Research Article

Reactive Power Optimization of Power System Based on Improved Differential Evolution Algorithm

Rui Chi , Zheng Li , Xuexin Chi , Zhijian Qu , and Hong-bin Tu

School of Electrical and Automation Engineering, East China Jiaotong University, Nanchang 30013, China

Correspondence should be addressed to Rui Chi; rui85@126.com

Received 30 October 2020; Revised 29 December 2020; Accepted 22 January 2021; Published 13 February 2021

Academic Editor: Rahib Abiyev

Copyright © 2021 Rui Chi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a novel differential evolution (DE) algorithm, with its improved version (IDE) for the benchmark functions and the optimal reactive power dispatch (ORPD) problem. Minimization of the total active power loss is usually considered as the objective function of the ORPD problem. The constraints involved are generators, transformers tapings, shunt reactors, and other reactive power sources. The aim of this study is to discover the best vector of control variables to minimize power loss, under the premise of considering the constraints system. In the proposed IDE, a new initialization strategy is developed to construct the initial population for guaranteeing its quality and simultaneously maintaining its diversity. In addition, to enhance the convergence characteristic of the original DE, two kinds of self-adaptive adjustment strategies are employed to update the scaling factor and the crossover factor, respectively, in which the detailed information about the two factors can be exchanged for each generation dynamically. Numerical applications of different cases are carried out on several benchmark functions and two standard IEEE systems, i.e., 14-bus and 30-bus test systems. The results achieved by using the proposed IDE, compared with other optimization algorithms, are discussed and analyzed in detail. The obtained results demonstrated that the proposed IDE can successfully be used to deal with the ORPD problem.

1. Introduction

The optimal reactive power dispatch (ORPD) problem can be considered as an essential part of the optimal power flow (OPF) problem. It is a large-scale, nonlinear, discrete, and optimization problem [1] and refers to the reasonable regulation of reactive power through various technologies under the condition of sufficient reactive power, so as to achieve the optimal distribution of reactive power and the reasonable compensation of reactive power for various loads. The control variables of the ORPD problems include the generators, transformers tapings, shunt reactors, and other reactive power sources. The objective function involves voltage deviation, reactive power production cost, network active loss, and comprehensive cost of equipment adjustment. In general, it is challenging work to find an efficient and convenient approach to operate a modern power system, because we must consider the necessity to compensate the system for continually changing load demand and

provide energy of a high quality [2]. In recent research on the power system, ORPD has attained more attention in order to fulfill the needs of system security and operation as well as social economy. Therefore, proper distribution and efficient management of reactive power are the major issues which need to be solved urgently in our days [3].

ORPD problems have dealt with various decisions by using a large number of classical algorithms like linear programming [4], interior point methods [5], and Lagrange decomposition method [6]. From the survey about the use of classical algorithms, it may be observed that there are evident drawbacks in these classical algorithms such as insecure convergence, continuity limit, and excessive numerical iterations [7]. Fortunately, as an alternative choice, many researchers have transformed the focus from classical algorithms into computational intelligence-based algorithms which are used to solve the ORPD problems in various power systems [8]. Due to the nonlinear, nonconvex, and multimodal nature of the ORPD, most of the methods

applied to deal with it are based on computational intelligence-based algorithms [9]. In [9], the authors have made a brief literature review of the previous studies related to ORPD, in which different kinds of intelligent algorithms applied to the ORPD and their achievements and limitations are given. Cuckoo search (CS) algorithm [10], slime mould algorithm (SMA) [11], harmony search algorithm (HSA) [12], particle swarm optimization (PSO) algorithm [13, 14], chaotic krill herd (CKH) algorithm [15], genetic algorithm (GA) [16], immune algorithm (IA) [17], earthworm optimization algorithm (EWA) [18], elephant herding optimization (EHO) [19], moth search (MS) algorithm [20], Harris hawks optimization (HHO) [21], and artificial bee colony (ABC) algorithm [22] have been proposed and used to deal with the ORPD problem. With good flexibility, versatility, and robustness, they have attracted great attention.

Differential evolution (DE) algorithm is one of the most popular intelligent optimization algorithms and was first put forward by Storn and Price in 1995 [23]. It was originally used to solve real number coding and continuous function problems, and then it was quickly extended to optimize integer, discrete, and mixed integer problems. The main feature of DE is its simple structure, the low parameter requirements, and the ease of use especially in dealing with cases with nonlinear constraints [24]. Due to the above-mentioned advantages, DE has been widely used in the power system engineering, such as reactive power dispatching [25–29] and capacitor arrangement [30]. To further improve the convergence speed and optimization effect of DE in dealing with ORPD, many researchers have made various modifications from different perspectives like adjusting scaling factors and crossover factors, mixed with other algorithms and other operations.

Awad et al. [31] propounded an efficient DE algorithm for optimal active-reactive power dispatch problems, in which an arithmetic recombination crossover factor and a new scaling factor based on Laplace distribution were adapted to enhance the performance of the original DE algorithm. Wang et al. [32] proposed a differential evolution algorithm with an adaptive population size adjustment mechanism (SapsDE), which can adaptively determine a more appropriate mutation strategy and its parameter settings according to the previous state at different stages of the evolution process, thereby improving the performance of DE. Zhang et al. [33] divided the DE population into multiple groups firstly and presented a self-adaptive strategy for the control parameters for the purpose of achieving better results. Zhang et al. [34] proposed a self-adaptive differential evolution algorithm (JADE), which avoids the requirement for prior knowledge of parameter settings, so it can work well without user interaction. Gao et al. [35] proposed a novel selection mechanism to enhance the general DE algorithm (NSODE), which selects new individuals from N parents and N children as N better solutions to achieve better optimization results. The goal of modified DE algorithm in [36] was to enhance the convergence speed of the original DE by updating the adaptive scaling factor, which was able to dynamically exchange information for

each generation. Wang et al. [37] proposed a hybrid backtracking search optimization algorithm with differential evolution (HBD), which speeds up function convergence. Mohammad et al. [38] proposed a new hybrid algorithm based on the cluster center initialization algorithm (CCIA), bee algorithm (BA), and differential evolution (DE) (called CCIA-BADE-K) and evaluated its performance through standard data sets. Surender [39] presented a hybrid DE and harmony search algorithm for the optimal power flow problem. In [40], monarch butterfly optimization (MBO) is used to extricate the problem of optimal power flow (OPF) for standard IEEE 30- and 118-bus test power systems. Pulluri et al. [41] introduced an enhanced self-adaptive DE with mixed crossover algorithm for dealing with the multiobjective optimal power flow (MO-OPF) problems with conflicting objectives.

Although some improved DE algorithms for the ORPD problems have been achieved [31–41], further investigation is still indispensable for the complexity of objectives and constraints. Based on the previous work, this paper proposes an improved differential evolution (IDE) algorithm for solving ORPD problems. The crucial idea behind IDE is summarized as follows. A new initialization strategy with elimination and generation is constructed for guaranteeing the quality and the diversity of the initial population. In addition, intending to enhance the convergence speed of the original DE and reduce premature phenomenon, two self-adaptive adjustment methods are employed to update the scaling factor F and the crossover factor CR . F and CR are adaptively controlled by evolutionary generation, and the detailed information can be exchanged for each generation dynamically. The proposed IDE is used to optimize a series of benchmark functions and applied to IEEE 14-bus system and IEEE 30-bus system, aiming at minimizing the loss of active power and achieving good reactive power optimization effect.

The paper is organized as follows. Section 2 explains the typical formulation of an ORPD problem. Section 3 is dedicated to reviewing the fundamentals of the original DE algorithm. Section 4 discusses the structure of the proposed IDE algorithm. In Section 5, the feasibility of the proposed IDE is studied and tested on a series of standard benchmark functions. Section 6 covers optimization results and performance analysis of the applied IDE algorithm used to solve case studies on IEEE 14-bus system and IEEE 30-bus system. Section 7 presents the conclusion and outlines directions for further investigation.

2. Problem Description

The ORPD problem of power system is to determine the value of control variables in order to minimize the value of corresponding objective function. Control variables include generator terminal voltage, adjustable transformer ratio, and compensation capacity of shunt capacitor compensator. Generally, the main objective is optimizing generator fuel cost, active power loss or voltage stability, etc. The united form of the objective function is formulated as

$$\begin{aligned} & \text{Minimize : } f_i(x, u), i = 1, 2, \dots, N, \\ & \text{Subject to : } g(x, u) = 0, h(x, u) \leq 0, \end{aligned} \quad (1)$$

where f_i is the objective function; N is the number of objective functions; g is the equality constraint; h is the inequality constraint; x is the vector of control variables; and u is the vector of state variables including generator active power output, generator bus voltage, and shunt VAR compensation. In the ORPD problem considered in this study, active power loss is chosen as the objective function.

2.1. Objective Function

2.1.1. Minimization of Active Power Loss. Active power loss is mainly composed of power loss caused by current flowing through transformers and power lines. More active power loss not only increases power generation cost, but also reduces power factor of power system. Therefore, the active power loss is one of the most important objective functions, and in this paper we take the active power loss as the optimizing objective. The specific formula is expressed as follows [42].

$$\min F = P_{\text{loss}} = \sum_{k=i,j}^{k \in N_E} G_{ij} (U_i^2 + U_j^2 - 2U_i U_j \cos \theta_{ij}), \quad (2)$$

where P_{loss} is the total active power loss, N_E is total number of nodes, and U_i, U_j are the voltage magnitudes at bus i and bus j , respectively. G_{ij}, θ_{ij} are the branch conductance and phase angles difference between bus i and bus j , respectively.

Due to the possibility that the reactive power of the generator and the voltage of PQ node may exceed the constraint, a common method is the use of penalty function [43]. The idea of the penalty method is to transform the constrained problem into an unconstrained one by introducing a penalty factor. In reactive power optimization, the penalty function can always play an appropriate role in the whole process of the algorithm operation, limiting the change of control variables and keeping the test system in a relatively reasonable state.

The penalty function of the constrained problem (2) can be defined as follows:

$$\left\{ \begin{aligned} \min F &= P_{\text{loss}} + \lambda_v \sum_{i=1}^{N_l} \Delta U_i^2 + \lambda_Q \sum_{i=1}^{N_G} \Delta Q_{Gi}^2, \\ \Delta U_i &= \frac{U_i - U_{\text{lim}}}{U_{i,\text{max}} - U_{i,\text{min}}}, \\ \Delta Q_{Gi} &= \frac{Q_{Gi} - Q_{G\text{lim}}}{Q_{Gi,\text{max}} - Q_{Gi,\text{min}}}, \end{aligned} \right. \quad (3)$$

where P_{loss} is the objective function. λ_v and λ_Q are penalty factors. ΔU_i and ΔQ_{Gi} are the limiting value of voltage and reactive power from generator. U_{lim} and $Q_{G\text{lim}}$ are set values when the corresponding variables exceed the maximum and minimum limits. They are defined as follows.

$$\begin{aligned} U_{\text{lim}} &= \begin{cases} U_{i,\text{max}}, & U_i > U_{i,\text{max}}, \\ U_{i,\text{min}}, & U_i < U_{i,\text{min}}, \end{cases} \\ Q_{G\text{lim}} &= \begin{cases} Q_{Gi,\text{max}}, & Q_{Gi} > Q_{Gi,\text{max}}, \\ Q_{Gi,\text{min}}, & Q_{Gi} < Q_{Gi,\text{min}}. \end{cases} \end{aligned} \quad (4)$$

2.2. Constraints

2.2.1. Equality Constraints. Equality constraints involve the balance of active power and reactive power in power system:

(a) Active power balance equation:

$$P_{Gi} - P_{Di} - U_i \sum_{j \in N_i} U_j (G_{ij} \cos \theta_{ij} + B_{ij} \sin \theta_{ij}) = 0, \quad i \in N_0. \quad (5)$$

(b) Reactive power balance equation:

$$Q_{Gi} + Q_{Ci} - Q_{Di} - U_i \sum_{j \in N_i} U_j (G_{ij} \sin \theta_{ij} + B_{ij} \cos \theta_{ij}) = 0, \quad i \in N_{PQ}, \quad (6)$$

where G_{ij}, B_{ij} , and θ_{ij} are the transfer conductance, susceptance, and voltage angle difference between bus i and bus j , respectively. P_{Gi} and Q_{Gi} are the active and reactive power output of the generating units at bus i , respectively; P_{Di} and Q_{Di} are the demanded active power and reactive power of loads at bus i , respectively; and Q_{Ci} is the reactive power compensation capacity of node i .

2.2.2. Inequality Constraints

(i) Generator bus voltages limit:

$$U_{Gi,\text{min}} \leq U_{Gi} \leq U_{Gi,\text{max}}, \quad i \in N_G. \quad (7)$$

(ii) Generator active power outputs limit:

$$P_{Gi,\text{min}} \leq P_{Gi} \leq P_{Gi,\text{max}}, \quad i \in N_G. \quad (8)$$

(iii) Generator reactive power outputs limit:

$$Q_{Gi,\text{min}} \leq Q_{Gi} \leq Q_{Gi,\text{max}}, \quad i \in N_G. \quad (9)$$

(iv) Transformer tap settings limit:

$$K_{T,\text{min}} \leq K_T \leq K_{T,\text{max}}, \quad i \in N_T. \quad (10)$$

(v) Shunt VAR compensations limit:

$$Q_{C,\text{min}} \leq Q_C \leq Q_{C,\text{max}}, \quad i \in N_C. \quad (11)$$

(vi) Transmission line loadings limit:

$$S_l \leq S_l^{\text{max}}, \quad l \in N_l, \quad (12)$$

where U_{Gi}, P_{Gi} , and Q_{Gi} are generator bus voltages, active power outputs, and reactive power outputs.

K_T is transformer tap settings. Q_C is shunt VAR compensations. S_l is transmission line loadings. N_G , N_T , N_C , and N_l are total number of generators, total number of transformers, total number of compensation capacitors, and the number of system nodes.

3. The Original Differential Evolution Algorithm

DE algorithm is a stochastic research approach based on group evolution. It deals with real-coded study cases and is extended to deal with mixed integer optimal problem, which depends on the idea of natural evolution. The evolving process is fully completed over $G_{\max}$ generations to reach a final solution which is achieved by the individuals' collaborating and competing. Supposing that "NP" is the size of population, a vector or a candidate solution in the population is represented as $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, D is the dimension of search space. DE starts by the initialization of population; then, it is improved by using the following operations: mutation, crossover, and selection, which are not mechanisms that can generate completely novel individuals because the existing individuals are updated on the basis of differences in the certain preselected individuals. The three operations are repeated continuously until reaching the preset stopping criterion. The detailed DE procedures are listed as follows.

3.1. Initialization. At the beginning of DE algorithm, an initial population is randomly created within constraint range according to the following formula:

$$\vec{x}_i = \vec{x}_i^{\min} + \text{rand}(0, 1) \times (\vec{x}_i^{\max} - \vec{x}_i^{\min}), \quad i = 1, 2, \dots, NP, \quad (13)$$

where $\text{rand}(0, 1)$ is a uniformly distributed random number between 0 and 1. $\vec{x}_i^{\min}$ and $\vec{x}_i^{\max}$ are the lower and upper limits of the decision variable.

3.2. Mutation. After accomplishing the initialization, the following three operations can be applied: mutation, crossover, and selection. There are various strategies of these operations which have a full effect on the optimization process. From the different mutation types, we select DE/rand/1 in this paper. The mutation operation is largely dependent on perturbing the chosen vector using the difference between two other selected vectors. All of these selected vectors should be different from each other. A mutant vector $\vec{w}_i$ is generated by applying the following detailed mutation process.

$$\vec{w}_i(k+1) = \vec{x}_{r1}(k) + F \times (\vec{x}_{r2} - \vec{x}_{r3}), \quad (14)$$

where $i = 1, 2, \dots, NP$, $r1, r2, r3 \in \{1, 2, \dots, NP\}$, $i \neq r1 \neq r2 \neq r3$, F is the scaling factor and its value is between 0 and 2, and k is the current generation.

3.3. Crossover. To increase the diversity of the current population, the crossover process is introduced. A trial vector is produced by using the following crossover operation.

$$u_{ij}(k+1) = \begin{cases} w_{ij}(k+1), & \text{if } \text{rand}_j \leq CR \text{ or } j = j_{\text{rand}}, \\ x_{ij}(k), & \text{otherwise.} \end{cases} \quad (15)$$

Here, $j = 1, \dots, D$. CR is the crossover factor which is usually set to $(0, 1)$. rand_j is a random integer in $[1, D]$. The random integer rand_j can ensure that at least one element in the trial vector comes from the mutation vector.

3.4. Selection. In this process, the fitness of the current target vector $\vec{x}_i(k)$ is compared with the trial vector $\vec{u}_i(k+1)$ by using the greedy criterion, and the vector with better fitness can be selected to compose the next generation. The detailed selection operator is as follows.

$$\vec{x}_i(k+1) = \begin{cases} \vec{u}_i(k+1), & \text{if } f(\vec{u}_i(k+1)) \leq f(\vec{x}_i(k)), \\ \vec{x}_i(k), & \text{otherwise.} \end{cases} \quad (16)$$

4. The Proposed IDE Algorithm

In this section, the crucial idea behind the proposed IDE algorithms is discussed in detail.

4.1. A New Initialization Strategy. How to sample the characters of the target space by producing a minimum number of initial vectors is a vital issue, because the initial individuals with high quality may help the algorithm find out the final result more quickly. Due to the blindness when generating the initial population, a new initialization strategy will be presented in detail in this section. All the vectors in the initial population are divided into three parts according to the fitness value of these initial individuals. In concrete terms, this means that the fitness value of the initial population is calculated and sorted from low to high firstly, and now the current initial population can be renamed as a sorted population. In the sorted population, all individuals are evenly divided into three groups as follows: the first third of the sorted population is classified into group C , in which the fitness values are better than the other two groups. The middle third is classified into group E , and the last third is classified into group H . Because the individuals in group H have the worst fitness among the three groups, this phenomenon motivates us to rebuild group H which is renamed as H' to maintain better individuals. Employing the existing group C and group E , we construct a new group H' according to the following equation:

$$H' = \frac{\text{rand}(0, 1) \times (C + E)}{2}, \quad (17)$$

where $\text{rand}(0, 1)$ is a random number between $[0, 1]$; the purpose of dividing $(C + E)$ by 2 is to prevent the variable

from exceeding the upper limit. Then, a new initial population with higher quality is built by group C, group E, and group H. The experimental analysis of the initialization improvement strategy is presented in Section 5.1.

4.2. Self-Adaptive Scaling Factor. Scaling factor F is one of the essential parameters in DE algorithm, which is used in the mutation process. It controls the robustness and speed of the whole search. The value selection of F is also quite important, because low values of F will increase the speed of convergence but at the same time will enlarge the risk of being trapped in local optima [35]. From (14), it can be inferred that the disturbance to the basis vector can increase the diversity of the population. However, the degree of disturbance depends on the value selection of F. In the original DE, F is a fixed constant and has certain limitations to quicken the convergence speed and strengthen the global search capability. Ela believed that the scaling factor F is preferably 0.5 and the adjusting range is between 0.4 and 1 [24]. Based on the above-mentioned former work, following the adjusting range, a new self-adaptive scaling factor is constructed to dynamically change its value according to the number of iterations. The detailed updating equation is presented as follows.

$$F = e^{((G - G_{\max})/G_{\max})}, \quad (18)$$

where G is the current evolution generation and $G_{\max}$ is the maximum evolution generation. With the increase of iteration number, the value of scaling factor F changes dynamically between 0.37 and 1, which follows the adjusting range. The experimental analysis of the parameter improvement strategy is presented in Section 5.1.

4.3. Dynamic Crossover Factor. Like scaling factor in DE, crossover factor is also an indispensable parameter because the value of crossover factor CR has an important influence on the final obtained results. Higher values of CR will increase the inheriting genes from mutated vector $\vec{w}_i$ and enhance the local search capacity of the algorithm. Certainly lower values of CR will raise the proportion of the inheriting genes from target vector $\vec{x}_i$ and improve the global search performances of the algorithm. According to the above-mentioned suggestions, in the early stage of evolution, lower values of CR are set in order to make individuals of the population explore the global optimization more efficiently. In the later stage of evolution, in order to enhance the local research ability of the algorithm, higher values of CR are used. The dynamic crossover factor is constructed as follows:

$$CR = A \times B^{(G/G_{\max})}, \quad (19)$$

where G is the current evolution generation and $G_{\max}$ is the maximum evolution generation. The value of the cross factor CR is increased by a concave function ranging from A to $A \times B$.

In general, the value of parameter has great influence on the performance and convergence of the algorithm. According to the changes of IDE parameters A and B , the

following 8 groups were selected for sensitivity analysis. (1) $A = 0.05$, $B = 18$. (2) $A = 0.1$, $B = 10$. (3) $A = 0.15$, $B = 6$. (4) $A = 0.2$, $B = 5$. (5) $A = 0.25$, $B = 4$. (6) $A = 0.3$, $B = 3$. (7) $A = 0.35$, $B = 2.8$. (8) $A = 0.4$, $B = 2.5$. For example, from Figure 1, we can see that $A = 0.15$ and $B = 0.6$ are the best parameters for the Rastrigin function in IDE.

In this way, (19) can be converted into another form as follows:

$$CR = 0.15 \times 6^{(G/G_{\max})}. \quad (20)$$

According to (20), during consecutive generations, the value of CR is gradually increased from 0.15 to 0.9 within (0, 1). The excellent performance of this dynamic adjusting method is verified in Section 5.1.

4.4. IDE Procedure. By using the above-mentioned approach, the detailed steps to find the optimal results are described in this part. In summary, this proposed algorithm includes three improvements. The first improvement is that the initial population is generated by a new initialization strategy, which is described in detail in Section 4.1. The second improvement is that, according to repeated tests, the value of scaling factor in IDE is produced by a self-adaptive updating equation, which is suggested in Section 4.2. The third improvement is that the value of crossover factor in IDE alters dynamically by using (20), which is discussed in Section 4.3. The proposed IDE is summarized in Figure 2.

5. Experimental Studies on Benchmark Functions

5.1. Simulation Results of Three Improvements. The effectiveness of the three new components in the proposed algorithm is verified by experiments, respectively. We take f1 (Rastrigin function) as an example to verify the three improvement points. The name (Function), the formula (Formula), the range of dimension (D), the searching range (Range), and the known optimal value (Optima) of f1 (Rastrigin function) are listed in Table 1. The simulation results are shown in Figures 3–7 and Table 2. In the legend of Figures 3–7, “DE” represents the original DE algorithm, “DE-F” represents DE algorithm with self-adaptive scaling factor, “DE-CR” represents DE algorithm with dynamic crossover factor, “DE-initialization” represents DE algorithm with a new initialization strategy, “DE-F + CR” represents DE algorithm with self-adaptive scaling factor and dynamic crossover factor, “DE-F + initialization” represents DE algorithm with self-adaptive scaling factor and a new initialization strategy, “DE-CR + initialization” represents DE algorithm with dynamic crossover factor and a new initialization strategy, and “DE-F + CR + initialization” represents DE algorithm with the above three new components, which is also called IDE algorithm.

In these figures, the labels of X-axis and Y-axis represent the algorithm iteration numbers and the function value, respectively. From Figure 3, we compare the performance of the original DE and the DE-F, and the results show that the convergence speed of DE-F is faster. Figure 4 shows that

FIGURE 1: Curves of parameter A and B effect on optimal value.

Algorithm 1: An improved differential evolution algorithm (IDE)

Initialize objective function $f(\vec{x})$, $\vec{x} = (x_1, \dots, x_D)^T$
Initialize a population H' by using Eq. (13) and Eq. (17)
For all $\vec{x}_i (i = 1, 2, \dots, N)$ do
 Evaluate the fitness F_i
 $\vec{x}_{best} = \arg \min f(\vec{x}_i)$
End for
While $(k < \text{Max Generation})$ and $(\text{the stop criterion is not met})$ do
 for $i = 1$ to NP do
 Calculate the dynamic crossover factor by using (18)
 $\vec{w}_i(k+1) = \text{new solutions updated using (14)}$
 Calculate dynamic crossover factor by using (20)
 $\vec{u}_i(k+1) = \text{new solutions updated using (15)}$
 Evaluate its quality/fitness F_i
 $\vec{x}_i(k+1) = \text{new solutions updated using (16)}$
 End for
 Rank the solutions and find the current best
End while
Post-process results and visualization

FIGURE 2: Pseudocode of the proposed algorithm.

TABLE 1: Rastrigin function.

Function	Formula	D	Range	Optima
Rastrigin	$f_1(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	10	$[-5.12, 5.12]$	0

the convergence speed is obviously faster by using the improving CR strategy. From Figures 5(a) and 5(b), it can be clearly seen that DE algorithm with a new initialization

strategy performs better, and the convergence generation is reduced to 20. Therefore, in Figure 5(b), in order to provide a better display, the labels of X-axis are reduced to 60. Figure 6

FIGURE 3: Comparison of DE and DE-F.

FIGURE 4: Comparison of DE and DE-CR.

shows the comparison results between the original DE and DE-F + CR, and we can see that DE-F + CR convergence is faster. Figure 7 shows the convergence curves of these three algorithms: DE-F + initialization, DE-CR + initialization, and DE-F + CR + initialization, and we can draw the conclusion that DE-F + CR + initialization performs the best and DE-F + initialization performs the worst. From Figures 6 and 7, it can be seen that the combined effect of improvement F , improvement CR, and improvement initialization is better than that of a single improvement point.

Table 2 presents the computational results obtained for $f1$ (Rastrigin function). The best results among the eight algorithms are shown in bold. “Best” represents the best fitness, “Worst” represents the worst fitness, “Mean” represents the mean value of the fitness, and “Std” represents the standard deviation of the fitness value. Each algorithm

runs for 30 times. From the statistical results in Table 2, it can be seen that the optimal values of each algorithm can converge to zero. Among them, DE-F + CR + initialization, which combine the three improvement points, has the best performance, and it has achieved the best values of “Best,” “Worst,” “Mean,” and “Std.” To sum up, it is known that the improved algorithm with three improvement points has good performance, and this algorithm is also called the improved differential evolution (IDE) algorithm in this paper.

5.2. Simulation and Comparison. In order to further verify the performance of the proposed IDE algorithm and make a further comparative study, 10 well-known test functions are selected as benchmark problems [44–47]. These functions’ dimension can be fixed or unfixed. We set the value of dimension to 10 for unfixed dimension functions. The detailed information about these functions is listed in Table 3.

To test the effectiveness of IDE, the proposed IDE and other five algorithms, DE [23], CS [10], PSO [13], JADE [34], and SapsDE [32], are selected to test the functions above. Four evaluation indexes, “Best,” “Worst,” “Mean,” and “Std,” are used in the comparison. The explanation of these indexes is the same as Table 2. The detailed computational data of all test functions for all these algorithms are presented in Table 4 and Figures 8–17. For all algorithms, the population size is set as 75, and the total number of iterations is set as 2000. It should be noted that, for some functions, in order to show the convergence curves clearly, the convergence generation is changed flexibly in some figures. To reduce the random error of the simulation, all experiments on each test function are repeated 30 times. All computational experiments for the benchmark problems are implemented using Matlab 6.0 on a PC with an Intel core i5-4460 3.20 GHz processor and 8.0 GB memory.

The algorithms and other specific parameters settings are given below:

FIGURE 5: Comparison of DE and DE-initialization: (a) the original DE; (b) DE algorithm with a new initialization strategy.

FIGURE 6: Comparison of DE and DE-F + CR.

- (1) IDE: $p_{\min} = 0.005$, $p_{\max} = 0.5$, $\alpha_{\min} = 0.01$, $\alpha(\max) = 0.5$.
- (2) PSO [13]: $c_1 = c_2 = 2$, $w_{\min} = 0.4$, $w_{\max} = 0.9$.
- (3) DE [19]: $F = 0.5$, $CR = 0.9$.
- (4) CS [10]: $p_a = 0.25$, $\alpha = 1$.

The detailed statistical results are shown in Table 4. From Table 4, for Rastrigin function, it is remarkable that both IDE and SapsDE get the best “Best” result. IDE also gets the smallest values in terms of “Worst,” “Mean,” and “Std” results in comparison with the other five algorithms. Overall, IDE performs the best and it could approximate to the global optimum successfully. JADE and DE show a moderate performance. For Sphere function, it is a unimodal function and is usually used to verify the accuracy of optimization algorithms. In this paper, it is easily optimized by the six

algorithms. From the results of Table 4, the proposed IDE gets the smallest value of “Best,” “Worst,” “Mean,” and “Std.” That is to say, IDE shows the highest search performance in terms of the solution qualities. The performance of SapsDE is also promising but slightly worse than IDE. CS performs the worst.

For sum-of-squares function, IDE achieves the best “Best,” “Worst,” “Mean,” and “Std” results, and it shows the highest performance. JADE and SapsDE show a moderate performance. CS performs the worst. For Trid10 function, except for CS and JADE, all algorithms get the best “Best” result. Both SapsDE and IDE achieve the best “Mean” result. SapsDE achieves the best “Worst” and “Std” results. In general, SapsDE performs the best in comparison with the five algorithms, and IDE is also competitive and effective in solving Trid10 function.

FIGURE 7: Comparison of DE-F + initialization, DE-CR + initialization, and DE-F + CR + initialization.

TABLE 2: Test statistics results.

Function Index	Rastrigin					Time	Maxgen
	Best	Worst	Mean	Std			
DE	0	$2.98 E + 00$	$2.65 E - 01$	$6.36 E - 01$	24.25		4000
DE-F	0	$9.94 E - 01$	$9.94 E - 02$	$3.14 E - 01$	25.31		4000
DE-CR	0	0	0	0	21.73		4000
DE-initialization	0	0	0	0	0.64		60
DE-F + CR	0	0	0	0	21.296		4000
DE-F + initialization	0	$6.9 E - 14$	$2.42 E - 15$	$1.26 E - 14$	0.54		60
DE-CR + initialization	0	$7.1 E - 15$	$4.14 E - 16$	$1.59 E - 15$	0.60		60
DE-F + CR + initialization	0	0	0	0	0.53		60

TABLE 3: The benchmark test functions.

Function name	Function expression	D	Variable range	Optimal value
Rastrigin	$f_1(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	10	$[-5.12, 5.12]$	0
Sphere	$f_2(x) = \sum_{i=1}^D x_i^2$	10	$[-5.12, 5.12]$	0
Sum of squares	$f_3(x) = \sum_{i=1}^D i x_i^2$	10	$[-100, 100]$	0
Trid10	$f_4(x) = \sum_{i=1}^D (x_i - 1)^2 - \sum_{i=2}^D x_i x_{i-1}$	10	$[-100, 100]$	-210
Griewank	$f_5(x) = (1/4000) \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(x_i / \sqrt{i}) + 1$	10	$[-600, 600]$	0
Zakharov	$f_6(x) = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D 0.5 i x_i)^2 + (\sum_{i=1}^D 0.5 i x_i)^4$	10	$[-5, 10]$	0
Eason	$f_7(x) = -\cos(x_1) \cos(x_2) \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	2	$[-100, 100]$	-1
Matyas	$f_8(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	$[-10, 10]$	0
Beale	$f_9(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2) + (2.625 - x_1 + x_1x_2^3)^2$	2	$[-4.5, 4.5]$	0
Powell	$f_{10}(x) = \sum_{i=1}^{(D/k)} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$	8	$[-4, 5]$	0

The results of Griewank function and Zakharov function are also presented in Table 4. For Griewank function, DE, SapsDE, and IDE get the best “Best” result. IDE achieves the best “Best” results, and it also gets the best “Worst,” “Mean,” and “Std” results. In comparison with the other five algorithms, IDE performs the best. PSO performs the worst. For

Zakharov function, it is obvious that IDE shows the best performance by comparing the computational results of the other algorithms.

For Eason function, except for CS, the other five algorithms all get the best “best” result. DE and IDE provide the same best “worst” result. IDE gets the best “Mean” and “Std”

TABLE 4: Comparison results of six optimization algorithms.

Function		Rastrigin				Sphere			
Index	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
DE	0	2.98E+00	2.65E-01	6.36E-01	1.66E-05	1.80E-04	6.31E-05	3.72E-05	
CS	2.06E-05	3.27E-01	3.85E-02	6.92E-02	1.07E+00	7.60E+00	3.73E+00	1.57E+00	
PSO	1.98E+00	1.29E+01	5.53E+00	2.78E+00	8.00E-08	2.47E-01	2.26E-02	5.41E-02	
JADE	0	2.36E-20	1.83E-12	3.75E-16	5.69E-16	6.94E-5	5.32E-11	3.26E-15	
SapsDE	0	0	0	0	2.31E-35	2.38E-12	5.53E-22	8.63E-30	
IDE	0	0	0	0	3.30E - 91	5.76E-58	2.54E - 59	1.09E - 58	
Function		Sum of squares				Trid10			
Index	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
DE	4.39E-05	4.77E-04	1.93E-04	1.10E-04	-2.09E + 02	-2.09E+02	-2.09E + 02	3.14E-03	
CS	4.51E+00	2.64E + 01	1.53E+01	5.16E+00	-2.08E+02	-1.9E+02	-2.01E+02	3.20E+00	
PSO	1.33E-05	1.06E+01	7.17E-01	2.14E+00	-2.09E + 02	4.04E + 02	-1.14E+02	1.43E+02	
JADE	6.34E-56	6.29E-22	3.64E-31	5.14E-30	-2.08E+02	-2.05E+02	-2.07E+02	4.68E+01	
SapsDE	2.58E-30	3.17E-15	6.18E-25	4.59E-23	-2.09E + 02	-2.09E+02	-2.09E + 02	5.61E-06	
IDE	1.07E - 85	2.23E-60	7.43E - 62	4.07E - 61	-2.09E + 02	-2.06E+02	-2.09E + 02	9.09E-01	
Function		Griewank				Zakharov			
Index	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
DE	0	3.04E-01	9.92E-02	1.05E-01	1.02E-07	2.97E-06	8.96E-07	7.12E-07	
CS	1.53E-02	5.79E-02	3.76E-02	1.02E-02	1.57E-01	4.78E - 01	3.03E-01	9.30E-02	
PSO	6.09E-02	1.56E + 00	5.78E-01	4.15E-01	8.77E-14	3.78E-07	4.36E-08	9.54E-08	
JADE	3.25E-04	4.85E-02	4.63E-03	5.38E-07	4.28E-52	3.29E-35	1.39E-42	8.34E-26	
SapsDE	0	3.25E-15	2.38E-11	2.63E-10	5.69E-92	5.36E-64	3.28E-72	2.36E-51	
IDE	0	0	0	0	7.88E - 118	3.74E-80	1.24E - 81	6.83E - 81	
Function		Eason				Matyas			
Index	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
DE	-1.00E + 00	-9.99E-01	-9.99E-01	1.84E-14	5.20E-25	2.96E-20	2.05E-21	5.58E-21	
CS	-9.99E-01	-9.72E-01	-9.98E-01	4.97E-03	6.04E-10	1.65E - 07	2.32E-08	4.04E-08	
PSO	-1.00E + 00	-4.92E - 01	-6.27E-01	4.69E-01	3.37E-15	5.44E-10	2.84E-11	1.07E-10	
JADE	-1.00E + 01	-9.91E-01	-9.95E-01	3.25E-05	3.52E-58	5.36E-34	3.65E-46	8.95E-40	
SapsDE	-1.00E + 00	-9.53E-01	-9.89E-01	4.65E-07	2.34E - 92	3.68E-50	4.69E - 72	3.86E-31	
IDE	-1.00E + 00	-9.99E-01	-1.00E + 00	2.06E - 17	1.20E-61	1.75E-45	7.73E-47	3.28E - 46	
Function		Beale				Powell			
Index	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
DE	2.76E - 23	4.64E-03	1.78E-04	8.52E - 04	2.39E-04	3.51E-03	1.17E-03	8.90E-04	
CS	4.85E-07	4.16E-04	8.28E-05	1.20E-04	1.19E-01	2.26E + 00	8.71E-01	4.76E-01	
PSO	4.76E-13	7.62E-01	1.27E-01	2.88E-01	3.98E-05	1.93E-02	4.39E-03	5.16E-03	
JADE	2.69E-18	5.38E-08	2.21E - 11	3.27E-03	2.68E-43	3.22E-28	1.13E-31	2.34E-22	
SapsDE	8.56E-11	9.35E-06	1.38E-08	2.36E-02	3.22E-52	3.25E-19	8.38E-27	6.59E-29	
IDE	3.26E-12	9.64E - 01	3.21E-02	1.76E-01	5.35E - 55	2.33E-30	8.12E - 32	4.26E - 31	

results. Overall, IDE shows the best performance, and DE shows a moderate performance. For Matyas function, it is obvious that SapsDE performs the best in comparison with the other five algorithms. IDE is also competitive and effective in solving Matyas function. CS shows the worst performance.

For Beale function, DE gets the best “Best,” “Worst,” and “Std” results. JADE gets the best “Mean” results. In general, DE gets the best performance, and JADE and IDE show a moderate performance. For Powell function, IDE performs the best in terms of the best “Best,” “Worst,” “Mean,” and “Std” results. JADE and Saps DE get a moderate performance. CS performs the worst.

To sum up, the proposed IDE algorithm performs the best in most functions. For Trid10 and Matyas function, the IDE algorithm is second only to the Saps DE algorithm.

There is one exception; that is, for Beale function, DE performs better than IDE.

In order to make a visualized and detailed comparison, Figures 8–17 give the convergence curves of the proposed IDE algorithm and the other five algorithms. The plot depicts some convergence trends of the six algorithms in a random run. The labels of X-axis in a figure show the iteration numbers, and Y-axis shows the objective best value. It needs to be noticed that, in some figures’ X-axes, the maximum iteration number is changed flexibly, and it is set as 100, 200, 300, 1000, or 4000. The main purpose of this is to show the convergence curves more clearly.

From Figures 8–14 and Figure 17, it can be observed that the proposed IDE quickly converges to the global optimum, and its convergence speed is faster than the other five algorithms on

FIGURE 8: The convergence curves of f1.

FIGURE 9: The convergence curves of f2.

f1-f7 and f10. From Figure 15, for function f8, at the beginning of the optimization process, JADE and SapsDE converge faster than IDE. However, the proposed IDE is apparently the fastest algorithm for finding the best result at later stage of the optimizing process. By carefully looking at Figure 16, for function f9, PSO, JADE, and SapsDE converge faster than the proposed IDE, while IDE is more capable of improving its solution and performs steadily at later stage of evolution. Overall, the proposed IDE is efficient in solving these benchmark problems, while JADE, SapsDE, and PSO show a moderate performance in the comparison.

Through the above analysis and discussion about Table 4 and Figures 8–17, we can draw the conclusion that the

proposed IDE performs well for most of the test functions in comparison with the other five algorithms and it is efficient in solving these benchmark functions.

6. Application Studies on ORPD Problem

In order to further verify the accuracy and effectiveness of the proposed IDE, we use the optimal reactive power dispatch (ORPD) problem as application. The objective of ORPD problem is to determine appropriate control variables and minimize the active power loss of the system. Tests are carried out on IEEE 14-bus test system and IEEE 30-bus test system. The obtained results are

FIGURE 10: The convergence curves of f_3 . FIGURE 12: The convergence curves of f_5 . FIGURE 11: The convergence curves of f_4 . FIGURE 13: The convergence curves of f_6 .

compared with seven algorithms: DE, DE-F + CR, JADE, SapsDE, CS, PSO, and BA. The mathematical model of ORPD problem has been established in Section 2. The IEEE 14-bus test system contains 10 control variables. The IEEE 30-bus test system contains 13 control variables. The population size is set to 66, and the maximum number of iterations (Maxgen) is set to 200. The simulation results on each algorithm are achieved over 30 independent trails. The results of the critical performance indexes are presented: the best active power losses (Best), the worst active power losses (Worst), the mean active power losses (Mean), and the standard deviation (Std). The platform for implementation of these algorithms is

the same as Section 5, and the power flow calculation is obtained by Newton–Raphson method in MATPOWER system.

6.1. IEEE 14-Bus Test System. IEEE 14-bus standard test system is one of the bus systems with moderate complexity of variables and networks. The following is information related to IEEE 14-bus test system. The one-line diagram of IEEE 14-bus system is shown in Figure 18. The reference power is 100 MW, and the initial active power loss is 13.49 MW [7].

The network has 14 nodes and 20 branches. There are five generators at nodes 1, 2, 3, 6, and 8, in which the

FIGURE 14: The convergence curves of f_7 . FIGURE 15: The convergence curves of f_8 .

balancing node is located at node 1 and the other nodes are PV nodes. The adjustable transformers are distributed on 3 branches: 4–7, 4–9, and 5–6. The shunt reactive power sources are included at nodes 9 and 14.

The detailed information about the bus data, the line, the minimum and maximum limits of real power generations, and the limits of the control variables all can be get from [16, 48].

From Table 5 and Table 6, it can be clearly seen that the IDE algorithm performs the best in comparison with the other seven algorithms. IDE gets the best active power losses “12.322 MW,” and its network loss reduction rate is 8.66%. DE, DE- F + CR, and CS obtain the same “Best”

value, the “Best” value of SapsDE ranks the second, and BA gets the worst “Best” value. In Table 5, it also can be observed that DE gets the best “Std” value, but the proposed IDE also gets a smaller “Std” value which reflects the good stability of IDE. In order to further illustrate the performance characteristics of these algorithms, the optimal fitness of the active losses objective over 200 iterations is plotted in Figure 19. From this figure, it can be clearly observed that the proposed IDE is apparently the fastest algorithm for finding the best result at later stage of the optimizing process. Table 7 gives the values of each control variable of the proposed IDE algorithm.

FIGURE 16: The convergence curves of f_9 . FIGURE 17: The convergence curves of f_{10} .

FIGURE 18: IEEE 14-node network.

TABLE 5: Statistical data for IEEE 14-bus test system.

Function Index	Best	Worst	Ploss Mean	Std	Maxgen
DE	1.2338E+01	1.2346E+01	1.2343E+01	1.8997E-07	200
DE-F + CR	1.2338E+01	1.2345E+01	1.2341E+01	1.4886E+05	200
JADE	1.2336E+01	1.2342E+01	1.2340E+01	2.5961E-04	200
SapsDE	1.2331E+01	1.2339E+01	1.2335E+01	3.2865E-05	200
CS	1.2338E+01	1.2341E+01	1.2338E+01	5.7574E-07	200
PSO	1.2339E+01	1.2462E+01	1.2419E+01	6.9027E-02	200
BA	1.2554E+01	1.2568E+01	1.2562E+01	7.9556E-03	200
IDE	1.2322E+01	1.2325E+01	1.2323E+01	8.5236E-06	200

TABLE 6: Results of comparison with seven algorithms.

Ploss (MW)	DE	DE-F + CR	JADE	SapsDE	CS	PSO	BA	IDE
Before optimization	13.49	13.49	13.49	13.49	13.49	13.49	13.49	13.49
After optimization	12.338	12.338	12.336	12.331	12.338	12.339	12.554	12.322
Decrease	1.152	1.152	1.154	1.159	1.152	1.151	0.936	1.168
Decrease rate (%)	8.54	8.54	8.55	8.59	8.54	8.53	6.94	8.66

FIGURE 19: Performance characteristics of algorithms for IEEE 14-bus test system for best solution.

6.2. IEEE 30-Bus Test System. The one-line diagram of IEEE 30-bus test system is shown in Figure 20. The reference power is 100 MW, and the initial active power loss is 17.53 MW [49].

The network has 30 nodes and 41 branches. There are six generators on nodes 1, 2, 5, 8, 11, and 13, in which the balancing node is located at node 1 and the other nodes are PV nodes. The adjustable transformers are distributed on 4 branches: 6–9, 6–10, 4–12, and 27–28. The shunt VAR compensation nodes are 3, 10, and 24.

The reactive power optimization problem has three types of control variables: generator terminal voltage U_i , transformer ratio K_T , and parallel reactive power compensation capacity Q_C at reactive power compensation points. The

value range of these control variables can be found in [46, 47].

The values of critical performance indexes for IEEE 30-bus test system using the considered eight algorithms for 30 runs are presented in Tables 8 and 9. According to the comparison result, it can be seen that the best active power loss of IDE is “15.972 MW,” and its network loss reduction rate is 8.89%, ranking the second, close to the first SapsDE effect and better than the other seven algorithms. The convergence characteristics of all algorithms are also shown in Figure 21. From Figure 21, we can clearly see that, compared with the other considered methods, the proposed IDE has a higher convergence speed and better results, and its convergence speed is higher than SapsDE. Table 10 shows

TABLE 7: The values of control variables after IEEE 14-bus test system optimization.

Control variables (p.u)	Node	IDE
V1	1	1.0999
V2	2	1.0826
V3	5	1.0507
V6	8	1.0999
V8	11	1.0999
T1	4-7	1.0551
T2	4-9	0.9624
T3	5-6	1.0563
Q3	9	0.2544
Q10	14	0.0586

FIGURE 20: IEEE 30-node network.

TABLE 8: Statistical data for IEEE 14-bus test system.

Function Index	Best	Worst	Ploss Mean	Std	Maxgen
DE	$1.5978E+01$	$1.5978E+01$	$1.5978E+01$	$1.0721E-06$	200
DE-F + CR	$1.5980E+01$	$1.5981E+01$	$1.5981E+01$	$1.6093E-04$	200
JADE	$1.5975E+01$	$1.5986E+01$	$1.5981E+01$	$6.2957E-04$	200
SapsDE	$1.5970E+01$	$1.5976E+01$	$1.5973E+01$	$3.4862E-05$	200
CS	$1.5978E+01$	$1.5979E+01$	$1.5978E+01$	$3.4787E-06$	200
PSO	$1.6083E+01$	$1.6093E+01$	$1.6088E+01$	$7.0240E-03$	200
BA	$1.6056E+01$	$1.6160E+01$	$1.6108E+01$	$7.3094E-02$	200
IDE	$1.5972E+01$	$1.5975E+01$	$1.5973E+01$	$4.3255E-06$	200

TABLE 9: Results of comparison with seven algorithms.

Ploss(MW)	DE	DE-F + CR	JADE	SapsDE	CS	PSO	BA	IDE
Before optimization	17.53	17.53	17.53	17.53	17.53	17.53	17.53	17.53
After optimization	15.978	15.980	15.975	15.970	15.978	16.083	16.056	15.972
Decrease	1.552	1.550	1.555	1.560	1.552	1.447	1.474	1.558
Decrease rate (%)	8.85	8.84	8.87	8.90	8.85	8.25	8.41	8.89

FIGURE 21: Performance characteristics of algorithms for IEEE 30-bus test system for best solution.

TABLE 10: The values of control variables after IEEE 30-bus test system optimization.

Control variables (p.u)	Node	IDE
V1	1	1.0999
V2	2	1.0842
V5	5	1.0524
V8	8	1.0595
V11	11	1.0999
V13	13	1.0999
T1	6–9	0.9011
T2	6–10	1.0674
T3	4–12	1.0624
T4	27–28	1.0321
Q3	3	0.1998
Q10	10	0.1999
Q24	24	0.1160

the values of each control variable of the proposed IDE algorithm.

7. Conclusions

In this paper, we propose an improved differential evolution (IDE) algorithm. It is used to optimize several benchmark test functions and solve the ORPD problem. We have verified the proposed IDE which has the following three improvement points.

The proposed IDE performs well in adopting a new initialization strategy to produce new solutions, and two new adaptive control strategies to generate self-adaptive scaling factors and dynamic cross factors. 10 benchmark test functions are used to verify the superior performance of IDE. Then, it is applied to solve the ORPD problem. Compared with other intelligent optimization algorithms, the proposed IDE performs much better than the considered seven algorithms: DE, DE-F + CR, JADE, SapsDE, CS, PSO, and BA.

Overall, the proposed IDE is suitable for solving the ORPD problem. Exploring a more efficient DE algorithm or establishing multiple objectives for the ORPD problem is the future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Authors' Contributions

All authors contributed equally to this study.

Acknowledgments

This work was supported by the Science and Technology Research Project of Education Department of Jiangxi Province (#GJJ190354 and #GJJ190295), the National Natural Science Foundation of China (#61961018), the Jiangxi Key Research and Development Plan (20192BBEL50006), and the Jiangxi Provincial Natural Science Foundation (#20181BAB202017 and #2020BAB204023).

References

- [1] W. S. Sakr, R. A. El-Sehiemy, and A. M. Azmy, "Adaptive differential evolution algorithm for efficient reactive power management," *Applied Soft Computing*, vol. 53, pp. 336–351, 2017.
- [2] K. Abaci and V. Yamacli, "Differential search algorithm for solving multi-objective optimal power flow problem," *International Journal of Electrical Power & Energy Systems*, vol. 79, pp. 1–10, 2016.
- [3] K. Medani, S. Sayah, and A. Bekrar, "Whale optimization algorithm based optimal reactive power dispatch: a case study of the Algerian power system," *Electric Power Systems Research*, vol. 163, pp. 696–705, 2018.
- [4] N. I. Deeb and S. M. Shahidehpour, "An efficient technique for reactive power dispatch using a revised linear programming approach," *Electric Power Systems Research*, vol. 15, no. 2, pp. 121–134, 1988.
- [5] J. Z. Zhu and X. F. Xiong, "Optimal reactive power control using modified interior point method," *Electric Power Systems Research*, vol. 66, no. 2, pp. 187–192, 2003.
- [6] V. A. Sousa, E. C. Baptista, and G. R. M. Costa, "Optimal reactive power flow via the modified barrier Lagrangian function approach," *Electric Power Systems Research*, vol. 84, no. 1, pp. 159–164, 2012.
- [7] M. Ghasemi, M. Taghizadeh, S. Ghavidel, J. Aghaei, and A. Abbasian, "Solving optimal reactive power dispatch problem using a novel teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 100–108, 2015.
- [8] S. Sayah, "Modified differential evolution approach for practical optimal reactive power dispatch of hybrid AC-DC power systems," *Applied Soft Computing*, vol. 73, pp. 591–606, 2018.
- [9] D. G. Rojas, J. L. Lezama, and W. Villa, "Metaheuristic techniques applied to the optimal reactive power dispatch: a review," *IEEE Latin America Transactions*, vol. 14, no. 5, pp. 2253–2263, 2016.
- [10] S. B. Raha, K. K. Mandal, and N. Chakraborty, "Hybrid SMES based reactive power dispatch by cuckoo search algorithm," *IEEE Transactions on Industry Applications*, vol. 55, no. 1, pp. 907–917, 2018.
- [11] M. Manar, R. Hegazy, A. Mokhtar, and E. M. Ahmed, "A new strategy based on slime mould algorithm to extract the optimal model parameters of solar PV panel," *Sustainable Energy Technologies and Assessments*, vol. 42, Article ID 100849, 2020.
- [12] D. Karthikaikannan and G. Ravi, "Optimal reactive power dispatch considering multi-type FACTS devices using harmony search algorithms," *Automatika*, vol. 59, no. 3–4, pp. 311–322, 2018.
- [13] S. Devi and M. Geethanjali, "Optimal location and sizing determination of distributed generation and DSTATCOM using particle swarm optimization algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 62, pp. 562–570, 2014.
- [14] T. M. Aljohani, A. F. Ebrahim, and O. Mohammed, "Single and multiobjective optimal reactive power dispatch based on hybrid artificial physics-particle swarm optimization," *Energies*, vol. 12, no. 12, p. 2333, 2019.
- [15] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.
- [16] P. Subbaraj and P. N. Rajnarayanan, "Optimal reactive power dispatch using self-adaptive real coded genetic algorithm," *Electric Power Systems Research*, vol. 79, no. 2, pp. 374–381, 2009.
- [17] S. A. Taher and S. A. Afsari, "Optimal location and sizing of DSTATCOM in distribution systems by immune algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 34–44, 2014.
- [18] I. Ghosh and P. K. Roy, "Application of earthworm optimization algorithm for solution of optimal power flow," 2019 international conference on opto-electronics and applied optics, *Optronix*, Article ID 8862335, 2019.
- [19] S. S. Dhillon, S. Agarwal, G.-G. Wang, and J. S. Lather, "Automatic generation control of interconnected power systems using elephant herding optimization," *Intelligent Computing Techniques for Smart Energy Systems*, vol. 607, pp. 9–18, 2020.
- [20] P. Singh, S. K. Bishnoi, and N. K. Meena, "Moth search optimization for optimal integration of DERs for annual energy loss minimization in distribution systems," in *Proceedings of the India International Conference on Power Electronics*, December 2018, Article ID 8709410.
- [21] C. Qu, W. He, X. Peng, and X. Peng, "Harris Hawks optimization with information exchange," *Applied Mathematical Modelling*, vol. 84, pp. 52–75, 2020.
- [22] X. He, W. Wang, J. Jiang, and L. Xu, "An improved artificial bee colony algorithm and its application to multi-objective optimal power flow," *Energies*, vol. 8, no. 4, pp. 2412–2437, 2015.
- [23] R. Storn and K. Price, "Differential evolution: a simple and efficient heuristic for global optimization over continuous space," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [24] A. A. A. Ela, M. A. Abido, and S. R. Spea, "Differential evolution algorithm for optimal reactive power dispatch," *Electric Power Systems Research*, vol. 81, no. 2, pp. 458–464, 2011.
- [25] J. M. Ramirez, J. M. Gonzalez, and T. O. Ruben, "An investigation about the impact of the optimal reactive power dispatch solved by DE," *International Journal of Electrical Power & Energy Systems*, vol. 33, no. 2, pp. 236–244, 2011.
- [26] Y. Li, Y. Wang, and B. Li, "A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow," *International Journal of Electrical Power & Energy Systems*, vol. 52, no. 1, pp. 25–33, 2013.
- [27] M. Varadarajan and K. S. Swarup, "Differential evolutionary algorithm for optimal reactive power dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 8, pp. 435–441, 2008.
- [28] M. Ghasemi, M. M. Ghanbarian, S. Ghavidel, S. Rahmani, and E. Mahboubi Moghaddam, "Modified teaching learning algorithm and double differential evolution algorithm for optimal reactive power dispatch problem: a comparative study," *Information Sciences*, vol. 278, pp. 231–249, 2014.

- [29] J. Sanam, S. Ganguly, A. K. Panda, and C. Hemanth, "Optimization of energy loss cost of distribution networks with the optimal placement and sizing of DSTATCOM using differential evolution algorithm," *Arabian Journal for Science and Engineering*, vol. 42, no. 7, pp. 2851–2865, 2017.
- [30] A. M. Shaheen, R. A. El-Sehiemy, S. M. Farrag, and S. M. Farrag, "Adequate planning of shunt power capacitors involving transformer capacity release benefit," *IEEE Systems Journal*, vol. 12, no. 1, pp. 373–382, 2018.
- [31] N. H. Awad, M. Z. Ali, R. Mallipeddi, and P. N. Suganthan, "An efficient Differential Evolution algorithm for stochastic OPF based active-reactive power dispatch problem considering renewable generators," *Applied Soft Computing*, vol. 76, pp. 445–458, 2019.
- [32] X. Wang and S. G. Zhao, "Differential evolution algorithm with self-adaptive population resizing mechanism," *Mathematical Problems in Engineering*, vol. 2013, Article ID 419372, 14 pages, 2013.
- [33] X. Zhang, W. Chen, C. Dai, and W. Cai, "Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization," *International Journal of Electrical Power & Energy Systems*, vol. 32, no. 5, pp. 351–357, 2010.
- [34] J. Q. Zhang and A. C. Sanderson, "Self-adaptive Differential Evolution with Fast and Reliable Convergence Performance," 2007, pp. 2251–2258, IEEE Congress on Evolutionary Computation, Singapore, 2007.
- [35] D. Gao, G. G. Wang, and W. Pedrycz, "Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 12, pp. 3265–3275, 2020.
- [36] W. S. Sakr, R. A. EL-Sehiemy, and A. M. Azmy, "Adaptive differential evolution algorithm for efficient reactive power management," *Applied Soft Computing*, vol. 53, pp. 336–351, 2017.
- [37] L. J. Wang, Y. W. Zhong, Y. L. Yin, W. T. Zhao, B. O. Wang, and Y. L. Xu, "A hybrid backtracking search optimization algorithm with differential evolution," *Mathematical Problems in Engineering*, vol. 2015, Article ID 769245, 16 pages, 2015.
- [38] B. B. Mohammad, S. Z. M. Hashim, E. N. B. Nor, and K. Z. A. Ahmed, "An effective hybrid of bees algorithm and differential evolution algorithm in data clustering," *Mathematical Problems in Engineering*, vol. 2015, Article ID 240419, 17 pages, 2015.
- [39] S. R. Surender, "Optimal power flow using hybrid differential evolution and harmony search algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 1077–1091, 2019.
- [40] V. Yadav and S. P. Ghoshal, "Optimal power flow for IEEE 30 and 118-bus systems using monarch butterfly optimization," in *Proceedings of the 2018 Technologies for Smart-City Energy Security and Power*, pp. 1–6, ICSESP, Bhubaneswar, India, 2018.
- [41] H. Pulluri, R. Naresh, and V. Sharma, "An enhanced self-adaptive differential evolution based solution methodology for multiobjective optimal power flow," *Applied Soft Computing*, vol. 54, pp. 229–245, 2017.
- [42] B. Zhao, C. X. Guo, and Y. J. Cao, "A multiagent-based particle swarm optimization approach for optimal reactive power dispatch," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1070–1078, 2005.
- [43] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming, Theory and Algorithm*, Academic Press, New York, NY, USA, 1979.
- [44] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.
- [45] A.-R. Hedar and M. Fukushima, "Tabu search directed by direct search methods for nonlinear global optimization," *European Journal of Operational Research*, vol. 170, no. 2, pp. 329–349, 2006.
- [46] L. Wang, F. Zou, X. Hei et al., "A hybridization of teaching-learning-based optimization and differential evolution for chaotic time series prediction," *Neural Computing and Applications*, vol. 25, no. 6, pp. 1407–1422, 2014.
- [47] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real parameter optimization," Technical Report, Nanyang Technological University, Singapore, 2005.
- [48] S. Biswas and N. Chakrabort, "Tuned reactive power dispatch through modified differential evolution technique," *Front. Energy*, vol. 6, no. 2, pp. 138–147, 2012.
- [49] S. Kartikpandya, "optimal reactive power dispatch (ORPD) using particle swarm optimization (PSO)," 2014, <https://in.mathworks.com/matlabcentral/fileexchange/41265-optimal-reactive-power-dispatch-orpd-using-particle-swarm-optimization-pso>.