i*th seeker's step length. For the Bell function, the membership degree values of the input variables beyond [-3$\delta_j$, 3$\delta_j$] are less than 0.0111 ($\mu(\pm 3\delta_j)$=0.0111), which can be neglected for a linguistic atom [34]. Thus, the minimum value $\mu_{\min}$=0.0111 is fixed. Moreover, the parameter $\delta_j$ of the Gaussian membership function is the *j*th element of the vector $\vec{\delta} = [\delta_1,\cdots,\delta_D]$ which is given by:

$$\vec{\delta} = \omega \cdot abs(\vec{x}_{best} - \vec{x}_{rand}) \tag{9}$$

where $abs(\cdot)$ returns an output vector such that each element of the vector is the absolute value of the corresponding element of the input vector, the parameter $\omega$ is used to decrease the *step length* with time step increasing so as to gradually improve the search precision. In general, the $\omega$ is linearly decreased from 0.9 to 0.1 during a run. The $\vec{x}_{best}$ and $\vec{x}_{rand}$ are the best seeker and a randomly selected seeker in the *same* subpopulation to which the *i*th seeker belongs, respectively. Notice that $\vec{x}_{rand}$ is different from $\vec{x}_{best}$, and $\vec{\delta}$ is shared by all the seekers in the same subpopulation. Then, the *action* part of the fuzzy reasoning (shown in Fig. 3) gives the *j*th element of the *i*th seeker's step length $\vec{\alpha}_i = [\alpha_{i1},\cdots,\alpha_{iD}]$ ($i$=1,2,···,$s$; $j$=1,2,···,$D$):

$$\alpha_{ij} = \delta_j \sqrt{-\log(RAND(\mu_i, 1))} \tag{10}$$

where $\delta_j$ is the *j*th element of the vector $\vec{\delta}$ in (9), the function log(·) returns the natural logarithm of its input, the function $RAND(\mu_i,1)$ returns a uniform random number within the range of [$\mu_i$,1] which is used to introduce the randomicity for each element of $\vec{\alpha}_i$ and improve local search capability.

### 2.1.5 Further analysis on the SOA

Unlike GA, SOA conducts focusing search by following the promising empirical directions until to converge to the optimum for as few generations as possible. In this way, it does not easily get lost and then locates the region in which the global optimum exists.

Although the SOA uses the same terms of the personal/population best position as PSO and DE, they are essentially different. As far as we know, PSO is not good at choosing step length [35], while DE sometimes has a limited ability to move its population large distances across the search space and would have to face with stagnation puzzledom [36]. Unlike PSO and DE, SOA deals with search direction and step length, independently. Due to the use of fuzzy rule: "If {*fitness value* is *small*}, Then {*step length* is *short*}", the better the position of the seeker is, the shorter his step length is. As a result, from the worst seeker to the best seeker, the search is changed from a coarse one to a fine one, so as to ensure that the population can not only keep a good search precision but also find new regions of the search space. Consequently, at every time step, some seekers are better for "exploration", some others better for "exploitation". In addition, due to self-organized aggregation behavior and the decreasing parameter $\omega$ in (9), the feasible search range of the seekers is decreasing with time step increasing. Hence, the population favors "exploration" at the early stage and "exploitation" at the late stage. In a word, not only at every time step but also within the whole search process, the SOA can effectively balance exploration and exploitation, which could ensure the effectiveness and efficiency of the SOA [37].

According to [38], a "nearer is better (NisB)" property is almost always assumed: most of iterative stochastic optimization algorithms, if not all, at least from time to time look around a good point in order to find an even better one. Furthermore, the reference [38] also pointed out that an effective algorithm may perfectly switch from a NisB assumption to a "nearer is worse (NisW)" one, and vice-versa. In our opinion, SOA is potentially provided with the NisB property because of the use of fuzzy reasoning and can switch between a NisB assumption and a NisW one. The main reason lies in the following two aspects. On the one hand, the search direction of each seeker is based on a compromise among several empirical directions, and different seekers often learn from different empirical points on different dimensions instead of a single *good point* as mentioned by NisB assumption. On the other hand, uncertainty reasoning (fuzzy reasoning) used by SOA would let a seeker's step length "uncertain", which uncertainly lets a seeker nearer to a certain *good point*, or farer away from another certain *good point*. Both the two aspects can boost the diversity of the population. Hence, from Clerc's point of view [38], it is further proved that SOA is effective.

```
begin
$t \leftarrow 0$;
generating $s$ positions uniformly and randomly in search space;
repeat
    evaluating each seeker;
    computing $\vec{d}_i(t)$ and $\vec{\alpha}_i(t)$ for each seeker $i$;
    updating each seeker's position using (1);
    $t \leftarrow t+1$;
until the termination criterion is satisfied
end.
```

Fig. 1. The main step of the SOA.

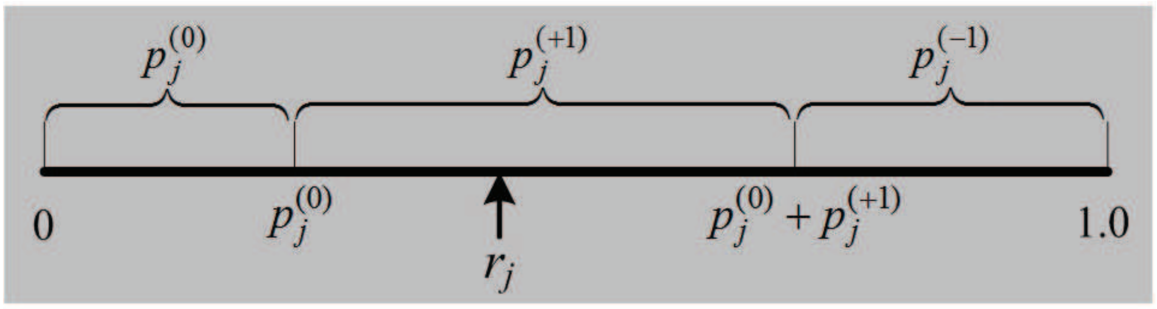


Fig. 2. The proportional selection rule of search directions

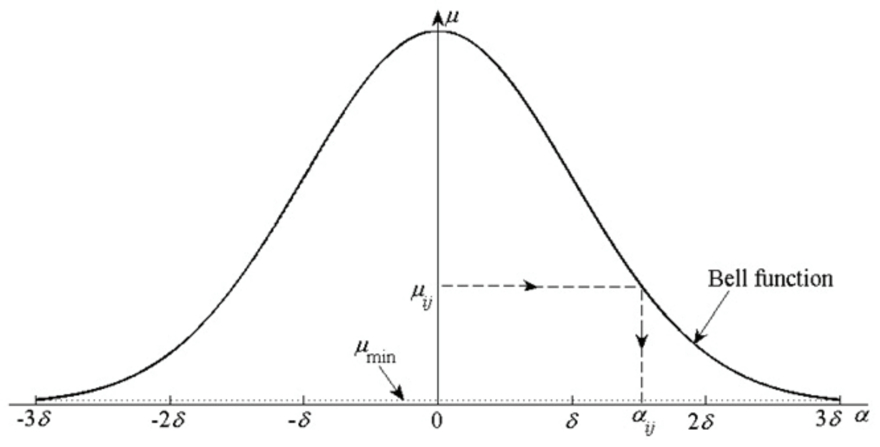


Fig. 3. The *action* part of the Fuzzy reasoning.

**2.2 SOA for benchmark function optimization (Refs.[3,16, 18)**

Twelve benchmark functions (listed in Table 1) are chosen from [39] to test the SOA with comparison of PSO-w (PSO with adaptive inertia weight) [40], PSO-cf (PSO with constriction factor) [41], CLPSO (comprehensive learning particle swarm optimizer) [42], the original DE [9], SACP-DE (DE with self-adapting control parameters) [39] and L-SaDE (the self-adaptive DE) [43]. The *Best*, *Mean* and *Std* (standard deviation) values of all the algorithms for each function over 30 runs are summarized in Table 2. In order to determine whether the results obtained by SOA are statistically different from the results generated by other algorithms, the *T*-tests are conducted and listed in Table 2, too. An *h* value of one indicates that the performances of the two algorithms are statistically different with 95% certainty, whereas *h* value of zero implies that the performances are not statistically different. The *CI* is confidence interval. The Table 2 indicates that SOA is suitable for solving the employed multimodal function optimizations with the smaller *Best*, *Mean* and *std* values than most of other algorithms for most of the functions. In addition, most of the *h* values are equal to one, and most of the *CI* values are less than zero, which shows that SOA is statistically superior to most of the other algorithms with the more robust performance. The details of the comparison results are as follows. Compared with PSO-w, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions. Compared with PSO-cf, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that PSO-cf also has the same *Best* values for the functions 2-4, 6, 11 and 12. Compared with CLPSO, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that CLPSO also has the same *Best* values for the functions 6, 7, 9, 11 and 12. Compared with SPSO-2007, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that SPSO-2007 also has the same *Best* values for the functions 7-12. Compared with DE, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that DE also has the same *Best* values for the functions 3, 6, 9, 11 and 12. Compared with SACP-DE, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that SACP-DE can also find the global optimal solutions for function 3 and has the same *Best* values for the functions 6, 7, 11 and 12. Compared with L-SaDE, SOA has the smaller *Best*, *Mean* and *std* values for all the twelve benchmark functions expect that L-SaDE can also find the global optimal solutions for function 3 and has the same *Best* values for the functions 6, 9 and 12.

| Functions | $n$ | $S$ | $f_{\min}$ |
|---|---|---|---|
| $f_1(\vec{x})=\sum_{i=1}^{n} ix_i^4+rand[0,1)$ | 30 | $[-1.28,1.2$ | $f_1(\vec{0})=0$ |
| $f_2(\vec{x})=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))\ +20+e$ | 30 | $[-32,32]^n$ | $f_2(\vec{0})=0$ |
| $f_3(\vec{x})=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$ | 30 | $[-600,600$ | $f_3(\vec{0})=0$ |
| $f_4(\vec{x})=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]$ $+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ | 30 | $[-50,50]^n$ | $f_4(-\vec{1})=0$ |
| $f_5(\vec{x})=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$ | 30 | $[-50,50]^n$ | $f_5(1,\ldots,1)=0$ |
| $f_6(\vec{x})=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$ | 4 | $[-5,5]^n$ | $f_6(0.1928,0.1908,0.1231,0.1358)=3.0749\times10^{-4}$ |
| $f_7(\vec{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$ | 2 | $[-5,5]^n$ | $f_7(-0.09,0.71)=-1.031628$ |
| $f_8(\vec{x})=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos(x_1)+10$ | 2 | $[-5,15]^n$ | $f_8(9.42,2.47)=0.$ |
| $f_9(\vec{x})=[1+(x_1+x_2+1)^2(19-14x_1+3x_1{}^2-14x_2+6x_1x_2+3x_2{}^2)]\times$ $[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1{}^2+48x_2-36x_1x_2+27x_2{}^2)]$ | 2 | $[-2,2]^n$ | $f_9(0,-1)=3$ |
| $f_{10}(\vec{x})=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2]$ | 3 | $[0,1]^n$ | $f_{10}(0.114,0.556,0$ |
| $f_{11}(\vec{x})=-\sum_{i=1}^{7}[(\vec{x}-a_i)(\vec{x}-a_i)^T+c_i]^{-1}$ | 4 | $[0,10]^n$ | $f_{11}(\approx\vec{4})=-10.40$ |
| $f_{12}(\vec{x})=-\sum_{i=1}^{10}[(\vec{x}-a_i)(\vec{x}-a_i)^T+c_i]^{-1}$ | 4 | $[0,10]^n$ | $f_{12}(\approx\vec{4})=-10.53$ |

Table 1. The employed benchmark functions.

| function | Index | PSO-ω | PSO-cf | CLPSO | SPSO-2007 | DE | SACP-DE | L-SaDE | SOA |
|---|---|---|---|---|---|---|---|---|---|
| 1 | *Best* | 2.7136e-3 | 1.0861e-3 | 3.3596e-3 | 7.6038e-3 | 1.8195e-3 | 3.7152e-3 | 1.0460e-3 | 4.0153e-5 |
| | *Mean* | 7.1299e-3 | 2.5423e-3 | 5.1258e-3 | 5.0229e-2 | 4.3505e-3 | 5.5890e-3 | 4.2653e-3 | 9.7068e-5 |
| | *Std* | 2.3404e-3 | 9.7343e-4 | 1.1883e-3 | 3.5785e-2 | 1.2317e-3 | 1.1868e-3 | 1.7366e-3 | 4.8022e-5 |
| | *h* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-0.0081 - 0.0060] | [-0.0029 - 0.0020] | [-0.0056 - 0.0045] | [-0.0663 - 0.0339] | [-0.0048 - 0.0037] | [-0.0060 - 0.0050] | [-0.0050 - 0.0034] | - |
| 2 | *Best* | 7.3196e-7 | 2.6645e-15 | 6.3072e-4 | 1.7780e+0 | 8.5059e-8 | 5.2355e-9 | 1.2309e-11 | 2.6645e-15 |
| | *Mean* | 1.7171e-6 | 8.0458e-1 | 8.2430e-4 | 3.1720e+0 | 1.6860e-7 | 1.12625e-8 | 7.0892e-11 | 2.6645e-15 |
| | *Std* | 8.8492e-7 | 7.7255e-1 | 1.2733e-4 | 9.1299e-1 | 7.3342e-8 | 4.1298e-9 | 4.1709e-11 | 0 |
| | *h* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-2.12e-6 - 1.32e-6] | [-1.1543 - 0.4549] | [-8.82e-4 - 7.67e-4] | [-3.5853 - 2.7587] | [-2.02e-7 - 1.35e-7] | [-1.31e-8 - 9.39e-9] | [-8.98e-11 -5.20e-11] | - |
| 3 | *Best* | 2.2204e-15 | 0 | 1.7472e-7 | 6.6613e-16 | 0 | 0 | 0 | 0 |
| | *Mean* | 8.3744e-3 | 1.9984e-2 | 2.4043e-6 | 1.0591e-2 | 4.9323e-4 | 0 | 0 | 0 |
| | *Std* | 7.7104e-3 | 2.1321e-2 | 3.6467e-6 | 1.1158e-2 | 2.2058e-3 | 0 | 0 | 0 |
| | *h* | 1 | 1 | 1 | 1 | 0 | - | - | - |
| | *CI* | [-0.0118 - 0.0049] | [-0.0296 - 0.0103] | [-4.06e-6 - 7.54e-7] | [-0.0156 - 0.0055] | [-0.0015 5.0527e-4] | - | - | - |
| 4 | *Best* | 1.2781e-13 | 1.5705e-32 | 1.8074e-7 | 5.2094e-22 | 2.9339e-15 | 2.2953e-17 | 2.5611e-21 | 1.5705e-32 |
| | *Mean* | 2.6878e-10 | 1.1402e-1 | 5.7391e-7 | 1.3483e+0 | 2.5516e-14 | 1.3700e-16 | 8.0092e-20 | 1.5808e-30 |
| | *Std* | 6.7984e-10 | 1.8694e-1 | 2.4755e-7 | 1.3321e+0 | 1.8082e-14 | 8.7215e-17 | 7.9594e-20 | 3.8194e-30 |
| | *h* | 0 | 1 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-5.75e-10 3.70e-11] | [-0.1986 - 0.0294] | [-6.86e-7 - 4.62e-7] | [-1.9513 - 0.7453] | [-3.4e-14 - 1.7e-14] | [-1.8e-16 - 9.8e-17] | [-1.12e-19 -4.41e-20] | - |
| 5 | *Best* | 1.6744e-12 | 1.3498e-30 | 4.2229e-6 | 1.0379e-19 | 2.5008e-14 | 3.8881e-16 | 1.0668e-21 | 6.1569e-32 |
| | *Mean* | 1.0990e-3 | 1.0987e-3 | 6.8756e-6 | 1.3031e+1 | 1.0165e-13 | 9.7736e-16 | 3.4614e-19 | 3.3345e-29 |
| | *Std* | 3.4744e-3 | 3.3818e-3 | 2.7299e-6 | 1.1416e+1 | 7.1107e-14 | 8.4897e-16 | 4.7602e-19 | 8.3346e-29 |
| | *h* | 0 | 0 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-0.0027 4.6368e-4] | [-0.0026 4.3212e-4] | [-8.11e-6 - 5.64e-6] | [-18.1990 - 7.8633] | [-1.3e-13 - 6.9e-14] | [-1.4e-15 - 5.9e-16] | [-5.62e-1 1.31e-19] | - |
| 6 | *Best* | 3.0750e-4 | 3.0749e-4 | 3.0749e-4 | 3.0749e-4 | 3.0749e-4 | 3.0749e-4 | 3.0749e-4 | 3.0749e-4 |
| | *Mean* | 4.9063e-4 | 4.4485e-4 | 3.5329e-4 | 4.9463e-4 | 4.4485e-4 | 3.0750e-4 | 3.0750e-4 | 3.0749e-4 |
| | *Std* | 3.8608e-4 | 3.3546e-4 | 2.0478e-4 | 1.7284e-4 | 3.3546e-4 | 3.0191e-9 | 2.8726e-9 | 9.6334e-20 |
| | *h* | 1 | 0 | 0 | 1 | 0 | 1 | 1 | - |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *CI* | [-3.57e-4 - 9.49e-5] | [-2.89e-4 1.45e-5] | [-1.39e-4 4.69e-5] | [-2.65e-4 - 1.09e-4] | [-2.89e-4 1.45e-5] | [-1.15e-8 - 8.74e-9] | [-1.14e-8 - 8.7e-9] | - |
| 7 | *Best* | -1.031626 | -1.031627 | -1.031628 | -1.031628 | -1.031627 | -1.031628 | -1.031627 | -1.031628 |
| | *Mean* | -1.031615 | -1.031612 | -1.031617 | -1.031627 | -1.031619 | -1.031617 | -1.031613 | -1.031628 |
| | *Std* | 8.6069e-6 | 7.8874e-6 | 7.4529e-6 | 3.5817e-6 | 8.4157e-6 | 8.0149e-6 | 9.0097e-6 | 7.6401e-13 |
| | *h* | 1 | 1 | 1 | 0 | 1 | 1 | 1 | - |
| | *CI* | [-1.72e-5 - 9.49e-6] | [-2.00e-5 - 1.28e-5] | [-1.53e-5 - 8.54e-6] | [-2.47e-6 7.76e-6] | [-1.28e-5 - 5.21e-6] | [-1.52e-5 - 7.99e-6] | [-1.92e-5 - 1.11e-5] | - |
| 8 | *Best* | 3.97890e-1 | 3.97898e-1 | 3.97897e-1 | 3.97887e-1 | 3.97902e-1 | 3.97888e-1 | 3.97889e-1 | 3.97887e-1 |
| | *Mean* | 3.97942e-1 | 3.97939e-1 | 3.97947e-1 | 3.97892e-1 | 3.97947e-1 | 3.97932e-1 | 3.97941e-1 | 3.97887e-1 |
| | *Std* | 3.3568e-5 | 3.0633e-5 | 3.1612e-5 | 1.8336e-5 | 3.0499e-5 | 3.3786e-5 | 3.76524e-5 | 1.2874e-7 |
| | *h* | 1 | 1 | 1 | 0 | 1 | 1 | 1 | - |
| | *CI* | [-6.95e-5 - 3.93e-5] | [-6.52e-5 - 3.74e-5] | [-7.37e-5 4.51e-5] | [-1.277e-5 3.92e-6] | [-7.38e-5 - 4.62e-5] | [-6.00e-5 - 2.94e-5] | [-7.09e-5 - 3.69e-5] | - |
| 9 | *Best* | 3.0000 | 3.0000 | 3 | 3 | 3 | 3.0000 | 3 | 3 |
| | *Mean* | 3.0000 | 3.0000 | 3.0000 | 3.0000 | 3 | 3.0000 | 3.0000 | 3 |
| | *Std* | 4.0898e-12 | 3.1875e-12 | 1.7278e-13 | 2.6936e-12 | 9.9103e-15 | 2.6145e-8 | 5.4283e-13 | 2.7901e-15 |
| | *h* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-5.1e-12 - 1.5e-12] | [-4.5e-12 - 1.61e-12] | [-3.1e-13 - 1.6e-13] | [-2.7e-12 - 2.6e-13] | [-8.5e-14 - 7.6e-14] | [-2.6e-8 - 2.6e-9] | [-6.4e-13 - 1.5e-13] | - |
| 10 | *Best* | -3.86174 | -3.86260 | -3.86254 | -3.86278 | -3.86256 | -3.86251 | -3.86228 | -3.86278 |
| | *Mean* | -3.86120 | -3.86142 | -3.86131 | -3.86196 | -3.86115 | -3.86137 | -3.86104 | -3.86278 |
| | *Std* | 4.1892e-4 | 7.0546e-4 | 6.6908e-4 | 3.6573e-3 | 7.9362e-4 | 6.1290e-4 | 6.8633e-4 | 2.0402e-15 |
| | *h* | 1 | 1 | 1 | 0 | 1 | 1 | 1 | - |
| | *CI* | [-0.0018 - 0.0014] | [-0.0017 - 0.0010] | [-0.0018 - 0.0012] | [-0.0025 8.3672e-4] | [-0.0020 - 0.0013] | [-0.0017 - 0.0011] | [-0.0021 - 0.0014] | - |
| 11 | *Best* | -1.0403e+1 | -1.0403e+1 | -1.0403e+1 | -1.0403e+1 | -1.0403e+1 | -1.0403e+1 | -1.0402e+1 | -1.0403e+1 |
| | *Mean* | -8.8741e+0 | -9.3713e+0 | -7.5794e+0 | -8.5881e+0 | -1.0403e+1 | -1.0307e+1 | -1.0307e+1 | -1.0403e+1 |
| | *Std* | 3.2230e+0 | 2.5485e+0 | 3.6087e+0 | 3.2342e+0 | 6.6816e-7 | 1.9198e-1 | 1.6188e-1 | 5.8647e-11 |
| | *h* | 1 | 0 | 1 | 1 | 1 | 1 | 1 | - |
| | *CI* | [-2.9785 - 0.0791] | [-2.1852 0.1220] | [-4.4570 - 1.1900] | [-3.2788 - 0.3508] | [-7.32e-7 -1.27e-7] | [-0.1828 - 0.0090] | [-0.1692 - 0.0226] | - |
| 12 | *Best* | -1.0536e+1 | -1.0536e+1 | -1.0536e+1 | -1.0536e+1 | -1.0536e+1 | -1.0536e+1 | -1.0534e+1 | -1.0536e |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | +1 |
| *Mean* | -8.4159e+0 | -8.6726e+0 | -9.2338e+0 | -9.7313e+0 | -1.0536e+1 | -1.0432e+1 | -1.0437e+1 | -1.0536e+1 |
| *Std* | 3.4860e+0 | 3.3515e+0 | 2.7247e+0 | 2.0607e+0 | 4.3239e-7 | 3.1761e-1 | 1.3003e-1 | 3.0218e-11 |
| *h* | 1 | 1 | 1 | 0 | 1 | 0 | 1 | - |
| *CI* | [-3.6885 - 0.5526] | [-3.3809 - 0.3467] | [-2.5360 - 0.0692] | [-1.7379 0.1277] | [-4.86e-7 - 9.43e-8] | [-0.2481 0.0394] | [-0.1586 - 0.0409] | - |

Table 2. The Comparisons of SOA with Other Evolutionary Methods on Benchmark Functions

## 2.3 SOA for optimal reactive power dispatch (Ref.[16])

### 2.3.1 Problem formulation

The objective of the reactive power optimization is to minimize the active power loss in the transmission network, which can be defined as follows:

$$P_{\text{loss}} = f(\vec{x}_1, \vec{x}_2) = \sum_{k \in N_E} g_k (V_i^2 + V_j^2 - 2V_i V_j \cos\theta_{ij}) \tag{11}$$

Subject to

$$\begin{cases} P_{Gi} - P_{Di} = V_i \sum\limits_{j \in N_i} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) & i \in N_0 \\ Q_{Gi} - Q_{Di} = V_i \sum\limits_{j \in N_i} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) & i \in N_{PQ} \\ V_i^{\min} \le V_i \le V_i^{\max} & i \in N_B \\ T_k^{\min} \le T_k \le T_k^{\max} & k \in N_T \\ Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max} & i \in N_G \\ Q_{Ci}^{\min} \le Q_{Ci} \le Q_{Ci}^{\max} & i \in N_C \\ S_l \le S_l^{\max} & l \in N_l \end{cases} \tag{12}$$

where $f(\vec{x}_1, \vec{x}_2)$ denotes the active power loss function of the transmission network, $\vec{x}_1$ is the control variable vector $[V_G \; K_T \; Q_C]^T$, $\vec{x}_2$ is the dependent variable vector $[V_L \; Q_G]^T$, $V_G$ is the generator voltage (continuous), $T_k$ is the transformer tap (integer), $Q_C$ is the shunt capacitor/inductor (integer), $V_L$ is the load-bus voltage, $Q_G$ is the generator reactive power, $k$=($i$,$j$), $i \in N_B$, $j \in N_i$, $g_k$ is the conductance of branch $k$, $\theta_{ij}$ is the voltage angle difference between bus $i$ and $j$, $P_{Gi}$ is the injected active power at bus $i$, $P_{Di}$ is the demanded active power at bus $i$, $V_i$ is the voltage at bus $i$, $G_{ij}$ is the transfer conductance between bus $i$ and $j$, $B_{ij}$ is the transfer susceptance between bus $i$ and $j$, $Q_{Gi}$ is the injected reactive power at bus $i$, $Q_{Di}$ is the demanded reactive power at bus , $N_E$ is the set of numbers of network branches, $N_{PQ}$ is the set of numbers of PQ buses, $N_B$ is the set of numbers of total buses, $N_i$ is the set of numbers of buses adjacent to bus $i$ (including bus $i$), $N_0$ is the set of numbers of total buses excluding slack bus, $N_C$ is the set of numbers of possible reactive power source installation buses, $N_G$ is the set of numbers of generator buses, $N_T$ is the set of numbers of transformer branches, $S_l$ is the power flow in branch $l$, the superscripts "min" and "max" in equation (12) denote the corresponding lower and upper limits, respectively.

The first two equality constraints in (12) are the power flow equations. The rest inequality constraints are used for the restrictions of reactive power source installation, reactive generation, transformer tap-setting, bus voltage and power flow of each branch.

Control variables are self-constrained, and dependent variables are constrained using penalty terms to the objective function. So the objective function is generalized as follows:

$$f = P_{loss} + \lambda_V \sum_{N_V^{\lim}} \Delta V_L^2 + \lambda_Q \sum_{N_Q^{\lim}} \Delta Q_G^2 \tag{13}$$

where $\lambda_V$, $\lambda_Q$ are the penalty factors, $N_V^{\lim}$ is the set of numbers of load-buses on which voltage outside limits, $N_Q^{\lim}$ is the set of numbers of generator buses on which injected reactive power outside limits, $\Delta V_L$ and $\Delta Q_G$ are defined as:

$$\Delta V_L = \begin{cases} V_L^{\min} - V_L & \text{if } V_L < V_L^{\min} \\ V_L - V_L^{\max} & \text{if } V_L > V_L^{\max} \end{cases} \tag{14}$$

$$\Delta Q_G = \begin{cases} Q_G^{\min} - Q_G & \text{if } Q_G < Q_G^{\min} \\ Q_G - Q_G^{\max} & \text{if } Q_G > Q_G^{\max} \end{cases} \tag{15}$$

### 2.3.2 Implementation of SOA for reactive power optimization

The basic form of the proposed SOA algorithm can only handle continuous variables. However, both tap position of transformations and reactive power source installation are discrete or integer variables in optimal reactive power dispatch problem. To handle integer variables without any effect on the implementation of SOA, the seekers will still search in a continuous space regardless of the variable type, and then truncating the corresponding dimensions of the seekers' real-value positions into the integers [44] is only performed in evaluating the objective function.

The fitness value of each seeker is calculated by using the objective function in (13). The real-value position of the seeker consists of three parts: generator voltages, transformer taps and shunt capacitors/inductors. After the update of the position, the main program is turned to the sub-program for evaluating the objective function where the latter two parts of the position are truncated into the corresponding integers as [44]. Then, the real-value position is changed into a mixed-variable vector which is used to calculate the objective function value by equation (13) based on Newton-Raphson power flow analysis [45]. The reactive power optimization based on SOA can be described as follows [16].

**Step 1.** Read the parameters of power system and the proposed algorithm, and specify the lower and upper limits of each variable.

**Step 2.** Initialize the positions of the seekers in the search space randomly and uniformly. Set the time step $t$=0.

**Step 3.** Calculate the fitness values of the initial positions using the objective function in (13) based on the results of Newton-Raphson power flow analysis [45]. The initial historical best position among the population is achieved. Set the personal historical best position of each seeker to his current position.

**Step 4.** Let $t = t + 1$ .
**Step 5.** Select the neighbors of each seeker.
**Step 6.** Determine the search direction and step length for each seeker, and update his position.
**Step 7.** Calculate the fitness values of the new positions using the objective function based on the Newton-Raphson power flow analysis results. Update the historical best position among the population and the historical best position of each seeker.
**Step 8.** Go to Step 4 until a stopping criterion is satisfied.

### 2.3.3 Simulation results

To evaluate the effectiveness and efficiency of the proposed SOA-based reactive power optimization approach, standard IEEE 57-bus power system is used.

Since proposed in 1995, PSO [46] and DE [9, 47] have received increasing interest from the evolutionary computation community as two of the relatively new and powerful population-based heuristic algorithms, and they both have been successfully applied to reactive power optimization problems [12, 48-53]. So, the proposed method is compared mainly with the two algorithms and their recently modified versions.

Since the original PSO proposed in [46] is prone to suffer from the so-called "explosion" phenomena [41], two improved versions of PSO: PSO with adaptive inertia weight (PSO-w) and PSO with a constriction factor (PSO-cf), were proposed by Shi, et al. [40] and Clerc, et al. [41], respectively. Considering that the PSO algorithm may easily get trapped in a local optimum when solving complex multimodal problems, Liang, et al. [42] proposed a variant of PSO called comprehensive learning particle swarm optimizer (CLPSO), which is adept at complex multimodal problems. Furthermore, in the year of 2007, Clerc, et al. [54] developed a "real standard" version of PSO, SPSO-07, which was specially prepared for the researchers to compare their algorithms. So, the compared PSOs includes PSO-w(learning rate $c_1$ = $c_2$=2, inertia weight linearly decreased from 0.9 to 0.4 with run time increasing, the maximum velocity $v_{\max}$ is set at 20% of the dynamic range of the variable on each dimension) [40], PSO-cf ($c_1$= $c_2$=2.01 and constriction factor $\chi$=0.729844) [41], CLPSO(its parameters follow the suggestions from [42] except that the refreshing gap $m$=2) and SPSO-07 [54].

Since the control parameters and learning strategies in DE are highly dependent on the problems under consideration, and it is not easy to select the correct parameters in practice, Brest, et al. [39] presented a version of DE with self-adapting control parameters (SACP-DE) based on the self-adaptation of the two control parameters: the crossover rate *CR* and the scaling factor *F*, while Qin, et al. [43] proposed a self-adaptive differential evolution (SaDE) where the choice of learning strategy and the two control parameters *F* and *CR* are not required to be pre-specified. So, the compared set of DEs consists of the original DE (DE: DE/rand/1/bin, *F*=0.5, *CR*=0.9) [9]), SACP-DE [39] and SaDE [43]. For the afore-mentioned DEs, since the local search schedule used in [43] can clearly improve their performances, the improved versions of the three DEs with local search, instead of their corresponding original versions, are used in this study and denoted as L-DE, L-SACP-DE and L-SaDE, respectively.

Moreover, a canonical genetic algorithm (CGA) and an adaptive genetic algorithm (AGA) introduced in [55] are implemented for comparison with SOA. The *fmincon*-based nonlinear programming method (NLP) [45, 56] is also considered.

All the algorithms are implemented in Matlab 7.0 and run on a PC with Pentium 4 CPU 2.4G 512MB RAM. For all the evolutionary methods in the experiments, the same population size *popsize*=60 except SPSO-2007 whose popsize is automatically computed by the algorithm, total 30 runs and the maximum generations of 300 are made. The NLP method uses a different uniformly random number in the search space as its start point in each run. The transformer taps and the reactive power compensation are discrete variables with the update step of 0.01p.u. and 0.048 p.u., respectively. The penalty factors $\lambda_V$ and $\lambda_Q$ in (13) are both set to 500.

The IEEE 57-bus system shown in Fig. 4 consists of 80 branches, 7 generator-buses and 15 branches under load tap setting transformer branches. The possible reactive power compensation buses are 18, 25 and 53. Seven buses are selected as $PV$-buses and $V\theta$-bus as follows: $PV$-buses: bus 2, 3, 6, 8, 9, 12; $V\theta$-bus: bus 1. The others are $PQ$-buses. The system data, variable limits and the initial values of control variables were given in [57]. In this case, the search space has 25 dimensions, i.e., the 7 generator voltages, 15 transformer taps, and 3 capacitor banks. The variable limits are given in Table 3.

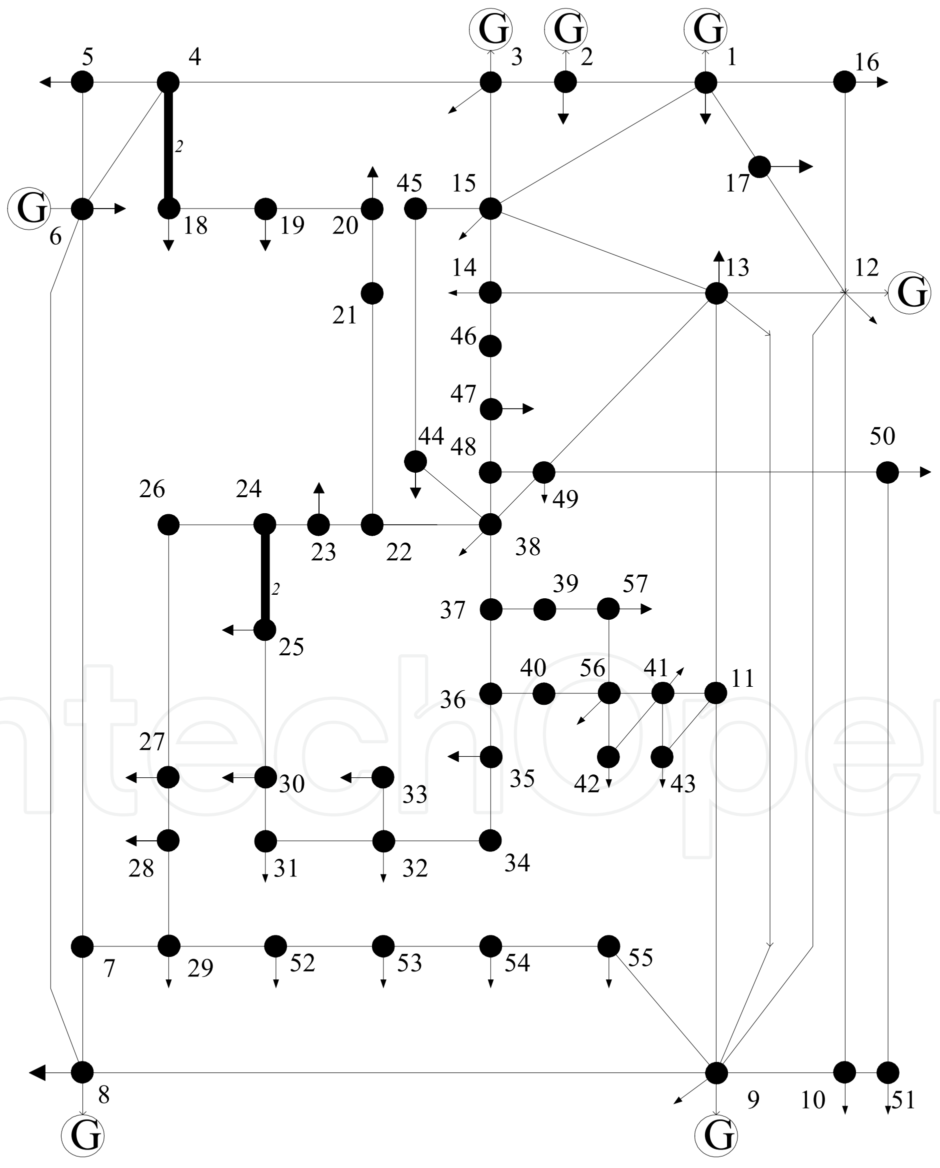


Fig. 4. Network configuration of IEEE 57-bus power system

| Limits of Generation Reactive Power | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bus | 1 | 2 | 3 | 6 | 8 | 9 | 12 |
| $Q_G^{\max}$ | 1.5 | 0.5 | 0.6 | 0.25 | 2.0 | 0.09 | 1.55 |
| $Q_G^{\min}$ | -0.2 | -0.17 | -0.1 | -0.08 | -1.4 | -0.03 | -1.5 |
| Limits of Voltage and Tap Setting | | | | | | | |
| $V_G^{\max}$ | $V_G^{\min}$ | $V_{PQ}^{\max}$ | $V_{PQ}^{\min}$ | $T_k^{\max}$ | $T_k^{\min}$ | | |
| 1.06 | 0.94 | 1.06 | 0.94 | 1.1 | 0.9 | | |
| Limits of Reactive Power Sources | | | | | | | |
| Bus | 18 | 25 | 53 | | | | |
| $Q_C^{\max}$ | 0.1 | 0.059 | 0.063 | | | | |
| $Q_C^{\min}$ | 0.0 | 0.0 | 0.0 | | | | |

Table 3. The Variable Limits (p.u.)

The system loads are given as follows:

$$P_{\text{load}}=12.508 \text{ p.u.}, Q_{\text{load}} =3.364 \text{ p.u.}$$

The initial total generations and power losses are as follows:

$$\sum P_G=12.7926 \text{ p.u.}, \sum Q_G=3.4545 \text{ p.u.},$$

$$P_{\text{loss}}=0.28462 \text{ p.u.}, Q_{\text{loss}}= -1.2427 \text{ p.u.}$$

There are five bus voltages outside the limits in the network: $V_{25}$=0.938, $V_{30}$=0.920, $V_{31}$=0.900, $V_{32}$=0.926, $V_{33}$= 0.924.

To compare the proposed method with other algorithms, the concerned performance indexes including the best active power losses (*Best*), the worst active power losses (*Worst*), the mean active power losses (*Mean*) and the standard deviation (*Std*) are summarized in Table 4 over total 30 runs. In order to determine whether the results obtained by SOA are statistically different from the results generated by other algorithms, the *T*-tests are conducted, and the corresponding *h* and *CI* values are presented in Table 4, too. Table 4 indicates that SOA has the smallest *Best*, *Mean* and *Std.* values than all the listed other algorithms, all the *h* values are equal to one, and all the confidence intervals are less than zero and don't contain zero. Hence, the conclusion can be drawn that SOA is significantly better and statistically more robust than all the other listed algorithms in terms of global search capacity and local search precision.

The best reactive power dispatch solutions from 30 runs for various algorithms are tabulated in Table 5 and Table 6. The $P_{\text{SAVE}}$% in Table 6 denotes the saving percent of the reactive power losses. Table 6 demonstrates that a power loss reduction of 14.7443% (from 0.28462 p.u. to 0.2426548 p.u.) is accomplished using the SOA approach, which is the biggest reduction of power loss than that obtained by the other approaches. The corresponding bus voltages are illustrated in Fig. 5 - Fig.8 for various methods. From Fig. 8, it can be seen that all the bus voltages optimized by SOA are kept within the limits, which implies that the proposed approach has better performance in simultaneously achieving the two goals of voltage quality improvement and power loss reduction than the other approaches on the employed test system.

The convergence graphs of the optimized control variables by the SOA are depicted in Fig. 9 - Fig. 11 with respect to the number of generations. From these figures, it can be seen that, due to the good global search ability of the proposed method, the control variables have a serious vibration at the early search phase, and then converge to a steady state at the late search phase, namely, a near optimum solution found by the method.

In this experiment, the computing time at every function evaluation is recorded for various algorithms. The total time of each algorithm is summarized in Table 7. Furthermore, the average convergence curves with active power loss vs. computing time are depicted for all the algorithms in Fig. 12. From Table 7, it can be seen that the computing time of SOA is less than that of the other evolutionary algorithms except SPSO-07 because of its smaller population size. However, Fig. 12 shows that, compared with SPSO-07, SOA has faster convergence speed and, on the contrary, needs less time to achieve the power loss level of SPSO-07. At the same time, SOA has better convergence rate than CLPSO and three versions of DE. Although PSO-w and PSO-cf have faster convergence speed at the earlier search phase, the two versions of PSO rapidly get trapped in premature convergence or search stagnation with the bigger final power losses than that of SOA. Hence, from the simulation results, SOA is synthetically superior to the other algorithms in computation complexity and convergence rate.

| Algorithms | *Best* | *Worst* | *Mean* | *Std.* | *h* | *CI* |
|---|---|---|---|---|---|---|
| NLP | 0.2590231 | 0.3085436 | 0.2785842 | $1.1677\times10^{-2}$ | 1 | $[-4.4368\times10^{-2}, -3.4656\times10^{-2}]$ |
| CGA | 0.2524411 | 0.2750772 | 0.2629356 | $6.2951\times10^{-3}$ | 1 | $[-2.2203\times10^{-2}, -1.8253\times10^{-2}]$ |
| AGA | 0.2456484 | 0.2676169 | 0.2512784 | $6.0068\times10^{-3}$ | 1 | $[-1.0455\times10^{-2}, -6.6859\times10^{-3}]$ |
| PSO-w | 0.2427052 | 0.2615279 | 0.2472596 | $7.0143\times10^{-3}$ | 1 | $[-6.7111\times10^{-3}, -2.3926\times10^{-3}]$ |
| PSO-cf | 0.2428022 | 0.2603275 | 0.2469805 | $6.6294\times10^{-3}$ | 1 | $[-6.3135\times10^{-3}, -2.2319\times10^{-3}]$ |
| CLPSO | 0.2451520 | 0.2478083 | 0.2467307 | $9.3415\times10^{-4}$ | 1 | $[-4.3117\times10^{-3}, -3.7341\times10^{-3}]$ |
| SPSO-07 | 0.2443043 | 0.2545745 | 0.2475227 | $2.8330\times10^{-3}$ | 1 | $[-5.6874\times10^{-3}, -3.9425\times10^{-3}]$ |
| L-DE | 0.2781264 | 0.4190941 | 0.3317783 | $4.7072\times10^{-2}$ | 1 | $[-1.0356\times10^{-1}, -7.4581\times10^{-2}]$ |
| L-SACP-DE | 0.2791553 | 0.3697873 | 0.3103260 | $3.2232\times10^{-2}$ | 1 | $[-7.7540\times10^{-2}, -5.7697\times10^{-2}]$ |
| L-SaDE | 0.2426739 | 0.2439142 | 0.2431129 | $4.8156\times10^{-4}$ | 1 | $[-5.5584\times10^{-4}, -2.5452\times10^{-4}]$ |
| SOA | **0.2426548** | **0.2428046** | **0.2427078** | $\mathbf{4.2081\times10^{-5}}$ | - | - |

Table 4. Comparisons of the Results of Various Methods on IEEE 57-Bus System over 30 Runs (p.u.)

| Variable | Base Case | NLP | CGA | AGA | PSO-w | PSO-cf | CLPSO | SPSO-07 | L-DE | L-SACP-DE | L-SaDE | SOA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Generator Voltage $V_G$ | | | | | | | | | | | | |
| $V_{G1}$ | 1.04 | 1.06 | 0.9686 | 1.0276 | 1.06 | 1.06 | 1.0541 | 1.0596 | 1.0397 | 0.9884 | 1.0600 | 1.06 |
| $V_{G2}$ | 1.01 | 1.06 | 1.0493 | 1.0117 | 1.0578 | 1.0586 | 1.0529 | 1.0580 | 1.0463 | 1.0543 | 1.0574 | 1.0580 |
| $V_{G3}$ | 0.985 | 1.0538 | 1.0567 | 1.0335 | 1.04378 | 1.0464 | 1.0337 | 1.0488 | 1.0511 | 1.0278 | 1.0438 | 1.0437 |
| $V_{G6}$ | 0.98 | 1.06 | 0.9877 | 1.0010 | 1.0356 | 1.0415 | 1.0313 | 1.0362 | 1.0236 | 0.9672 | 1.0364 | 1.0352 |
| $V_{G8}$ | 1.005 | 1.06 | 1.0223 | 1.0517 | 1.0546 | 1.06 | 1.0496 | 1.06 | 1.0538 | 1.0552 | 1.0537 | 1.0548 |
| $V_{G9}$ | 0.98 | 1.06 | 0.9918 | 1.0518 | 1.0369 | 1.0423 | 1.0302 | 1.0433 | 0.94518 | 1.0245 | 1.0366 | 1.0369 |
| $V_{G12}$ | 1.015 | 1.060 | 1.0044 | 1.0570 | 1.0334 | 1.0371 | 1.0342 | 1.0356 | 0.99078 | 1.0098 | 1.0323 | 1.0336 |
| Transformer Tap Ratio | | | | | | | | | | | | |
| $T_{4\text{-}18}$ | 0.97 | 0.91 | 0.92 | 1.03 | 0.9 | 0.98 | 0.99 | 0.95 | 1.02 | 1.05 | 0.94 | 1 |
| $T_{4\text{-}18}$ | 0.978 | 1.06 | 0.92 | 1.02 | 1.02 | 0.98 | 0.98 | 0.99 | 0.91 | 1.05 | 1 | 0.96 |
| $T_{21\text{-}20}$ | 1.043 | 0.93 | 0.97 | 1.06 | 1.01 | 1.01 | 0.99 | 0.99 | 0.97 | 0.95 | 1.01 | 1.01 |
| $T_{24\text{-}26}$ | 1.043 | 1.08 | 0.9 | 0.99 | 1.01 | 1.01 | 1.01 | 1.02 | 0.91 | 0.98 | 1.01 | 1.01 |
| $T_{7\text{-}29}$ | 0.967 | 1 | 0.91 | 1.1 | 0.97 | 0.98 | 0.99 | 0.97 | 0.96 | 0.97 | 0.97 | 0.97 |
| $T_{34\text{-}32}$ | 0.975 | 1.09 | 1.1 | 0.98 | 0.97 | 0.97 | 0.93 | 0.96 | 0.99 | 1.09 | 0.97 | 0.97 |
| $T_{11\text{-}41}$ | 0.955 | 0.92 | 0.94 | 1.01 | 0.9 | 0.9 | 0.91 | 0.92 | 0.98 | 0.92 | 0.9 | 0.9 |
| $T_{15\text{-}45}$ | 0.955 | 0.91 | 0.95 | 1.08 | 0.97 | 0.97 | 0.97 | 0.96 | 0.96 | 0.91 | 0.97 | 0.97 |
| $T_{14\text{-}46}$ | 0.9 | 0.98 | 1.03 | 0.94 | 0.95 | 0.96 | 0.95 | 0.95 | 1.05 | 1.08 | 0.96 | 0.95 |
| $T_{10\text{-}51}$ | 0.93 | 0.98 | 1.09 | 0.95 | 0.96 | 0.97 | 0.98 | 0.97 | 1.07 | 0.99 | 0.96 | 0.96 |
| $T_{13\text{-}49}$ | 0.895 | 0.98 | 0.9 | 1.05 | 0.92 | 0.93 | 0.95 | 0.92 | 0.99 | 0.91 | 0.92 | 0.92 |
| $T_{11\text{-}43}$ | 0.958 | 0.98 | 0.9 | 0.95 | 0.96 | 0.97 | 0.95 | 1 | 1.06 | 0.94 | 0.96 | 0.96 |
| $T_{40\text{-}56}$ | 0.958 | 0.98 | 1 | 1.01 | 1 | 0.99 | 1 | 1 | 0.99 | 0.99 | 1 | 1 |
| $T_{39\text{-}57}$ | 0.98 | 1.08 | 0.96 | 0.94 | 0.96 | 0.96 | 0.96 | 0.95 | 0.97 | 0.96 | 0.96 | 0.96 |
| $T_{9\text{-}55}$ | 0.94 | 1.03 | 1 | 1 | 0.97 | 0.98 | 0.97 | 0.98 | 1.07 | 1.1 | 0.97 | 0.97 |
| Capacitor Banks | | | | | | | | | | | | |
| $Q_{C18}$ | 0 | 0.08352 | 0.084 | 0.0168 | 0.05136 | 0.09984 | 0.09888 | 0.03936 | 0 | 0 | 0.08112 | 0.09984 |
| $Q_{C25}$ | 0 | 0.00864 | 0.00816 | 0.01536 | 0.05904 | 0.05904 | 0.05424 | 0.05664 | 0 | 0 | 0.05808 | 0.05904 |
| $Q_{C53}$ | 0 | 0.01104 | 0.05376 | 0.03888 | 0.06288 | 0.06288 | 0.06288 | 0.03552 | 0 | 0 | 0.06192 | 0.06288 |
| $P_{\text{loss}}$ | 0.28462 | .2590231 | .2524411 | .2456484 | 0.2427052 | 0.2428022 | 0.2451520 | 0.2443043 | 0.2781264 | 0.2791553 | 0.2426739 | 0.2426548 |

Table 5. Values of Control Variable & $P_{\text{loss}}$ After Optimization by Various Methods for IEEE 57-Bus Sytem (p.u.)

| Algorithms | $\sum P_G$ | $\sum Q_G$ | $P_{loss}$ | $Q_{loss}$ | $P_{SAVE}$% |
|---|---|---|---|---|---|
| NLP | 12.7687 | 3.1578 | 0.2590231 | -1.1532 | 8. 9934 |
| CGA | 12.7604 | 3.0912 | 0.2524411 | -1.1176 | 11.3059 |
| AGA | 12.7536 | 3.0440 | 0.2456484 | -1.1076 | 13.6925 |
| PSO-w | 12.7507 | 3.0300 | 0.2427052 | -1.0950 | 14.7266 |
| PSO-cf | 12.7508 | 2.9501 | 0.2428022 | -1.0753 | 14.6925 |
| CLPSO | 12.7531 | 3.0425 | 0.2451520 | -1.0853 | 13.8669 |
| SPSO-07 | 12.7523 | 3.0611 | 0.2443043 | -1.0845 | 14.1647 |
| L-DE | 12.7861 | 3.3871 | 0.2781264 | -1.2158 | 2.28150 |
| L-SACP-DE | 12.7871 | 3.2712 | 0.2791553 | -1.2042 | 1.92000 |
| L-SaDE | 12.7507 | 2.9855 | 0.2426739 | -1.0758 | 14.7376 |
| SOA | 12.7507 | 2.9684 | **0.2426548** | -1.0756 | **14.7443** |

Table 6. The Best Solutions for All the Methods on IEEE 57-Bus System (p.u.)

| Algorithms | Shortest time (s) | Longest time (s) | Average time (s) |
|---|---|---|---|
| CGA | 353.08 | 487.14 | 411.38 |
| AGA | 367.31 | 471.86 | 449.28 |
| PSO-w | 406.42 | 411.66 | 408.48 |
| PSO-cf | 404.63 | 410.36 | 408.19 |
| CLPSO | 423.30 | 441.98 | 426.85 |
| SPSO-07 | 121.98 | 166.23 | 137.35 |
| L-DE | 426.97 | 443.22 | 431.41 |
| L-SACP-DE | 427.23 | 431.16 | 428.98 |
| L-SaDE | 408.97 | 413.03 | 410.14 |
| SOA | 382.23 | 411.02 | 391.32 |

Table 7. The Average Computing Time for Various Algorithms

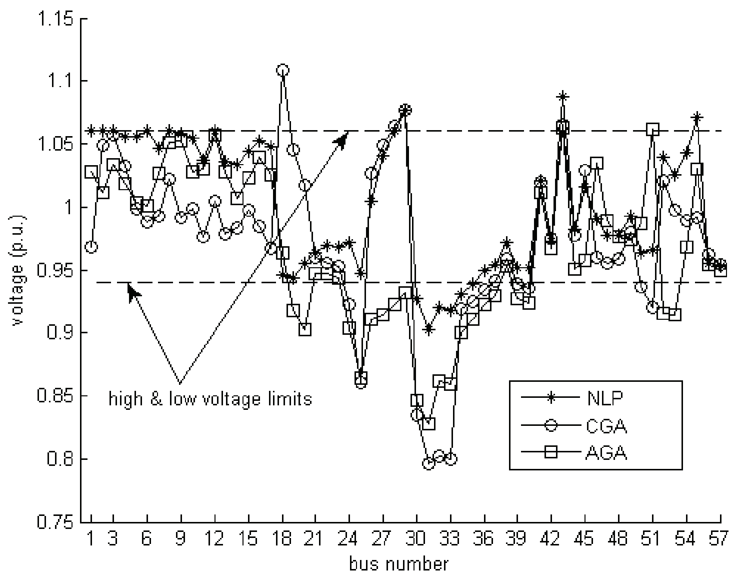


Fig. 5. Bus voltage profiles for NLP and GAs on IEEE 57-bus system

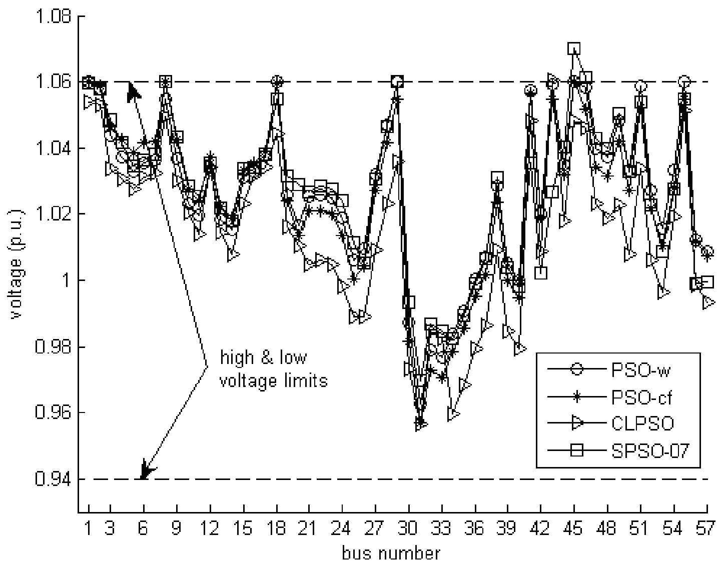


Fig. 6. Bus voltage profiles for PSOs on IEEE 57-bus system

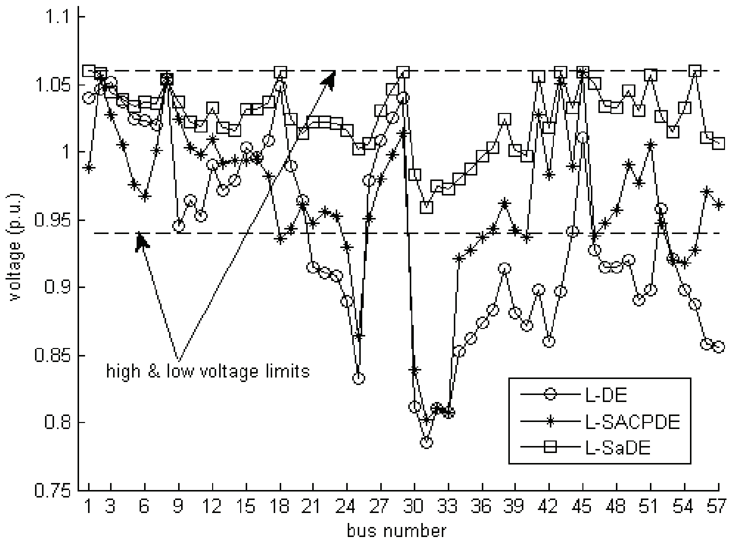


Fig. 7. Bus voltage profiles for DEs on IEEE 57-bus system

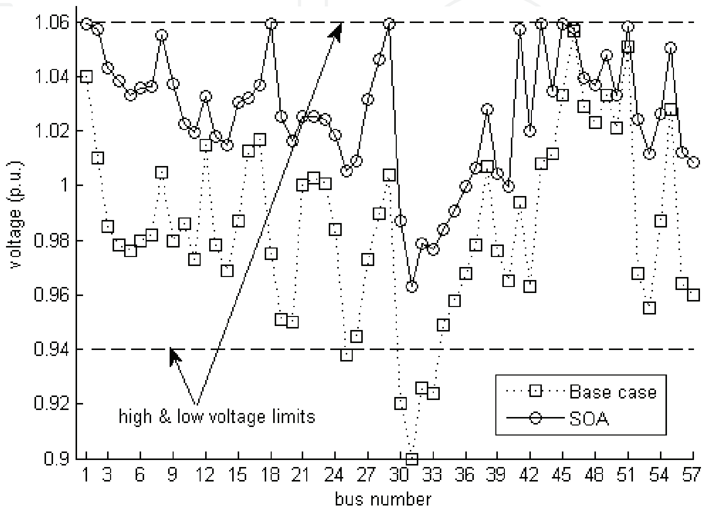
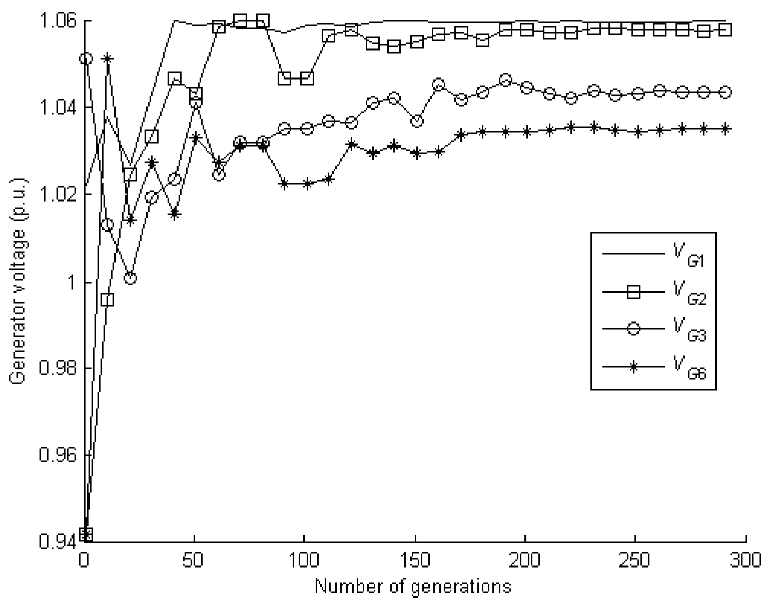
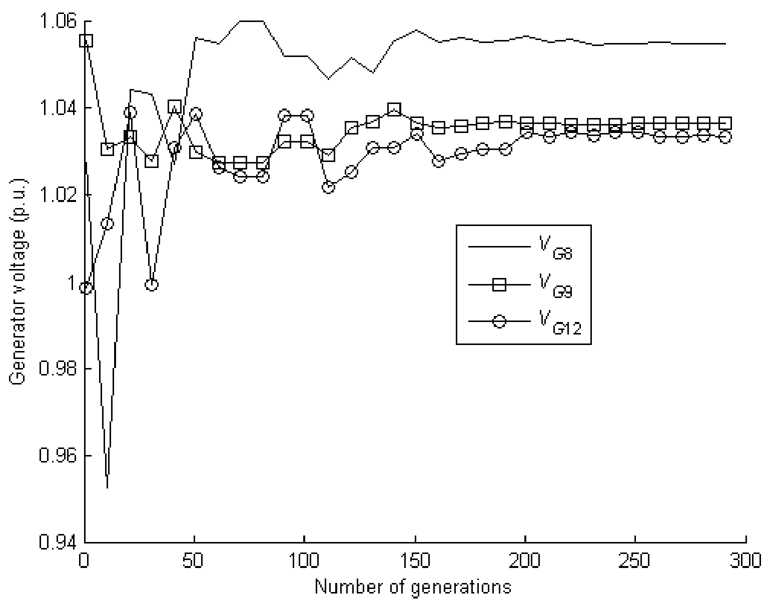


Fig. 8. Bus voltage profiles before and after optimization for SOA on IEEE 57-bus system

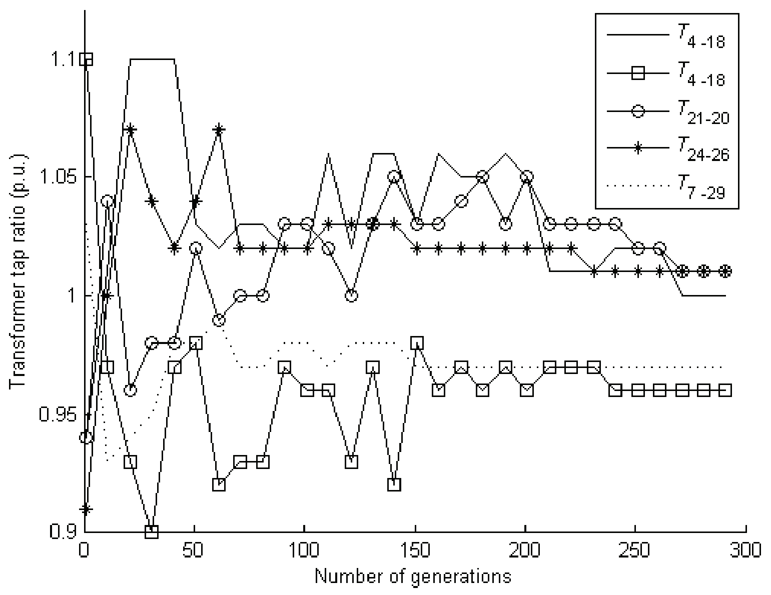


(a)



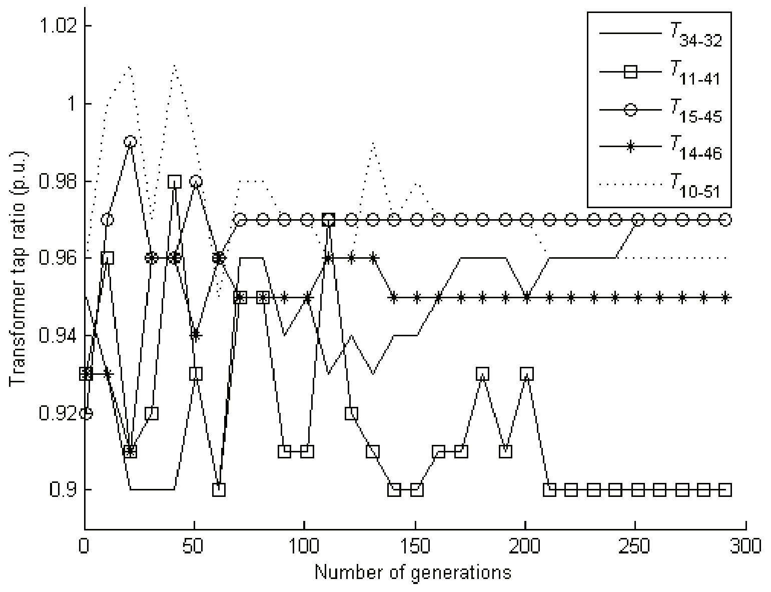
(b)

Fig. 9. Convergence of generator voltages $V_G$ for IEEE 57-bus system



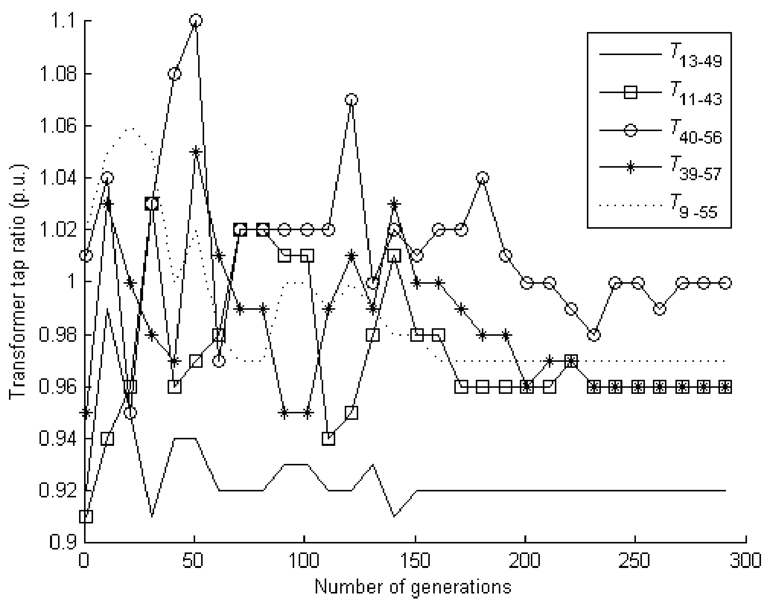
Transformer tap ratio (p.u.)
Number of generations
$T_{4\text{-}18}$
$T_{4\text{-}18}$
$T_{21\text{-}20}$
$T_{24\text{-}26}$
$T_{7\text{-}29}$

(a)



Transformer tap ratio (p.u.)
Number of generations
$T_{34\text{-}32}$
$T_{11\text{-}41}$
$T_{15\text{-}45}$
$T_{14\text{-}46}$
$T_{10\text{-}51}$

(b)



(c)

Fig. 10. Convergence of transformer taps *T* for IEEE 57-bus system

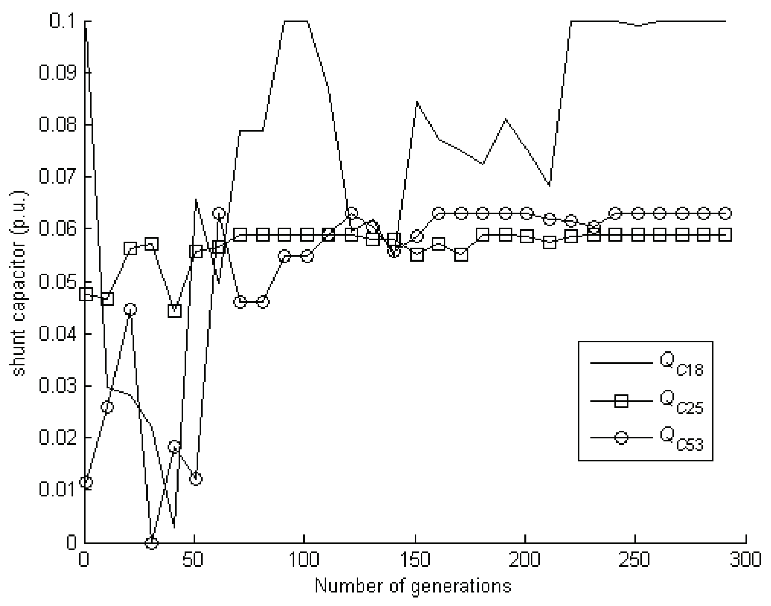


Fig. 11. Convergence of shunt capacitor $Q_C$ for IEEE 57-bus system

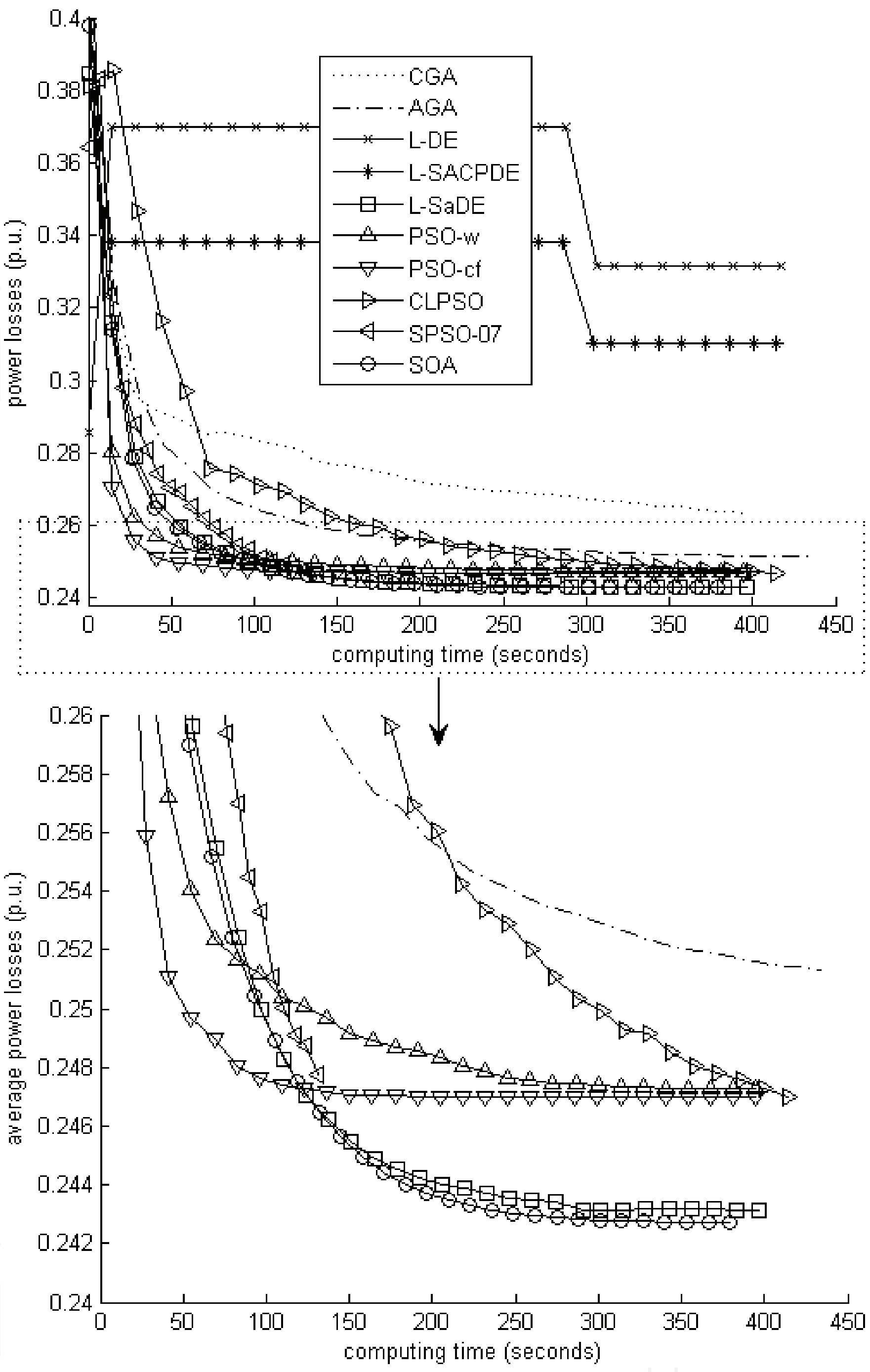


Fig. 12. Convergence graphs of various algorithms on IEEE 57-bus system (power loss vs. time)

## 2.4 SOA for multi-objective reactive power dispatch

### 2.4.1 Problem formulation

The multi-objective functions of the ORPD include the technical and economic goals. The economic goal is mainly to minimize the active power transmission loss. The technical goals are to minimize the load bus voltage deviation from the ideal voltage and to improve the voltage stability margin (VSM) [58]. Hence, the objectives of the ORPD model in this chapter are active power loss ($P_{\text{loss}}$), voltage deviation ($\Delta V_L$) and voltage stability margin ($VSM$).

*A. The Active Power Loss*

The active power loss minimization in the transmission network can be defined as follows [16, 17, 44]:

$$\min P_{\text{loss}} = f(\vec{x}_1, \vec{x}_2) = \sum_{k \in N_E} g_k (V_i^2 + V_j^2 - 2V_i V_j \cos\theta_{ij}) \tag{16}$$

Subject to

$$\begin{cases} P_{Gi} - P_{Di} = V_i \sum\limits_{j \in N_i} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) & i \in N_0 \\ Q_{Gi} - Q_{Di} = V_i \sum\limits_{j \in N_i} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) & i \in N_{PQ} \\ V_i^{\min} \le V_i \le V_i^{\max} & i \in N_B \\ T_k^{\min} \le T_k \le T_k^{\max} & k \in N_T \\ Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max} & i \in N_G \\ Q_{Ci}^{\min} \le Q_{Ci} \le Q_{Ci}^{\max} & i \in N_C \\ S_l \le S_l^{\max} & l \in N_l \end{cases} \tag{17}$$

where $f(\vec{x}_1, \vec{x}_2)$ denotes the active power loss function of the transmission network, $\vec{x}_1$ is the control variable vector $[V_G \ K_T \ Q_C]^T$, $\vec{x}_2$ is the dependent variable vector $[V_L \ Q_G]^T$, $V_G$ is the generator voltage (continuous), $T_k$ is the transformer tap (integer), $Q_C$ is the shunt capacitor/inductor (integer), $V_L$ is the load-bus voltage, $Q_G$ is the generator reactive power, $k$=($i$,$j$), $i \in N_B$, $j \in N_i$, $g_k$ is the conductance of branch $k$, $\theta_{ij}$ is the voltage angle difference between bus $i$ and $j$, $P_{Gi}$ is the injected active power at bus $i$, $P_{Di}$ is the demanded active power at bus $i$, $V_i$ is the voltage at bus $i$, $G_{ij}$ is the transfer conductance between bus $i$ and $j$, $B_{ij}$ is the transfer susceptance between bus $i$ and $j$, $Q_{Gi}$ is the injected reactive power at bus $i$, $Q_{Di}$ is the demanded reactive power at bus , $N_E$ is the set of numbers of network branches, $N_{PQ}$ is the set of numbers of PQ buses, $N_B$ is the set of numbers of total buses, $N_i$ is the set of numbers of buses adjacent to bus $i$ (including bus $i$), $N_0$ is the set of numbers of total buses excluding slack bus, $N_C$ is the set of numbers of possible reactive power source installation buses, $N_G$ is the set of numbers of generator buses, $N_T$ is the set of numbers of transformer branches, $S_l$ is the power flow in branch $l$, the superscripts "min" and "max" in equation (17) denote the corresponding lower and upper limits, respectively.

*B. Voltage Deviation*

Treating the bus voltage limits as constraints in ORPD often results in all the voltages toward their maximum limits after optimization, which means the power system lacks the required reserves to provide reactive power during contingencies. One of the effective ways to avoid this situation is to choose the deviation of voltage from the desired value as an objective function [59], i.e.:

$$\min \ \Delta V_L = \sum_{i=1}^{N_L} \left| V_i - V_i^* \right| / N_L \tag{18}$$

where $\Delta V_L$ is the per unit average voltage deviation, $N_L$ is the total number of the system load buses, $V_i$ and $V_i^*$ are the actual voltage magnitude and the desired voltage magnitude at bus $i$.

*C. Voltage Stability Margin*

Voltage stability problem has a closely relationship with the reactive power of the system, and the voltage stability margin is inevitably affected in optimal reactive power flow (ORPF) [58]. Hence, the maximal voltage stability margin should be one of the objectives in ORPF [49, 58, 59]. In the literature, the minimal eigenvalue of the non-singular power flow Jacobian matrix has been used by many researchers to improve the voltage stability margin [58]. Here, it is also employed [58]:

$$\max\ VSM = \max(\min\ |\text{eig}(Jacobi)|) \tag{19}$$

where *Jacobi* is the power flow Jacobian matrix, eig(*Jacobi*) returns all the eigenvalues of the Jacobian matrix, min(eig(*Jacobi*)) is the minimum value of eig(*Jacobi*), max(min(eig(*Jacobi*))) is to maximize the minimal eigenvalue in the Jacobian matrix.

*D. Multi-objective Conversion*

Considering different sub-objective functions have different ranges of function values, every sub-objective uses a transform to keep itself within [0,1]. The first two sub-objective functions, i.e., active power loss and voltage deviation, are normalized:

$$f_1 = \begin{cases} 0 & \text{if } P_{\text{loss}} < P_{\text{loss}_{\min}} \\ \dfrac{P_{\text{loss}} - P_{\text{loss}_{\min}}}{P_{\text{loss}_{\max}} - P_{\text{loss}_{\min}}} & \text{if } P_{\text{loss}_{\min}} \le P_{\text{loss}} \le P_{\text{loss}_{\max}} \\ 1 & \text{if } P_{\text{loss}} > P_{\text{loss}_{\max}} \end{cases} \tag{20}$$

$$f_2 = \begin{cases} 0 & \text{if } \Delta V_L < \Delta V_{L_{\min}} \\ \dfrac{\Delta V_L - \Delta V_{L_{\min}}}{\Delta V_{L_{\max}} - \Delta V_{L_{\min}}} & \text{if } \Delta V_{L_{\min}} \le \Delta V_L \le \Delta V_{L_{\max}} \\ 1 & \text{if } \Delta V_L > \Delta V_{L_{\max}} \end{cases} \tag{21}$$

where the subscripts "min" and "max" in equations (20) and (21) denote the corresponding expectant minimum and possible maximum value, respectively.
Since voltage stability margin sub-objective function is a maximization optimization problem, it is normalized and transformed into a minimization problem using the following equation:

$$f_3 = \begin{cases} 0 & \text{if } VSM > VSM_{\max} \\ \dfrac{VSM_{\max} - VSM}{VSM_{\max} - VSM_{\min}} & \text{else} \end{cases} \tag{22}$$

where the subscripts "min" and "max" in equation (22) denote the possible minimum and expectant maximum value, respectively.

Control variables are self-constrained, and dependent variables are constrained using penalty terms. Then, the overall objective function is generalized as follows:

$$\min\ f = \omega_1 f_1 + \omega_2 f_2 + \omega_3 f_3 + \lambda_V \sum_{N_V^{\lim}} \Delta V_L^2 + \lambda_Q \sum_{N_Q^{\lim}} \Delta Q_G^2 \tag{23}$$

where $\omega_i$ ($i$=1,2,3) is the user-defined constants which are used to weigh the contributions from different sub-objectives; $\lambda_V$, $\lambda_Q$ are the penalty factors; $N_V^{\lim}$ is the set of numbers of load-buses on which voltage outside limits, $N_Q^{\lim}$ is the set of numbers of generator buses on which injected reactive power outside limits; $\Delta V_L$ and $\Delta Q_G$ are defined as:

$$\Delta V_L = \begin{cases} V_L^{\min} - V_L & \text{if } V_L < V_L^{\min} \\ V_L - V_L^{\max} & \text{if } V_L > V_L^{\max} \end{cases} \tag{24}$$

$$\Delta Q_G = \begin{cases} Q_G^{\min} - Q_G & \text{if } Q_G < Q_G^{\min} \\ Q_G - Q_G^{\max} & \text{if } Q_G > Q_G^{\max} \end{cases} \tag{25}$$

### 2.4.2 Implementation of SOA for reactive power optimization

The fitness value of each seeker is calculated by using the objective function in (23). The real-value position of the seeker consists of three parts: generator voltages, transformer taps and shunt capacitors/inductors. According to the section 3.4 of this paper, after the update of the position, the main program is turned to the sub-program for evaluating the objective function where the latter two parts of the position are truncated into the corresponding integers as [44, 55]. Then, the real-value position is changed into a mixed-variable vector which is used to calculate the objective function value by equation (23) based on Newton-Raphson power flow analysis [45]. The reactive power optimization based on SOA can be described as follows [17].

**Step 1.** Read the parameters of power system and the proposed algorithm, and specify the lower and upper limits of each variable.

**Step 2.** Initialize the positions of the seekers in the search space randomly and uniformly. Set the time step $t$=0.

**Step 3.** Calculate the fitness values of the initial positions using the objective function in (23) based on the results of Newton-Raphson power flow analysis [45]. The initial historical best position among the population is achieved. Set the historical best position of each seeker to his current position.

**Step 4.** Let $t$=$t$+1.

**Step 5.** Determine the neighbors, search direction and step length for each seeker.

**Step 6.** Update the position of each seeker.

**Step 7.** Calculate the fitness values of the new positions using the objective function based on the Newton-Raphson power flow analysis results. Update the historical best position among the population and the historical best position of each seeker.

**Step 8.** Go to Step 4 until a stopping criterion is satisfied.

### 2.4.3 Simulation results

To evaluate the effectiveness and efficiency of the proposed SOA-based reactive power optimization approach, the standard IEEE 57-bus power system is used as the test system.
For the comparisons, the following algorithms are also considered: PSO-w (learning rate $c_1$ = $c_2$=2, inertia weight linearly decreased from 0.9 to 0.4 with run time increasing, the maximum velocity $v_{\max}$ is set at 20% of the dynamic range of the variable on each dimension) [40], PSO-cf ($c_1$= $c_2$=2.01 and constriction factor $\chi$=0.729844) [41], CLPSO (its parameters follow the suggestions from [42] except that the refreshing gap $m$=2) and SPSO-07 [54], the original DE (DE: DE/rand/1/bin, $F$=0.5, $CR$=0.9) [39]), SACP-DE and SaDE. For the afore-mentioned DEs, since the local search schedule used in [43] can clearly improve their performances, the improved versions of the three DEs with local search, instead of their corresponding original versions, are used in this study and denoted as L-DE, L-SACP-DE and L-SaDE, respectively.
Moreover, a canonical genetic algorithm (CGA) and an adaptive genetic algorithm (AGA) introduced in [55] are considered for comparison with SOA.
All the algorithms are implemented in Matlab 7.0 and run on a PC with Pentium 4 CPU 2.4G 512MB RAM. In the experiments, the same population size *popsize*=60 for the IEEE 57-bus system except SPSO-2007 whose popsize is automatically computed by the algorithm, total 30 runs and the maximum generations of 300 are made. The transformer taps and the reactive power compensation are discrete variables with the update step of 0.01p.u. and 0.048 p.u., respectively.
The main parameters involved in SOA include: the population size $s$, the number of subpopulations, and the parameters of membership function of Fuzzy reasoning (including the limits of membership degree value, i.e., $\mu_{\max}$ and $\mu_{\min}$ in (8) and the limits of $\omega$, i.e., $\omega_{\max}$ and $\omega_{\min}$ in (9)). In this paper, $s$=60 for IEEE 57-bus system and $s$=80 for IEEE 118-bus system, $K$=3, $\mu_{\max}$=0.95, $\mu_{\max}$=0.0111, $\omega_{\max}$=0.8, $\omega_{\min}$=0.2 for both the test systems.
The IEEE 57-bus system [45] shown in Fig. 4 consists of 80 branches, 7 generator-buses and 15 branches under load tap setting transformer branches. The possible reactive power compensation buses are 18, 25 and 53. Seven buses are selected as $PV$-buses and $V\theta$-bus as follows: $PV$-buses: bus 2, 3, 6, 8, 9, 12; $V\theta$-bus: bus 1. The others are $PQ$-buses. The system data, operating conditions, variable limits and the initial generator bus voltages and transformer taps were given in [57], or can be obtained from the authors of this paper on request. The model parameters in the equations (20)-(23) are set as: $P_{loss_{\max}} = 0.5, P_{loss_{\min}} = 0.2,\ \Delta V_{L_{\max}} = 1,\quad \Delta V_{L_{\min}} = 0$, $VSM_{\max}$=0.4, $VSM_{\min}$=0.05, $\omega_1$=0.6, $\omega_2$=0.2, $\omega_3$=0.2, $\lambda_V$=500 and $\lambda_Q$=500.
The system loads are : $P_{\text{load}}$=12.508 p.u., $Q_{\text{load}}$ =3.364 p.u. The initial total generations and power losses are: $\sum P_G$=12.7926 p.u., $\sum Q_G$=3.4545 p.u., $P_{\text{loss}}$=0.28462 p.u., $Q_{\text{loss}}$= -1.2427 p.u. There are five bus voltages outside the limits: $V_{25}$=0.938, $V_{30}$=0.920, $V_{31}$=0.900, $V_{32}$=0.926, $V_{33}$= 0.924.
To compare the proposed method with other algorithms, the concerned performance indexes including the *best*, *worst*, *mean* and standard deviation (*Std.*) of the overall and sub-objective function values are summarized in Tables 8 - 11. In order to determine whether the results obtained by SOA are statistically different from the results generated by other algorithms, the $T$-tests [56] are conducted. An $h$ value of one indicates that the performances of the two algorithms are statistically different with 95% certainty, whereas $h$ value of zero implies that the performances are not statistically different. The $CI$ is confidence interval.

The corresponding *h* and *CI* values for overall function values and active power losses are presented in Tables 8 and 9, respectively. The best reactive power dispatch solutions from 30 runs for various algorithms are tabulated in Table 12 where $P_{SAVE}$% denotes the saving percent of the reactive power losses. The corresponding bus voltages are illustrated in Fig. 13. The total time of each algorithm is summarized in Table 13. The average convergence curves for overall function value vs. computing time and active power loss vs. computing time are depicted for all the algorithms in Figs. 14 and 15, respectively.

Table 8 indicates that SOA has the smallest *Best*, *Mean*, *Worst* and *Std.* values of overall function than all the listed other algorithms except that SOA has the a little larger *Worst* value than that of PSO-w, only the *h* values for SOA vs. CLPSO and SOA vs. L-SaDE are equal to zeroes (Accordingly, their confidence intervals contain zero). Table 9 indicates that SOA has the smallest *Best*, *Mean*, *Worst* and *Std.* values of power loss than all the listed other algorithms except that SOA has the a little larger *Worst* value than that of L-SaDE with *h*=0 and *CI* containing zero. Tables 10 and 11 show that SOA has the better or comparable other two sub-objective values, i.e., voltage stability margin (*VSM*) and voltage deviation ($\Delta V_L$). Table 12 demonstrates that a power loss reduction of 13.4820% (from 0.28462 p.u. to 0.246248 p.u.) is accomplished using the SOA approach, which is the biggest reduction of power loss than that obtained by the other approaches. Hence, the conclusion can be drawn that SOA is better than, or comparable to, all the other listed algorithms in terms of global search capacity and local search precision. Furthermore, from Fig. 13, it can be seen that all the bus voltages optimized by SOA are acceptably kept within the limits.

From Table 13, it can be seen that the average computing time of SOA is less than that of other algorithms except SPSO-07 because of its smaller population size. However, Figs. 14 and 15 show that, compared with SPSO-07, SOA has faster convergence speed and, on the contrary, needs less time to achieve the overall function value and power loss level achieved by SPSO-07. At the same time, SOA also has better convergence rate than GAs, DEs and PSOs.

| Algorithms | *Best* | *Worst* | *Mean* | *Std.* | *h* | *CI* |
|---|---|---|---|---|---|---|
| CGA | 0.192750 | 0.195206 | 0.194024 | $4.8798\times10^{-4}$ | 1 | [-0.0684, -0.0378] |
| AGA | 0.192284 | 0.193994 | 0.193030 | $4.4517\times10^{-4}$ | 1 | [-0.0674, -0.0368] |
| PSO-w | 0.191851 | **0.191977** | 0.191901 | $\mathbf{4.2691\times10^{-5}}$ | 1 | [-0.0727, -0.0292,] |
| PSO-cf | 0.116954 | 0.192593 | 0.188312 | $16797\times10^{-2}$ | 1 | [-0.0634, -0.0314] |
| CLPSO | 0.120773 | 0.192739 | 0.148663 | $3.3476\times10^{-2}$ | 0 | [-0.0257, 0.0102] |
| SPSO-2007 | 0.191918 | 0.193559 | 0.192551 | $3.9668\times10^{-4}$ | 1 | [-0.0669, -0.0363,] |
| L-DE | 0.232519 | 0.388413 | 0.314205 | $4.0455\times10^{-2}$ | 1 | [-0.1923, -0.1543] |
| L-SACP-DE | 0.237277 | 0.395611 | 0.317571 | $4.1949\times10^{-2}$ | 1 | [-0.1959, -0.1574] |
| L-SaDE | 0.116819 | 0.192131 | 0.154692 | $3.8257\times10^{-2}$ | 0 | [-0.0324, 0.0049] |
| SOA | **0.116495** | 0.192083 | **0.140927** | $3.4163\times10^{-2}$ | - | - |

Table 8. The Results of Overall Objective Function Values for Various Algorithms on IEEE 57-bus System over 30 Runs (p.u.)

| Algorithms | *Best* | *Worst* | *Mean* | *Std.* | *h* | *CI* |
|---|---|---|---|---|---|---|
| CGA | 0.267170 | 0.419747 | 0.323181 | $4.2147\times10^{-2}$ | 1 | [-0.0787, -0.0529] |
| AGA | 0.258072 | 0.369785 | 0.296744 | $3.5776\times10^{-2}$ | 1 | [-0.0507, -0.0280,] |
| PSO-w | 0.259729 | 0.324923 | 0.283945 | $2.2313\times10^{-2}$ | 1 | [-0.0363 -0.0168] |
| PSO-cf | 0.247866 | 0.393221 | 0.297066 | $3.2551\times10^{-2}$ | 1 | [-0.0502, -0.0291,] |
| CLPSO | 0.257968 | 0.340029 | 0.273334 | $1.9252\times10^{-2}$ | 1 | [-0.0235, -0.0083,] |
| SPSO-2007 | 0.274210 | 0.386235 | 0.307093 | $2.7961\times10^{-2}$ | 1 | [-0.0591, -0.0402] |
| L-DE | 0.291864 | 0.5069975 | 0.373198 | $5.4894\times10^{-2}$ | 1 | [-0.1320, -0.0996] |
| L-SACP-DE | 0.273183 | 0.4438575 | 0.343407 | $4.5156\times10^{-2}$ | 1 | [-0.0997, -0.0723] |
| L-SaDE | 0.246712 | **0.282335** | 0.260983 | $1.3426\times10^{-2}$ | 0 | [-0.0101, 0.0030] |
| SOA | **0.246248** | 0.287541 | **0.257410** | **$1.1918\times10^{-2}$** | - | - |

Table 9. The Results of Active Power Loss for Various Algorithms on IEEE 57-bus System over 30 Runs (p.u.)

| *Algorithms* | *Best* | *Worst* | *Mean* | *Std.* |
|---|---|---|---|---|
| CGA | 0.186249 | 0.173969 | 0.1798794 | $2.4399\times10^{-3}$ |
| AGA | 0.188582 | 0.180030 | 0.1848524 | $2.2259\times10^{-3}$ |
| PSO-w | 0.190745 | 0.190117 | **0.1904974** | $2.1346\times10^{-4}$ |
| PSO-cf | **0.190754** | 0.1870317 | 0.1895324 | $122285\times10^{-3}$ |
| CLPSO | 0.187857 | 0.1783987 | 0.183922 | $3.0781\times10^{-3}$ |
| SPSO-2007 | 0.190411 | 0.182206 | 0.187245 | $1.9834\times10^{-3}$ |
| L-DE | 0.1778431 | 0.165211 | 0.171368 | $3.4560\times10^{-3}$ |
| L-SACP-DE | 0.183051 | 0.159702 | 0.170998 | $5.7523\times10^{-3}$ |
| L-SaDE | 0.190638 | 0.1853272 | 0.1882648 | $1.9748\times10^{-3}$ |
| SOA | 0.190709 | 0.176374 | 0.187451 | $2.6388\times10^{-3}$ |

Table 10. The Results of Voltage Stability Margin for Various Algorithms on IEEE 57-bus System over 30 Runs (p.u.)

| Algorithms | CGA | AGA | PSO-w | PSO-cf | CLPSO | SPSO-2007 | LDE | L-SACP-DE | L-SaDE | SOA |
|---|---|---|---|---|---|---|---|---|---|---|
| *Best* | 0 | 0 | 0 | 0 | 0 | 0 | 2.886554 | 2.317914 | 0 | 0 |
| *Worst* | 0 | 0 | 0 | 0 | 0.291757 | 0 | 0.561878 | 0.634840 | 0 | 0 |
| *Mean* | 0 | 0 | 0 | 0 | 0.014588 | 0 | 1.777176 | 1.890710 | 0 | 0 |
| *Std.* | 0 | 0 | 0 | 0 | $6.5239\times10^{-2}$ | 0 | $6.0402\times10^{-1}$ | $7.9319\times10^{-1}$ | 0 | 0 |

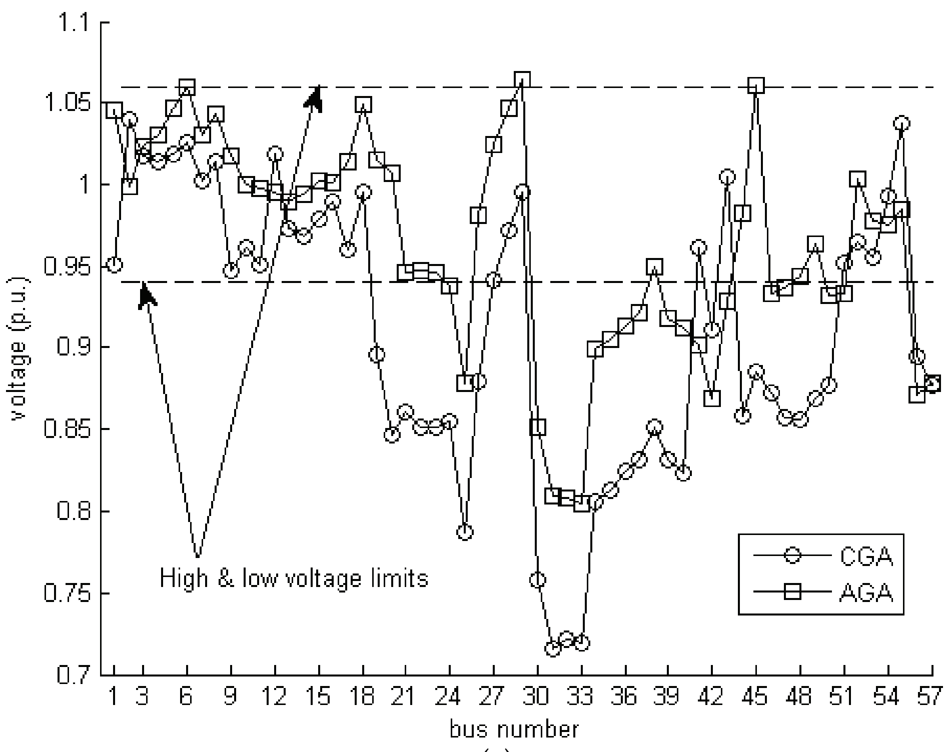
Table 11. The Results of Voltage Deviation for Various Algorithms on IEEE 57-bus System over 30 Runs (p.u.)

| Algorithms | $\sum P_G$ | $\sum Q_G$ | $P_{\text{loss}}$ | $Q_{\text{loss}}$ | $P_{\text{SAVE}}\%$ | *VSM* | $\Delta V_L$ |
|---|---|---|---|---|---|---|---|
| CGA | 12.7752 | 3.1744 | 0.267170 | -1.1565 | 6.1308 | 0.179828 | 0 |
| AGA | 12.7661 | 3.0679 | 0.258072 | -1.1326 | 9.3276 | 0.185845 | 0 |
| PSO-w | 12.7677 | 3.1026 | 0.259729 | -1.1598 | 8.7453 | **0.190117** | 0 |
| PSO-cf | 12.7559 | 3.0157 | 0.247866 | -1.1137 | 12.9132 | 0.1870317 | 0 |
| CLPSO | 12.7660 | 3.1501 | 0.257968 | -1.1295 | 9.3642 | 0.1849117 | 0.291757 |
| SPSO-2007 | 12.7822 | 3.1818 | 0.274210 | -1.2532 | 3.6576 | 0.1877947 | 0 |
| L-DE | 12.7999 | 3.3656 | 0.291864 | -1.2158 | -1.2380 | 0.1701207 | 2.886554 |
| L-SACP-DE | 12.7812 | 3.2085 | 0.273183 | -1.1868 | 4.0185 | 0.183051 | 4.282957 |
| L-SaDE | 12.7549 | 3.0191 | 0.246712 | -1.1209 | 13.2696 | 0.186182 | 0 |
| SOA | 12.7543 | 2.9837 | **0.246248** | -1.0914 | **13.4820** | 0.186895 | 0 |

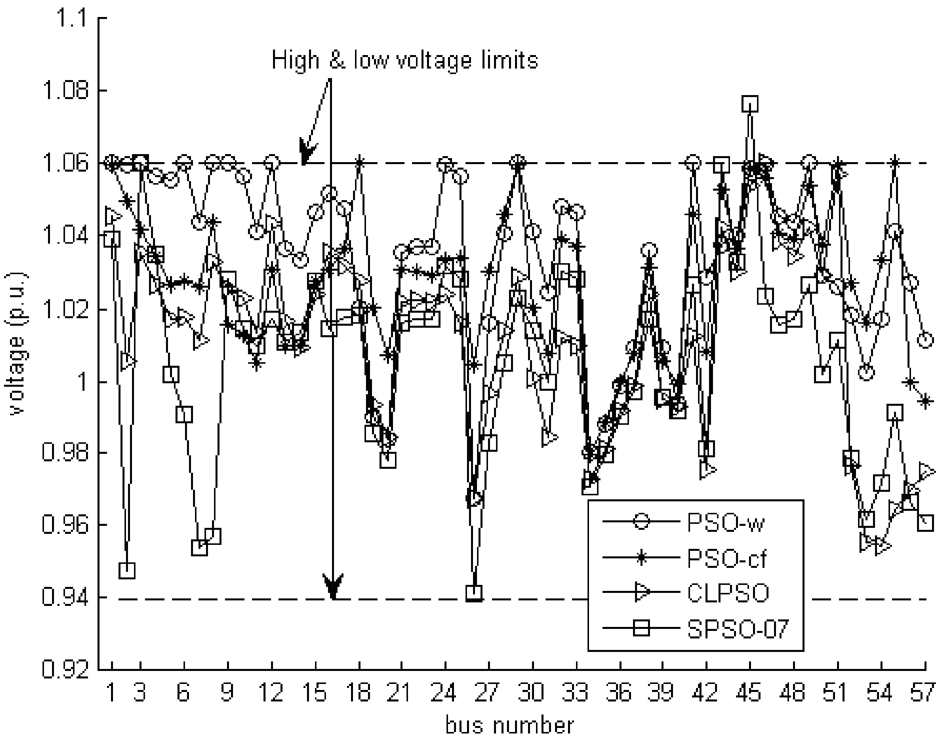
Table 12. The Best Dispatch Solutions for Various Algorithms on IEEE 57-bus System (p.u.)

| Algorithms | CGA | AGA | PSO-w | PSO-cf | CLPSO | SPSO-2007 | LDE | L-SACP-DE | L-SaDE | SOA |
|---|---|---|---|---|---|---|---|---|---|---|
| Shortest time (s) | 1265.34 | 1273.44 | 1216.91 | 1188.45 | 1399.48 | 433.36 | 1210.73 | 1212.95 | 1273.42 | 1192.83 |
| Longest time (s) | 1295.02 | 1323.91 | 1244.64 | 1268.00 | 1448.84 | 495.97 | 1239.86 | 1235.03 | 1368.03 | 1288.66 |
| Average time (s) | 1284.11 | 1293.78 | 1229.98 | 1225.14 | 1426.19 | 480.94 | 1224.27 | 1221.51 | 1306.86 | 1221.10 |

Table 13. The Computing Time for Various Algorithms on IEEE 57-bus System over 30 Runs

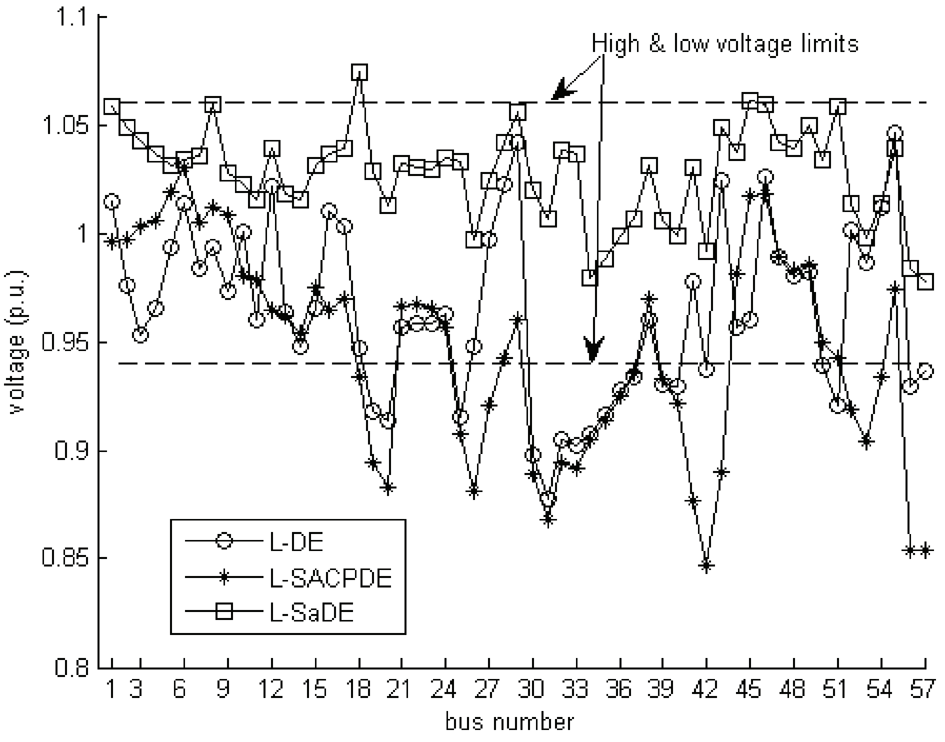


(a)



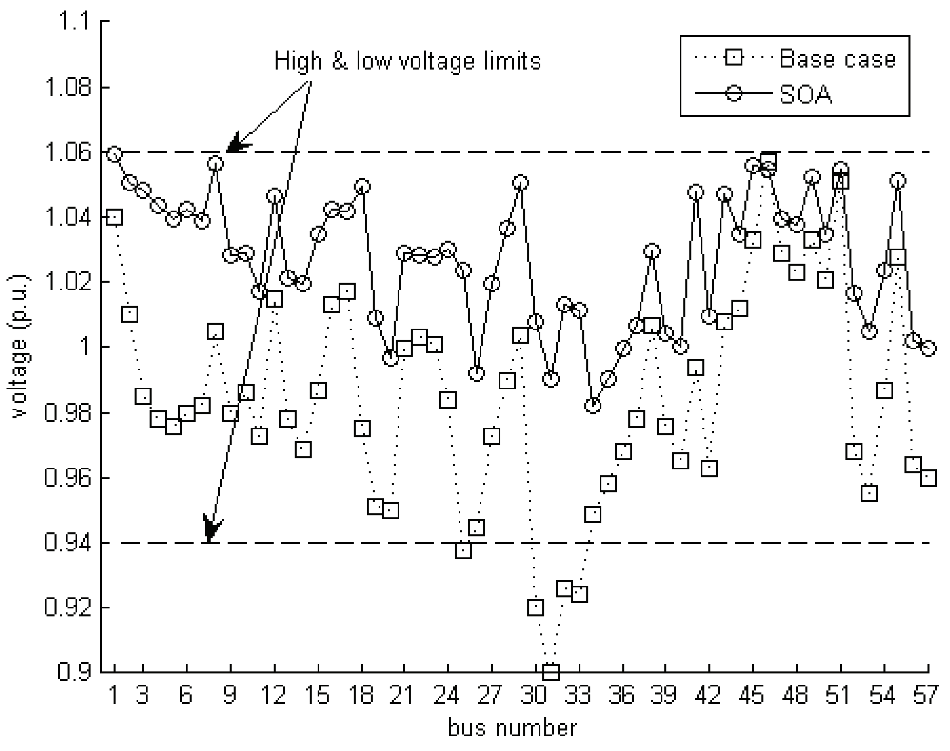
High & low voltage limits
voltage (p.u.)
bus number
PSO-w
PSO-cf
CLPSO
SPSO-07

(b)



High & low voltage limits
voltage (p.u.)
bus number
L-DE
L-SACPDE
L-SaDE

(c)



(d)

Fig. 13. Bus voltage profiles for various algorithms on IEEE 57-bus

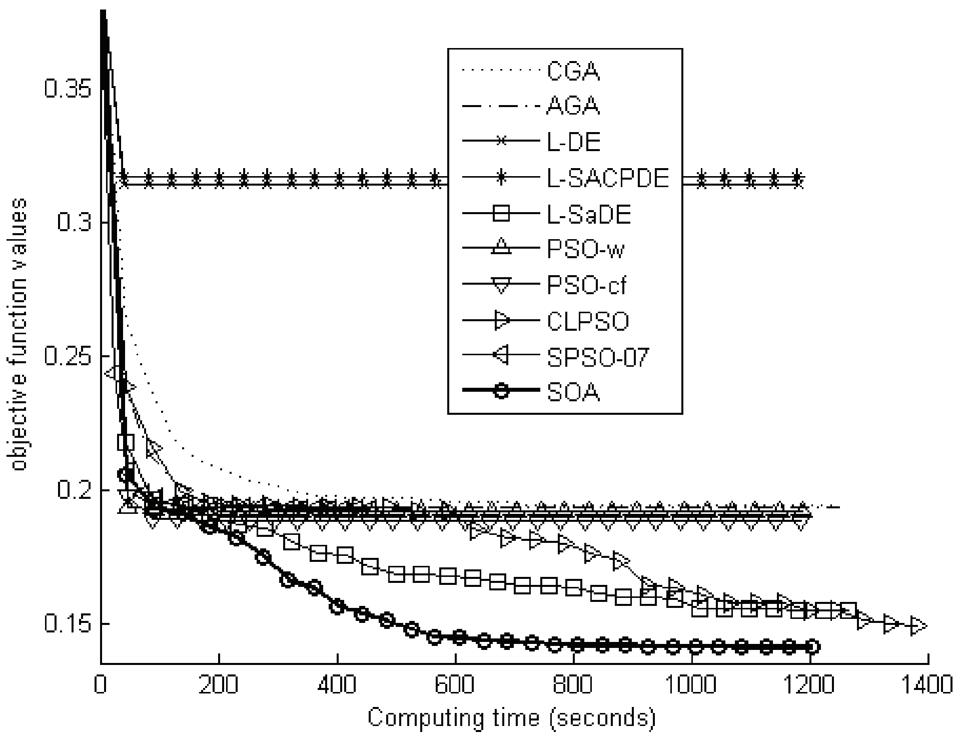


Fig. 14. Convergence Graphs of various algorithms on IEEE 57-bus (overall objective function values vs. time)

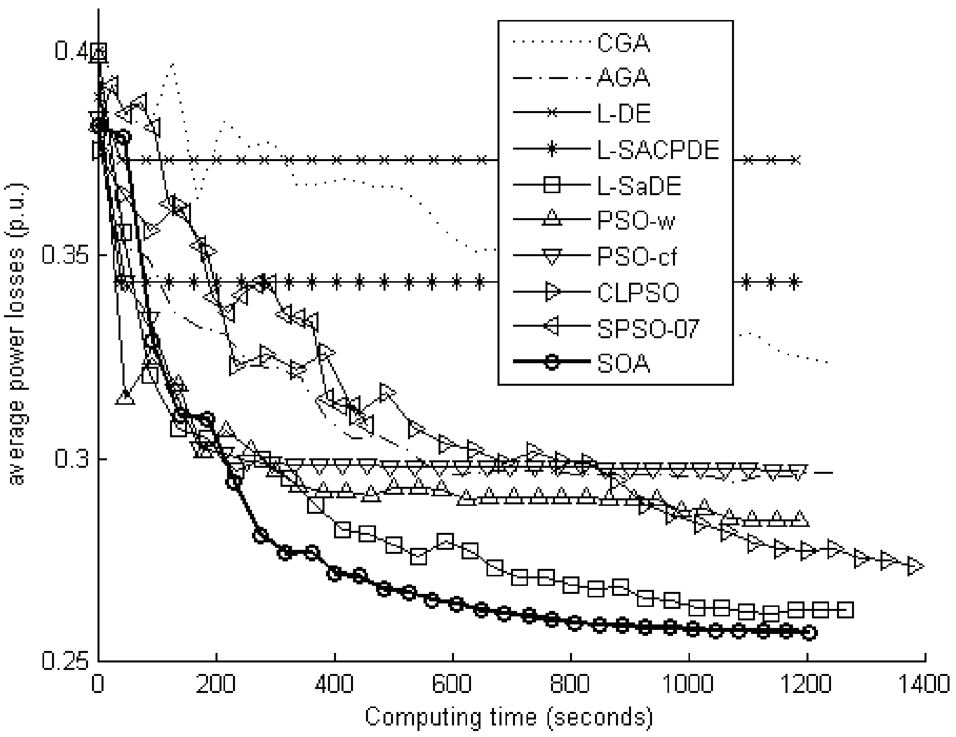


Fig. 15. Convergence Graphs of various algorithms on IEEE 57-bus (power loss vs. time)

## 3. Stochastic Focusing Search (SFS) and its application

### 3.1 Stochastic Focusing Search (SFS) (Ref.[20, 21])

Stochastic focusing search (SFS) is a simplified and improved version of PSO. In the SFS, particles make a focusing search around the best position so far and stochastically update their positions within a neighborhood of the best position with a decreasing search radius. Unlike PSO, the velocity and position iteration of the SFS is implemented according to the following equations:

$$\vec{v}_i(t)=\begin{cases} Rand()\times(R_{ti}-\vec{x}_i(t-1)) & \text{if } fun(\vec{x}_i(t-1)) \ \geq \ fun(\vec{x}_i(t-2)) \\ \vec{v}_i(t-1) & \text{if } fun(\vec{x}_i(t-1)) \ < \ fun(\vec{x}_i(t-2)) \end{cases} \tag{26}$$

$$\vec{x}_i(t)=\vec{v}_i(t)+\vec{x}_i(t-1) \tag{27}$$

where $Rand()$ returns a uniformly random number in the range [0, 1], $fun(\vec{x}_i(t))$ is the objective function value of $\vec{x}_i(t)$, $R_{ti}$ is a random selected point (position) in the neighborhood space $R_t$ of $\vec{g}_{best}$. $R_t$ is defined as: $\left[\vec{g}_{best}-\frac{w(\vec{g}_{best}-\vec{x}_{\min})}{(\vec{x}_{\max}-\vec{x}_{\min})^{1-w}}, \vec{g}_{best}+\frac{w(\vec{x}_{\max}-\vec{g}_{best})}{(\vec{x}_{\max}-\vec{x}_{\min})^{1-w}}\right]$, where $\vec{x}_{\max}$ and $\vec{x}_{\min}$ are the search space borders. When $w$ is linearly decreased from 1 to 0, $R_t$ is deflated from the entire search space to the best point $\vec{g}_{best}$.

According to Eq. (26), if a particle holds a good velocity at the time step *t*-1 (i.e., $fun(\vec{x}_i(t-1)) \ < \ fun(\vec{x}_i(t-2))$), its velocity keeps the same one as the past; else, the particle

randomly selects a position within a neighborhood of the best position so far. Moreover, the SFS also uses a greedy selection, namely: if the new position obtained by Eq. (27) is worse the early position (i.e., $fun(\vec{x}_i(t-1)) < fun(\vec{x}_i(t))$), the particle will come back to the early position (i.e., $\vec{x}_i(t) = \vec{x}_i(t-1)$).
According to Eqs. (26) and (27), it can be seen that each individual particle makes a search in a decreasing $R_t$ with time step increasing. It is of significance to select an appropriate $w$ to not only assure the global convergence ability but also avoid a local extremum. In this study, $w$ is defined as:

$$w = (\frac{G-t}{G})^{\delta} \tag{28}$$

where $G$ is the maximum generation, $\delta$ is a positive number. It is indicated that $w$ is decreased from 1 to 0 with the increasing of time step *t*.
To improve the global searching ability and avoid a local extremum, the particles are categorized into multiple subpopulations. The number of subpopulations $\mu$ is decreasing from particles size *s* to 1 according to the indexes of the particles with the inertia weight $w'$.

$$w' = (\frac{G-t}{G})^{\delta'} \tag{29}$$

$$\mu = \lfloor w's + 1 \rfloor \tag{30}$$

It can be seen that $w'$ has the same form of $w$ from equation (29). $w'$ decreases with the run time increasing so as to decrease the subpopulations $\mu$. In every subpopulation, there will be a different $\vec{g}_{best}$, which is the best position of the subpopulation. The pseudocode of the SFS is presented in Fig. 16.

```
begin
    $t$←0;
    generating $s$ positions uniformly and randomly in the whole search space;
    evaluating each particle;
    repeat
        $t$←$t$+1;
        finding the respective $\vec{g}_{best}$ in every subpopulation;
        updating and evaluating each particle's position using (3) and (4) with the
        greedy selection;
    until the stop condition is satisfied
end.
```

Fig. 16. The pseudo code of the main algorithm

### 3.2 SFS for benchmark function optimization (Ref.[20])

#### 3.2.1 The benchmark functions

In order to evaluate the novel algorithm, a test suite of benchmark functions previously introduced by Yao, Liu and Lin [60] was used (listed in Table 14), the ranges of their search spaces, their dimensionalities, and their global minimum function values (ideal values) are

| Functions | $n$ | $S$ | $f_{\min}$ |
|---|---|---|---|
| $f_1(\vec{x})=\sum_{i=1}^{n}x_i^2$ | 30 | $[-5.12,5.12]^n$ | $f_1(\vec{0})=0$ |
| $f_2(\vec{x})=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$ | 30 | $[-10,10]^n$ | $f_2(\vec{0})=0$ |
| $f_3(\vec{x})=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$ | 30 | $[-100,100]^n$ | $f_3(\vec{0})=0$ |
| $f_4(\vec{x})=\max_i\{\lvert x_i\rvert,1\le i\le n\}$ | 30 | $[-100,100]^n$ | $f_4(\vec{0})=0$ |
| $f_5(\vec{x})=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$ | 30 | $[-30,30]^n$ | $f_5(\vec{1})=0$ |
| $f_6(\vec{x})=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$ | 30 | $[-100,100]^n$ | $f_6(\vec{p})=0,\ -0.5\le p_i\le 0.5$ |
| $f_7(\vec{x})=\sum_{i=1}^{n}ix_i^4+rand[0,1)$ | 30 | $[-1.28,1.28]^n$ | $f_7(\vec{0})=0$ |
| $f_8(\vec{x})=-\sum_{i=1}^{n}(x_i\sin(\sqrt{\lvert x_i\rvert}))$ | 30 | $[-500,500]^n$ | $f_8(\vec{420.97})=-12569.5$ |
| $f_9(\vec{x})=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$ | 30 | $[-5.12,5.12]^n$ | $f_9(\vec{0})=0$ |
| $f_{10}(\vec{x})=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$ | 30 | $[-32,32]^n$ | $f_{10}(\vec{0})=0$ |
| $f_{11}(\vec{x})=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$ | 30 | $[-600,600]^n$ | $f_{11}(\vec{0})=0$ |
| $f_{12}(\vec{x})=\frac{\pi}{n}\{10\sin^2(\pi y_1)$ $+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]$ $+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ | 30 | $[-50,50]^n$ | $f_{12}(-\vec{1})=0$ |
| $f_{13}(\vec{x})=0.1\{\sin^2(3\pi x_1)$ $+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}$ $+\sum_{i=1}^{n}u(x_i,5,100,4)$ | 30 | $[-50,50]^n$ | $f_{13}(1,\ldots,1)=0$ |
| $f_{14}(\vec{x})=(0.002$ $+\sum_{j=1}^{25}(j+\sum_{i=1}^{2}(x_i-a_{ij})^6)^{-1})^{-1}$ | 2 | $[-65.54, 65.54]^n$ | $f_{14}(-\vec{31.95})=0.998$ |
| $f_{15}(\vec{x})=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$ | 4 | $[-5,5]^n$ | $f_{15}(0.1928,0.1908,0.1231,0.1358)$ $=3.075\times10^{-4}$ |
| $f_{16}(\vec{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$ | 2 | $[-5,5]^n$ | $f_{16}(-0.09,0.71)=-1.0316$ |
| $f_{17}(\vec{x})=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2$ $+10(1-\frac{1}{8\pi})\cos(x_1)+10$ | 2 | $[-5,15]^n$ | $f_{17}(9.42,2.47)=0.398$ |

| | | | |
|---|---|---|---|
| $f_{18}(\vec{x}) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$ | 2 | $[-2,2]^n$ | $f_{18}(0,-1)=3$ |
| $f_{19}(\vec{x}) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2]$ | 3 | $[0,1]^n$ | $f_{19}(0.114,0.556,0.852) = -3.86$ |
| $f_{20}(\vec{x}) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2]$ | 6 | $[0,1]^n$ | $f_{20}(0.201,0.15,0.477,0.275,0.311,0.657) = -3.32$ |
| $f_{21}(\vec{x}) = -\sum_{i=1}^{5}[(\vec{x}-a_i)(\vec{x}-a_i)^T+c_i]^{-1}$ | 4 | $[0,10]^n$ | $f_{21}(\approx\vec{4})=-10.3$ |
| $f_{22}(\vec{x}) = -\sum_{i=1}^{7}[(\vec{x}-a_i)(\vec{x}-a_i)^T+c_i]^{-1}$ | 4 | $[0,10]^n$ | $f_{21}(\approx\vec{4})=-10.6$ |
| $f_{23}(\vec{x}) = -\sum_{i=1}^{10}[(\vec{x}-a_i)(\vec{x}-a_i)^T+c_i]^{-1}$ | 4 | $[0,10]^n$ | $f_{21}(\approx\vec{4})=-10.7$ |

Table 14. The 23 Benchmark Functions

also included in Table 14. The problem set contains a diverse set of problems, including unimodal as well as multimodal functions, and functions with correlated and uncorrelated variables. Functions $f_1$ - $f_5$ are unimodal. Function $f_6$ is the step function, which has one minimum and is discontinuous. Function $f_7$ is a noisy quartic function. Functions $f_8$ - $f_{13}$ are multimodal functions where the number of local minima increases exponentially with the problem dimension. Functions $f_{14}$ - $f_{23}$ are low-dimensional functions which have only a few local minima. As still a preliminary study on the new algorithm, the optimization problems listed above are considered in this paper, and the more experiments are needed for future studies.

Where $n$ is the dimension size of the functions, $f_{\min}$ is the ideal function value, and $S \in R^n$ (search space).

Where $G$ is the maximum generation, Func. = Functions, Algo. = Algorithms, Accuracy stands for the fixed accuracy level, *Best* stands for the best function value over 30 runs, *Mean* indicates the mean best function values, *Std. Dev.* stands for the standard deviation, Time stands for the average CPU time (seconds) consumed within the fixed number of generations. Succ.Gens. and Succ. Time stand for the average generation and average CPU time (seconds) achieving the fixed accuracy, Succ. Runs stands for the success number over 30 runs.

### 3.2.2 Experimental setup

The algorithms used for comparison are differential evolution (DE) algorithm [47], particle swarm optimization with inertia weight (PSO-w) [40], PSO with constriction factor (PSO-cf) [41], and comprehensive learning particle swarm optimizer (CLPSO) [42]. In all the experiments, the same population size *popsize*=100, total 30 runs are made, and the experiments results are listed in Table 15 -Table 17. The initial population is generated uniformly and randomly in the range as specified in Table 14. The parameters of the PSO-w are that: learning rate $c_1$=$c_2$=2, inertia weight linearly decreased from 0.9 to 0.4 with run time increasing, the maximum velocity $v_{\max}$ is set at 20% of the dynamic range of the variable on each dimension; the parameters of the PSO-cf are that: $c_1$= $c_2$=2.01 and constriction factor χ=0.729844. The parameters of the CLPSO follow the suggestions from [42] except that the refreshing gap $m$=2 for functions $f_{14}$-$f_{23}$. The parameters of the SFS are that: $\delta = \delta' = 14$ . All the algorithms are run on a PC with Pentium 4 CPU 2.4GHz.

| Func. | Accuracy | Algo. | *Best* | *Mean* | *Std. Dev.* | Time | Succ.Gens. | Succ. Time | Succ. Runs |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ (*G* =1500) | 1e-6 | DE | 5.20e-14 | 3.74e-13 | 3.94e-13 | 5.4 | 933.4 | 3.7 | 30 |
| | | PSO-w | 1.79e-15 | 1.66e-13 | 4.59e-13 | 18.2 | 1056.3 | 12.1 | 30 |
| | | PSO-cf | 4.50e-45 | 2.28e-41 | 4.54e-41 | 19.8 | 349.8 | 4.3 | 30 |
| | | CLPSO | 3.22e-13 | 2.73e-12 | 1.68e-12 | 24.4 | 924.6 | 16.3 | 30 |
| | | SFS | 5.40e-34 | 8.78e-32 | 3.06e-31 | 18.5 | 573.8 | 7.55 | 30 |
| $f_2$ (*G* =2000) | 1e-6 | DE | 6.17e-10 | 3.74e-09 | 2.20e-09 | 9.0 | 1553.9 | 7.6 | 30 |
| | | PSO-w | 5.36e-12 | 6.67e-11 | 7.98e-11 | 26.2 | 1545.7 | 19.3 | 30 |
| | | PSO-cf | 3.29e-29 | 1.60e-00 | 4.22e-00 | 30.1 | 1612.7 | 22.5 | 23 |
| | | CLPSO | 1.63e-09 | 3.82e-09 | 1.73e-09 | 33.6 | 1453.8 | 21.3 | 30 |
| | | SFS | 3.36e-18 | 1.34e-14 | 7.28e-14 | 27.18 | 1323.7 | 18.7 | 30 |
| $f_3$ (*G*=5000) | 1e-6 | DE | 1.10e-11 | 1.85e-10 | 1.49e-10 | 32.8 | 3762.0 | 25.9 | 30 |
| | | PSO-w | 2.00e-02 | 2.40e-01 | 2.23e-01 | 75.0 | 5000 | 75.0 | 0 |
| | | PSO-cf | 3.01e-19 | 3.33e+02 | 1.78e+03 | 86.3 | 2736.1 | 42.5 | 26 |
| | | CLPSO | 3.37e-02 | 4.20 e-01 | 3.62e-01 | 93.9 | 5000 | 93.9 | 0 |
| | | SFS | 4.02e-23 | 3.03e-21 | 3.11e-21 | 81.1 | 2093.7 | 35.6 | 30 |
| $f_4$ (*G* =5000) | 1e-6 | DE | 6.83e-13 | 3.10e-02 | 8.70e-02 | 23.9 | 4423.3 | 20.2 | 9 |
| | | PSO-w | 1.18e-02 | 7.02e-02 | 4.66e-02 | 63.4 | 5000 | 63.4 | 0 |
| | | PSO-cf | 1.48e-16 | 7.13e-13 | 2.19e-12 | 73.2 | 2893.4 | 42.4 | 30 |
| | | CLPSO | 6.88e-04 | 2.05e-03 | 1.25e-03 | 83.9 | 5000 | 83.9 | 0 |
| | | SFS | 6.97e-19 | 3.77e-17 | 5.31e-17 | 68.5 | 2970.6 | 40.7 | 30 |
| $f_5$ (*G* =20000) | 1e-6 | DE | 0 | 3.47e-31 | 2.45e-30 | 84.1 | 3966 | 16.2 | 30 |
| | | PSO-w | 1.05e-02 | 1.82e+03 | 1.27e+03 | 251.5 | 20000 | 251.5 | 0 |
| | | PSO-cf | 1.87e-12 | 7.32e+03 | 2.46e+03 | 271.8 | 17837 | 242.4 | 9 |
| | | CLPSO | 1.68e-01 | 3.63e+01 | 3.12e+01 | 349.1 | 20000 | 349.1 | 0 |
| | | SFS | 7.00e-21 | 6.56e-16 | 1.81e-15 | 241.1 | 13827 | 172.4 | 30 |
| $f_6$ (*G* =1500) | 1e-6 | DE | 0 | 0 | 0 | 7.3 | 357.0 | 1.6 | 30 |
| | | PSO-w | 0 | 0 | 0 | 19.3 | 921.7 | 12.7 | 30 |
| | | PSO-cf | 0 | 0 | 0 | 20.7 | 189.0 | 2.6 | 30 |
| | | CLPSO | 0 | 0 | 0 | 25.7 | 723.5 | 12.5 | 30 |
| | | SFS | 0 | 0 | 0 | 21.8 | 109.9 | 1.52 | 30 |
| $f_7$ (*G* =5000) | 1e-4 | DE | 1.97e-03 | 4.66e-03 | 1.30e-03 | 29.5 | 5000 | 29.5 | 0 |
| | | PSO-w | 2.99e-03 | 6.28e-03 | 2.17e-03 | 72.5 | 5000 | 72.5 | 0 |
| | | PSO-cf | 9.86e-04 | 2.45e-03 | 1.38e-03 | 75.0 | 5000 | 75.0 | 0 |
| | | CLPSO | 1.03e-03 | 2.98e-03 | 9.72e-04 | 93.5 | 5000 | 93.5 | 0 |
| | | SFS | 4.74e-05 | 9.53e-05 | 3.26e-05 | 73.9 | 3860.8 | 64.1 | 18 |

Table 15. The simulation results for $f_1$-$f_7$

| Func. | Accuracy | Algo. | *Best* | *Mean* | *Std. Dev.* | Time | Succ.Gens | Succ. Time | Succ. Runs |
|---|---|---|---|---|---|---|---|---|---|
| $f_8$ ($G$ =5000) | -12000 | DE | -11719 | -11234 | 455.5 | 41.5 | 5000 | 41.5 | 0 |
| | | PSO-w | -10495 | -9363.3 | 445.3 | 72.8 | 5000 | 72.8 | 0 |
| | | PSO-cf | -10398 | -9026.1 | 656.9 | 83.3 | 5000 | 83.3 | 0 |
| | | CLPSO | -12569 | -12271 | 177.8 | 92.1 | 1774.2 | 28.4 | 30 |
| | | SFS | -8952 | -7216 | 721.9 | 74.3 | 5000 | 74.3 | 0 |
| $f_9$ ($G$ =5000) | 1e-3 | DE | 9.95e-00 | 8.10e+01 | 3.23e+01 | 36.1 | 5000 | 36.1 | 0 |
| | | PSO-w | 7.96e-00 | 2.10e+01 | 8.01e-00 | 67.0 | 5000 | 67.0 | 0 |
| | | PSO-cf | 2.69e+01 | 6.17e+01 | 1.84e+01 | 78.9 | 5000 | 78.9 | 0 |
| | | CLPSO | 9.91e-01 | 4.13e+00 | 1.79e+00 | 84.3 | 5000 | 84.3 | 0 |
| | | SFS | 2.98e-00 | 6.93e-00 | 1.68e-00 | 75.6 | 5000 | 75.6 | 0 |
| $f_{10}$ ($G$ =1500) | 1e-3 | DE | 5.79e-08 | 1.71e-07 | 7.66e-08 | 7.7 | 844.5 | 4.4 | 30 |
| | | PSO-w | 1.39e-07 | 1.66e-06 | 2.66e-06 | 21.0 | 1344.3 | 18.6 | 30 |
| | | PSO-cf | 2.67e-15 | 5.59e-01 | 7.30e-01 | 22.5 | 845.4 | 12.6 | 19 |
| | | CLPSO | 3.31e-06 | 6.81e-06 | 1.94e-06 | 27.1 | 1334.6 | 23.9 | 30 |
| | | SFS | 2.66e-15 | 8.82e-15 | 3.95e-15 | 21.5 | 552.8 | 8.2 | 30 |
| $f_{11}$ ($G$ =2000) | 1e-3 | DE | 0 | 4.44e-04 | 1.77e-03 | 10.8 | 714.4 | 4.0 | 30 |
| | | PSO-w | 0 | 1.59e-01 | 2.19e-02 | 28.5 | 1833.7 | 25.3 | 7 |
| | | PSO-cf | 0 | 1.11e-02 | 1.25e-02 | 30.9 | 1351.5 | 21.1 | 7 |
| | | CLPSO | 1.64e-14 | 2.96e-04 | 1.46e-03 | 36.7 | 1423.7 | 25.3 | 29 |
| | | SFS | 0 | 0 | 0 | 30.4 | 337.2 | 5.1 | 30 |
| $f_{12}$ ($G$ =1500) | 1e-3 | DE | 3.40e-15 | 3.67e-14 | 4.07e-14 | 9.5 | 594.7 | 3.8 | 30 |
| | | PSO-w | 8.85e-15 | 2.21 e-00 | 5.52e-00 | 29.0 | 1154.6 | 21.4 | 30 |
| | | PSO-cf | 1.57e-32 | 1.66e+01 | 1.81 e+01 | 31.9 | 698.1 | 15.7 | 21 |
| | | CLPSO | 8.80e-12 | 4.80e-11 | 3.96e-11 | 35.2 | 1023.9 | 23.5 | 30 |
| | | SFS | 2.60e-32 | 7.51e-31 | 2.08e-30 | 22.5 | 201.9 | 3.2 | 30 |
| $f_{13}$ ($G$=1500) | 1e-3 | DE | 4.13e-14 | 2.91e-13 | 2.88e-13 | 9.8 | 748.8 | 5.0 | 30 |
| | | PSO-w | 8.23e-07 | 5.72e+02 | 3.57e+02 | 37.0 | 778.7 | 18.8 | 29 |
| | | PSO-cf | 1.35e-32 | 2.40e+02 | 2.40e+02 | 33.6 | 606.8 | 13.6 | 22 |
| | | CLPSO | 1.18e-10 | 6.42e-10 | 4.46e-10 | 38.6 | 637.3 | 16.7 | 30 |
| | | SFS | 2.21e-32 | 4.90e-31 | 1.37e-30 | 22.4 | 266.5 | 4.2 | 30 |

Table 16. The simulation results for $f_8$-$f_{13}$

| Func. | Accuracy | Algo. | Best | Mean | Std. Dev. | Time | Succ.Gens | Succ. Time | Succ. Runs |
|---|---|---|---|---|---|---|---|---|---|
| $f_{14}$ ($G$=100) | 0.998+1e-3 | DE | 0.998 | 0.998 | 2.88e-16 | 1.2 | 32.5 | 0.3 | 30 |
| | | PSO-w | 0.998 | 1.026 | 1.52e-01 | 1.4 | 43.4 | 0.7 | 30 |
| | | PSO-cf | 0.998 | 0.998 | 8.69e-13 | 1.52 | 19.9 | 0.3 | 30 |
| | | CLPSO | 0.998 | 0.998 | 5.63e-10 | 2.1 | 37.5 | 0.8 | 30 |
| | | SFS | 0.998 | 0.998 | 1.43e-16 | 1.8 | 25.6 | 0.4 | 30 |
| $f_{15}$ ($G$ =4000) | 3.175×1e-4 | DE | 3.0749e-04 | 4.7231e-02 | 3.55e-04 | 31.5 | 3859.7 | 29.9 | 2 |
| | | PSO-w | 3.0749e-04 | 2.0218e-03 | 5.47e-03 | 40.3 | 2837.0 | 29.0 | 22 |
| | | PSO-cf | 3.0749e-04 | 2.0225e-03 | 5.47e-03 | 43.1 | 824.5 | 8.9 | 27 |
| | | CLPSO | 3.2847e-04 | 5.3715e-04 | 6.99e-05 | 67.7 | 1413.7 | 24.1 | 29 |
| | | SFS | 3.0749e-04 | 3.0749e-04 | 2.01e-19 | 54.5 | 612.9 | 8.7 | 30 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $f_{16}$ ($G$ =100) | -1.0317 | DE | -1.0316 | -1.0316 | 6.77e-13 | 0.6 | 24.7 | 0.1 | 30 |
| | | PSO-w | -1.0316 | -1.0316 | 8.80e-12 | 0.9 | 20.7 | 0.2 | 30 |
| | | PSO-cf | -1.0316 | -1.0316 | 5.92e-12 | 0.9 | 20.6 | 0.2 | 30 |
| | | CLPSO | -1.0316 | -1.0316 | 8.50e-14 | 1.5 | 79.4 | 1.3 | 30 |
| | | SFS | -1.0316 | -1.0316 | 5.90e-16 | 1.1 | 15.2 | 0.2 | 30 |
| $f_{17}$ ($G$ =100) | 0.3981 | DE | 0.3979 | 0.3979 | 1.14e-08 | 0.6 | 37.6 | 0.2 | 30 |
| | | PSO-w | 0.3979 | 0.3979 | 2.33e-12 | 0.9 | 32.4 | 0.3 | 30 |
| | | PSO-cf | 0.3979 | 0.3979 | 5.25e-12 | 0.9 | 21.4 | 0.2 | 30 |
| | | CLPSO | 0.3979 | 0.3979 | 1.08e-13 | 1.5 | 83.8 | 1.4 | 30 |
| | | SFS | 0.3979 | 0.3979 | 0 | 1.1 | 16.2 | 0.2 | 30 |
| $f_{18}$ ($G$ =100) | 3+1e-4 | DE | 3 | 3 | 3.31e-15 | 0.7 | 25.8 | 0.1 | 30 |
| | | PSO-w | 3 | 3 | 2.50e-11 | 1.0 | 48.1 | 0.5 | 30 |
| | | PSO-cf | 3 | 3 | 2.05e-11 | 1.0 | 31.1 | 0.3 | 30 |
| | | CLPSO | 3 | 3 | 5.54e-13 | 1.6 | 49.1 | 0.8 | 30 |
| | | SFS | 3 | 3 | 3.33e-15 | 1.1 | 24.8 | 0.2 | 30 |
| $f_{19}$ ($G$ =100) | -3.86+1e-4 | DE | -3.8628 | -3.8628 | 1.97e-15 | 0.7 | 14.6 | 0.1 | 30 |
| | | PSO-w | -3.8628 | -3.8628 | 2.66e-11 | 1.1 | 14.9 | 0.2 | 30 |
| | | PSO-cf | -3.8628 | -3.8628 | 2.92e-12 | 1.1 | 9.1 | 0.1 | 30 |
| | | CLPSO | -3.8628 | -3.8628 | 6.07e-12 | 1.7 | 28.2 | 0.4 | 30 |
| | | SFS | -3.8628 | -3.8621 | 2.60e-15 | 1.1 | 17.1 | 0.2 | 30 |
| $f_{20}$ ($G$ =200) | -3.32+0.01 | DE | -3.322 | -3.215 | 0.036 | 1.4 | 188.1 | 1.3 | 19 |
| | | PSO-w | -3.322 | -3.256 | 0.066 | 2.8 | 141.7 | 2.1 | 17 |
| | | PSO-cf | -3.322 | -3.277 | 0.058 | 2.8 | 91.2 | 1.3 | 15 |
| | | CLPSO | -3.322 | -3.274 | 0.059 | 3.5 | 122.2 | 2.1 | 13 |
| | | SFS | -3.322 | -3.322 | 1.36e-15 | 2.4 | 44.9 | 0.55 | 30 |
| $f_{21}$ ($G$ =100) | -10 | DE | -10.15 | -10.15 | 4.67e-06 | 1.0 | 48.2 | 0.5 | 30 |
| | | PSO-w | - 6.57 | - 2.01 | 1.10e-00 | 1.2 | 100 | 1.2 | 0 |
| | | PSO-cf | -10.15 | - 6.23 | 3.25e-00 | 1.3 | 86.4 | 1.1 | 13 |
| | | CLPSO | -10.14 | - 9.57 | 4.28e-01 | 1.8 | 80.2 | 1.5 | 17 |
| | | SFS | -10.15 | -10.15 | 5.70e-15 | 1.6 | 21.1 | 0.3 | 30 |
| $f_{22}$ ($G$ =100) | -10 | DE | -10.40 | -10.40 | 2.07e-07 | 1.2 | 39.5 | 0.5 | 30 |
| | | PSO-w | - 4.61 | - 2.14 | 8.34e-01 | 1.2 | 100 | 1.2 | 0 |
| | | PSO-cf | -10.40 | - 6.47 | 3.56e-00 | 1.4 | 49.5 | 0.7 | 21 |
| | | CLPSO | -10.34 | - 9.40 | 1.12e-00 | 1.9 | 43.2 | 0.8 | 23 |
| | | SFS | -10.40 | -10.40 | 4.66e-16 | 1.6 | 19.2 | 0.3 | 30 |
| $f_{23}$ ($G$ =100) | -10 | DE | -10.54 | -10.54 | 3.21e-06 | 1.3 | 38.1 | 0.5 | 30 |
| | | PSO-w | - 6.63 | - 2.20 | 1.01e-00 | 1.4 | 100 | 1.4 | 0 |
| | | PSO-cf | -10.54 | - 8.11 | 3.47e-01 | 1.8 | 51.5 | 0.9 | 19 |
| | | CLPSO | -10.46 | - 9.47 | 1.25e-00 | 2.0 | 47.4 | 1.0 | 25 |
| | | SFS | -10.54 | -10.54 | 1.65e-15 | 1.7 | 18.0 | 0.3 | 30 |

Table 17. The simulation results for $f_{14}$-$f_{23}$

### 3.2.3 Unimodal functions

The results of 30 independent runs for functions $f_1$ - $f_7$ are summarized in Table 15. From Table 15, SFS is successful over all the 30 runs for $f_1$ - $f_6$. For $f_7$, it is successful in 18 runs but all the PSOs failed over all the runs. Moreover, PSOs has more time consumption of achieving the fixed accuracy than that of SFS except that PSO-cf has smaller time consumption for $f_1$. Although DE has less time consumption within the fixed number of generations than SFS and PSOs, it failed in 21 runs for $f_4$ and all the 30 runs for $f_7$.

### 3.2.4 Multimodal functions

1. Multimodal functions with many local minima: Multimodal functions with many local minima are often regarded as being difficult to optimize. $f_8$ - $f_{13}$ are such functions where the number of local minima increases exponentially as the dimension of the function increases. The dimensions of $f_8$-$f_{13}$ were all set to 30 in our experiments as [60]. Table 16 gives the results of 30 independent runs. From Table 16, SFS is successful over all the 30 runs for functions $f_{10}$-$f_{13}$ but $f_8$ and $f_9$. For functions $f_{10}$-$f_{13}$, SFS has faster convergence speed with the fewer generations and computation time to achieve the fixed accuracy level than DE and PSOs except DE for $f_{10}$ and $f_{11}$.
2. Multimodal functions with only a few local minima: For functions $f_{14}$-$f_{23}$, the number of local minima and the dimension are small. Table 17 summarizes the results over 30 runs. From Table 17, it is apparent that SFS performs better than DE and PSOs for functions $f_{14}$-$f_{23}$.

Table 15 - Table 17 indicates that SFS is suitable for solving the most employed unimodal and multimodal function optimizations with better convergence ability. Compared with the three modified PSOs, SFS has better global search ability with more successful runs for the benchmark functions. The Tables also show that SFS has often higher computational complexity with more time consumption within the given generations than DE but PSO-cf and CLPSO.

## 4. References

[1] Andrew Sohn. Parallel bidirectional heuristic search on the EM-4 multiprocessor. In: Proceedings of the Sixth IEEE Symposium on Parallel and Distributed Processing, Dallas, TX, USA, 1994, pp. 100-107.

[2] Raphael B., Smith I.F.C.. A direct stochastic algorithm for global search. Applied Mathematics and Computation, 2003, vol.146, issues 2-3, pp. 729–758.

[3] Chaohua Dai, Weirong Chen, Yonghua Song and Yunfang Zhu. Seeker optimization algorithm: A novel stochastic search algorithm for global numerical optimization, Journal of Systems Engineering and Electronics, 2010, vol. 21, no. 2, pp. 300-311.

[4] Robert Michael Lewisa, Virginia Torczon, Michael W. Trosset. Direct search methods: then and now. Journal of Computational and Applied Mathematics, 2000, vol.124, issues 1-2, pp.191-207.

[5] Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. Second Edition. Hongkong: Pearson Education Asia Limited and Beijing: Tsinghua University Press, 2006, pp.120-122.

[6] Mathias Kern. Parameter adaptation in heuristic search: a population-based approach. Ph.D. thesis, Department of Computer Science, University of Essex, 2006.

[7] Mills Patrick, Tsang Edward, Zhang Qingfu, et al. A survey of AI-based meta-heuristics for dealing with local optima in local search. Technical Report Series, Report Number CSM-416, September 2004. Department of Computer Science, University of Essex, Colchester CO4 3SQ. (Available: http://cswww.essex.ac.uk/CSP/)

[8] Nurhan Karaboga. Digital IIR filter design using differential evolution algorithm. EURASIP Journal on Applied Signal Processing, 2005, no.8, pp. 1269-1276.
[9] Storn R. and Price K.. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, vol.11, no. 4, pp.341-359.
[10] Kennedy, J., Eberhart, R., and Shi, Yu. Swarm Intelligence. Morgan Kaufmann, San Francisco, CA, 2001.
[11] Bonabeau E., Dorigo M., Theraulaz G.. Inspiration for optimization from social insect behavior. Nature, 2002, vol.406, no.6, pp.39-42.
[12] Valle Y. del, Venayagamoorthy G., Mohagheghi S., et al. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Transactions on Evolutionary Computation, 2008, vol.12, no.2, pp.171-195.
[13] Marco Dorigo, Mauro Birattari, and Thomas Stützle. Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Computing Intelligence Magazine, 2006, vol.1, no.4, pp.28-39.
[14] Chaohua Dai, Yunfang Zhu and Weirong Chen. Seeker optimization algorithm, Lecture Notes in Artificial Intelligence, Y. Wang, Y. Cheung, and H. Liu (Eds.), Springer-Verlag Berlin Heidelberg: Revised selected paper from CIS 2006, pp. 167–176, 2007.
[15] Chaohua Dai, Weirong Chen, and Yunfang Zhu. Seeker optimization algorithm for digital IIR filter design, IEEE Transactions on Industrial Electronics, 2010, vol. 57, no. 5, pp. 1710-1718.
[16] Chaohua Dai, Weirong Chen, Yunfang Zhu and Xuexia Zhang. Seeker optimization algorithm for optimal reactive power dispatch, IEEE Transactions on Power Systems, 2009, vol. 24, no. 3, pp. 1218-1231.
[17] Chaohua Dai, Weirong Chen, Yunfang Zhu and Xuexia Zhang. Reactive power dispatch considering voltage stability with seeker optimization algorithm, Electric Power System Research, 2009, vol. 79, no. 10, pp. 1462-1471.
[18] Chaohua Dai, Weirong Chen, Zhanli Cheng, et al. Seeker Optimization Algorithm for Global Optimization: a Case Study on Optimal Modeling of Proton Exchange Membrane Fuel Cell (PEMFC). International Journal of Electrical Power and Energy Systems, accepted.
[19] Chaohua Dai, Weirong Chen, Yunfang Zhu, et al. Seeker optimization algorithm for tuning the structure and parameters of neural networks. Neurocomputing, accepted.
[20] Yongkang Zheng, Weirong Chen, Chaohua Dai, Weibo Wang. Stochastic focusing search: A novel optimization algorithm for real-parameter optimization. Journal of Systems Engineering and Electronics, 2009, vol. 20, no. 4, pp. 869-876.
[21] Yongkang Zheng, Weirong Chen, Chaohua Dai, et al. Optimization algorithm with stochastic focusing search (in Chinese). Control Theory & Applications, 2009, vol. 26, no. 8, pp. 915-917.
[22] Donald A. Pierre. Optimization Theory with Applications. New York: Dover Publications, Inc., 1986.
[23] Zimmermann Hans-Jürgen. “A fresh perspective on uncertainty modeling: uncertainty vs. uncertainty modeling,” in Fuzzy sets and operations research for decision support. Edited by Da Ruan, Chonghu Huang, Beijing: Beijing Normal University Press, 2000, pp.40-57.
[24] Zadeh L. A.. The concept of linguistic variable and its application to approximate reasoning. Information Science, 1975, vol.8, no. 3, pp.199-246.

[25] Vignesh Kumar, Ferat Sahin. Cognitive maps in swarm robots for the mine detection application. in Proc. of IEEE International Conference on Systems, Man and Cybernetics, 2003, vol.4, pp.3364-3369.
[26] Eustace D, Barnes DP, Gray JO. Co-operant mobile robots for industrial applications. in Proc. of the Inter. Conf. on Industrial Electronics, Control, and Instrumentation, 1993, vol.1, Maui, HI, USA , pp.39-44.
[27] James Kennedy. The particle swarm: Social adaptation of knowledge. In: Proceedings of IEEE International Conference on Evolutionary Computation, 1997, Indianapolis, IN, USA, pp. 303-308.
[28] Vito Trianni, Roderich Groß, Thomas H. Labella, et al. Evolving aggregation behaviors in a swarm of robots. W. Banzhaf et al. (Eds.): ECAL 2003, LNAI 2801, pp.865–874.
[29] Camazine S., Deneubourg J.-L., Franks N., Sneyd J., Theraulaz G., and Bonabeau E. Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ, 2001.
[30] Wooldridge M., Jennings N. R.. Intelligent agents: theory and practice. The Knowledge Engineering Review, 1995, vol.10, no.2, pp.115-152.
[31] Icek Ajzen. Residual effects of past on later behavior: Habituation and reasoned action perspectives. Personality and Social Psychology Review, 2002, vol.6, no.2, pp.107–122.
[32] Joel D. Hewlett, Bogdan M. Wilamowski, and Günhan Dündar. Optimization using a modified second-order approach with evolutionary enhancement. IEEE Trans. Ind. Electron., vol.55, no.9, pp.3374-3380, 2008.
[33] Steels L. Cooperation between distributed agents through self organization. in Dernazean Y, Müller J-P, eds. Decentralized AI: Proc. of the First European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-89), pp.175-196, 1990.
[34] Deyi Li. Uncertainty reasoning based on cloud models in controllers. Computers and Mathematics with Applications, vol.35, no.3, pp.99-123, 1998.
[35] Yu Liu, Zheng gin, and Zhewen Shi. Hybrid particle swarm optimizer with line search. in Proc. of the 2004 IEEE Inter. Conf. on Systems, Man and Cybernetics, pp.3751-3755, 2004.
[36] W. B. Langdon and Riccardo Poli. Evolving problems to learn about particle swarm and other optimizers. CEC-2005, vol.1, pp.81-88, 2005.
[37] D.E. Goldberg. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison Wesley, 1989.
[38] Clerc, M. “When nearer is better,” Online at https://hal.archives-ouvertes.fr/hal-00137320, 2007.
[39] J. Brest, S. Greiner, B. Bošković, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput., vol.10, no.6, pp. 646-657, 2006.
[40] Y. Shi and R. Eberhart, A modified particle swarm optimizer. in Proc. IEEE World Congr. Comput. Intell., pp.69-73, May 1998.
[41] M. Clerc and J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput., vol.6, no.1, pp.58-73, Feb. 2002.
[42] J. J. Liang, A.K. Qin, Ponnuthurai Nagaratnam Suganthan, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput., vol.10, no.3, pp.67-82, 2006.
[43] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. in Proc. of IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, pp.1785-1791, 2005.

[44] B. Zhao, C.X. Guo, Y.J. Cao, A multi-agent based particle swarm optimization approach for reactive power dispatch. IEEE Trans. Power Syst., vol.20, no.2, pp.1070-1078, 2005.
[45] Ray D. Zimmerman, Carlos E. Murillo-Sánchez and Deqiang Gan. Matlab Power System Simulation Package (Version 3.1b2), Available at http://www.pserc.cornell.edu/matpower/, 2006.
[46] J. Kennedy and R. Eberhart, Particle swarm optimization. in Proc. IEEE Int. Conf. Neural Netw., vol.4, pp.1942-1948, Nov. 1995.
[47] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, International Computer Science Institute, Berkeley, Calif, USA, March 1995.
[48] Manoj Fozdar, C. M. Arora and V. R. Gottipati. Recent trends in intelligent techniques to power systems. in Proc. of the 42nd International Universities Power Engineering Conference, pp.580-591, 2007.
[49] H. Yoshida, Y. Fukuyama, K. Kawata, et al. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. Power Syst., 2001, vol.15, no.4, pp.1232-1239.
[50] Ahmed A. A. Esmin, Germano Lambert-Torres, Antônio C. Zambroni de Souza. "A hybrid particle swarm optimization applied to loss power minimization," IEEE Trans. Power Syst., 2005, vol.20, no.2, pp.859-866.
[51] John G. Vlachogiannis and Kwang Y. Lee. "A comparative study on particle swarm optimization for optimal steady-state performance of power systems," IEEE Trans. Power Syst., 2006, vol.21, no.4, pp.1718-1728.
[52] M. Varadarajan, K.S. Swarup, Network loss minimization with voltage security using differential evolution. Electric Power Systems Research, 2008, vol.78, pp.815-823.
[53] M. Varadarajan, K.S. Swarup, Differential evolutionary algorithm for optimal reactive power dispatch. Int. J. Electr. Power Energ. Syst, 2008, vol. 30, no. 8, pp. 435-441.
[54] Standard PSO 2007 (SPSO-07) on the Particle Swarm Central, Programs section: http://www.particleswarm.info, 2007.
[55] Q. H. Wu, Y. J. Cao, and J. Y. Wen, Optimal reactive power dispatch using an adaptive genetic algorithm. Int. J. Elect. Power Energy Syst., 1998, vol.20, no. 8, pp.563-569.
[56] Statistics Toolbox 5.0, The MathWorks, Inc.
[57] Juan Yu, "New models and algorithms of optimal reactive power flow and applications in voltage stability risk assessment," A Ph.D. thesis submitted to Chongqing University, pp:107-112, Chongqing, China, 2008. (in Chinese).
[58] Xiong Hugang, Cheng Haozhong, Li Haiyu. Optimal reactive power flow incorporating static voltage stability based on multi-objective adaptive immune algorithm. Energy Conversion and Management, 2008, vol. 49, no. 5, pp. 1175-1181.
[59] Wen Zhang and Yutian Liu. Multi-objective reactive power and voltage control based on fuzzy optimization strategy and fuzzy adaptive particle swarm. Int. J. Electr. Power Energy Syst., 2008, vol. 30, no. 9, pp. 525-532.
[60] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 1999, vol. 3, no. 2, pp. 82-102.