An improved multiobjective ABC algorithm based on K-means clustering, called CMOABC, is proposed. To fasten the convergence rate of the canonical MOABC, the way of information communication in the employed bees’ phase is modified. For keeping the population diversity, the multiswarm technology based on K-means clustering is employed to decompose the population into many clusters. Due to each subcomponent evolving separately, after every specific iteration, the population will be reclustered to facilitate information exchange among different clusters. Application of the new CMOABC on several multiobjective benchmark functions shows a marked improvement in performance over the fast nondominated sorting genetic algorithm (NSGA-II), the multiobjective particle swarm optimizer (MOPSO), and the multiobjective ABC (MOABC). Finally, the CMOABC is applied to solve the real-world optimal power flow (OPF) problem that considers the cost, loss, and emission impacts as the objective functions. The 30-bus IEEE test system is presented to illustrate the application of the proposed algorithm. The simulation results demonstrate that, compared to NSGA-II, MOPSO, and MOABC, the proposed CMOABC is superior for solving OPF problem, in terms of optimization accuracy.

1. Introduction

Real-life optimization problems usually have several conflicting objectives. The solution of such problems is very difficult because their objectives tend to be in conflict with each other. So a decision maker (DM) needs to determine an optimal trade-off among them. These problems can be modeled as multiobjective optimization problems (MOPs). Generally, the optimal solution of a multiobjective optimization problem is a set of optimal solutions (largely known as Pareto-optimal solutions). A Pareto-optimal solution to an MOP is a candidate for the optimal trade-off but it is not possible to find a better solution than others from the Pareto-optimal solutions, because one cannot be better than others without any further information. Therefore, it is necessary to find as many Pareto-optimal solutions as possible [1].

Swarm intelligence (SI) is an innovative artificial intelligence technique for solving complex multiobjective optimization problems, such as Pareto-archived evolution strategy (PAES) [2], Pareto envelope-based selection algorithm- (PESA-) II [3], nondominated sorting genetic algorithm II (NSGA-II) [4], strength Pareto evolutionary algorithm (SPEA2) [5], indicator-based evolutionary algorithm (IBEA) [6], multiobjective particle swarm optimization (MOPSO) [7], multiobjective evolutionary algorithm based on decomposition (MOEA/D) [8], and two-best multiobjective particle swarm optimization (2LB-MOPSO) [9]. As the widely adopted intelligent search method, the artificial bee colony (ABC) algorithm is a powerful search technique that drew inspiration from the biological foraging behaviors observed in bee colony [10, 11]. Due to its simplicity and efficiency, ABC has become a high competitor in solving the MO problems. Many researchers have presented several existing multiobjective ABC (MOABC) algorithms [12, 13]. However, these proposed algorithms still suffer from low convergence rate and lack the diversity of swarm.
To conquer the weakness of initial MOABC, an improved multiobjective ABC algorithm based on K-means clustering, named CMOABC, is proposed. In the CMOABC, for accelerating the convergence rate of canonical MOABC, the employed bees’ phase is modified to adjust the strength of information communication. Additionally, the population is partitioned into several subpopulations based on K-means clustering. Information communication between the subpopulations depends on reclustering the population after each specific iteration. To further enhance the population diversity, a number of individuals with worse performance regenerate in the reclustering process. With the test against several multiobjective benchmark functions, the results demonstrate that the proposed CMOABC outperforms the MOABC, MOPSO, and NSGA-II on convergence metric and diversity metric.

In the real-world optimization problems, optimal power flow (OPF) is a classical multiobjective problem. Traditionally, the basic objective of optimal power flow (OPF) problem is to schedule the committed generating units to meet the system load demand at minimum operating cost while satisfying the various system equality and inequality constraints [14]. But the passage of clean air act amendments in 1990 forced the utilities to reduce the emission from fossil fuel fired thermal station [15]. Therefore, in addition to fuel cost, emission must also be considered as an objective. On the whole, OPF problem is a nonlinear, constrained optimization problem where many competing objectives are present.

In the early research, many traditional optimization techniques such as linear programming (LP) [16], nonlinear programming (NLP) [17, 18], quadratic programming (QP) [19], Newton method, and interior point method (IPM) [20–22] have already been employed to solve the OPF problem. However, these methods suffer from severe limitations in handling the OPF problem, including discrete-continuous functions, and complex constraints [20]. In order to overcome the limitations of classical optimization techniques, recently, a wide variety of the multiobjective heuristic evolutionary algorithms have been proposed to solve OPF problem, mainly including genetic algorithm (GA) [23, 24], enhanced genetic algorithm (EGA) [25], evolutionary programming (EP) [26, 27], multihive multiobjective bee algorithm (M²OBA) [28], modified shuffle frog leaping algorithm (MSFLA) [29], differential evolution (DE) [30, 31], Harmony search (HS) algorithm [32], biogeography-based optimization (BBO) [33, 34], multiobjective evolutionary algorithm (MOEA) [35], multiobjective particle swarm optimization [36, 37], and fuzzy clustering-based particle swarm optimization (FCPSO) [38]. However, all the mentioned mathematical techniques have some drawbacks such as being trapped in local optima and they are only suitable for considering a specific objective function in the OPF problem. Due to its outstanding performance on the test against benchmark functions, CMOABC is suitable for solving the multiobjective problems. Therefore, CMOABC is utilized to solve OPF problem. Compared with MOABC, MOPSO, and NSGA-II, the proposed CMOABC algorithm can accommodate considerable potential for solving OPF problem.

The rest of this paper is organized as follows. Section 2 presents optimal power flow (OPF) formulation, including the objective function and the constraints of the OPF problem. The proposed algorithm CMOABC is detailed in Section 3, and also in this section we test the CMOABC on benchmark functions: four ZDT and two DTLZ. Afterward, in Section 4, the CMOABC is applied to solve the OPF problem. Finally, Section 5 outlines the conclusions.

### 2. Optimal Power Flow Problem Formulation

The multiobjective optimal power flow (OPF) is a nonlinear, nonconvex optimization problem. The main goal of OPF is to optimize the settings of control variables in terms of one or more objective functions while satisfying several equality and inequality constraints. In general, the OPF problem can be mathematically formulated as follows:

\[
\text{Minimize } F(x, u) = (f_1(x, u), \ldots, f_m(x, u)) \\
\text{Subjected to } g(x, u) = 0, \\
h(x, u) \leq 0,
\]

where \( F(x, u) \) is the objective function to be optimized, \( g(x, u) \) is the equality constraints representing nonlinear power flow equations, and \( h(x, u) \) is the system operating constraints. Here, \( u \) is the vector of independent control variables including

1. generator active power output \( P_G \) except at slack bus \( P_{G1} \);
2. generator bus voltage \( V_G \);
3. transformer tap setting \( T \);
4. shunt VAR compensation \( Q_C \).

Hence, \( u \) can be expressed as

\[
u^T = [P_{G2} \cdots P_{GN_p}, V_{G1} \cdots V_{GN_p}, T_1 \cdots T_{N_T}, Q_{C1} \cdots Q_{CNC}]\]

where \( N_p, N_T, \) and \( N_C \) denote the number of generating units, number of regulating transformers, and number of shunt compensators, respectively. Generators active powers (except slack bus \( P_{G1} \)) and generators bus voltages are continuous variables, while the tap settings of the tap changing transformers and VAR injections of the shunt capacitors are discrete variables.

And \( x \) is the vector of dependent variables including

1. slack bus generated active power \( P_{G1} \);
2. load (PQ) bus voltage \( V_L \);
3. generator reactive power output \( Q_G \);
4. transmission line loading (line flow) \( S_L \).

Hence, \( x \) can be expressed as

\[
x^T = [P_{G1}, V_{L1} \cdots V_{LN_p}, Q_{G1} \cdots Q_{GN_p}, S_{L1} \cdots S_{LN}]\]

where \( N_p \) is the number of PQ buses and \( N_L \) is the total number of transmission lines.
2.1. Objective Function. In this paper, the OPF problem is to minimize three competing objective functions, namely, the total fuel cost, the total emission cost minimization, and total power losses, while satisfying several equality and inequality constraints. Generally, the problem is formulated as follows.

2.1.1. Minimization of Total Fuel Cost. This objective function is to minimize the total fuel cost \( f_{\text{cost}} \) of the system. The fuel cost curves of the thermal generators are modeled as quadratic cost curves and can be represented as follows:

\[
f_{\text{cost}} = \sum_{i=1}^{N} f_i \left( a_i P_{Gi}^2 + b_i Q_{Gi} + c_i \right) ,
\]

where \( a_i, b_i, \) and \( c_i \) are the fuel cost coefficients of the \( i \)th generator and \( P_{Gi} \) is real power output of the \( i \)th generator.

2.1.2. Minimization of Total Power Losses. The power flow solution gives all bus voltage magnitudes and angles. Then, the total MW active power loss in a transmission network can be described as follows:

\[
f_{\text{loss}} = \sum_{k=1}^{N_x} g_k \left( V_i^2 + V_j^2 - 2 V_i V_j \cos(\delta_i - \delta_j) \right) ,
\]

where \( N_x \) is the number of transmission lines, \( V_i \) and \( V_j \) are the voltage magnitudes at the \( i \)th bus and \( j \)th bus, respectively, and \( \delta_i \) and \( \delta_j \) are the voltage angles at the \( i \)th bus and the \( j \)th bus, respectively.

2.1.3. Total Emission Cost Minimization. In this paper, two important types of emission gasses, namely, sulphur oxides \( \text{SO}_x \) and nitrogen oxides \( \text{NO}_x \), are taken as the pollutant gasses. The emissions generated by each generating unit may be approximated by a combination of a quadratic and an exponential function of the generator active power output. Here, the total emission cost is defined as follows [26]:

\[
f_{\text{emission}} = \sum_{i=1}^{N} \left( \alpha_i + \beta_i P_{Gi} + \gamma_i P_{Gi}^2 \right) ,
\]

where \( f_{\text{emission}} \) is the total emission cost (ton/h) and \( \alpha_i, \beta_i, \) and \( \gamma_i \) are the emission coefficients of the \( i \)th unit.

2.2. Equality Constraints. The equality constraints \( g(x,u) \) represented by (2) are typical power flow equations which are defined as follows:

\[
P_{Gi} - P_{Di} - V_i \sum_{j=1}^{N} V_j \left( G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j) \right) = 0 \\
Q_{Gi} - Q_{Di} - V_i \sum_{j=1}^{N} V_j \left( G_{ij} \cos(\delta_i - \delta_j) - B_{ij} \sin(\delta_i - \delta_j) \right) = 0
\]

where \( N \) is the number of buses in the system. \( P_{Gi} \) and \( Q_{Gi} \) are the active and reactive power generators at the \( i \)th bus; \( P_{Di} \) and \( Q_{Di} \) are the active and reactive power loads at the \( i \)th bus, respectively; \( G_{ij} \) and \( B_{ij} \) are the real and imaginary parts of the \( j \)th element of the bus admittance matrix, respectively; \( V_i \) and \( V_j \) are the voltage magnitudes at the \( i \)th bus and the \( j \)th bus, respectively; \( \delta_i \) and \( \delta_j \) are the voltage angles at the \( i \)th bus and the \( j \)th bus, respectively.

The equality constraints of power balance equations shown in (9) are forced by unconstrained Newton-Raphson based power flow calculations; therefore, there is no need to integrate them into the objective function.

2.3. Inequality Constraints. The inequality constraints \( h(x,u) \) represented by (3) are the power system operating limits including the following.

(1) Generator Constraints. Generator active power \( P_{Gi} \), generator reactive power \( Q_{Gi} \), and generator voltage magnitude \( V_G \) are restricted by their lower and upper limits:

\[
P_{Gi,\text{min}} \leq P_{Gi} \leq P_{Gi,\text{max}} \quad i = 1, \ldots, N_g
\]

\[
Q_{Gi,\text{min}} \leq Q_{Gi} \leq Q_{Gi,\text{max}} \quad i = 1, \ldots, N_g
\]

\[
V_{Gi,\text{min}} \leq V_{Gi} \leq V_{Gi,\text{max}} \quad i = 1, \ldots, N_g
\]

(2) Transformer Constraints. Transformer taps have minimum and maximum setting limits:

\[
T_{Gi,\text{min}} \leq T_i \leq T_{i,\text{max}} \quad i = 1, \ldots, N_t
\]

(3) Switchable VAR Sources. The switchable VAR sources have restrictions as follows:

\[
Q_{Ci,\text{min}} \leq Q_{Ci} \leq Q_{Ci,\text{max}} \quad i = 1, \ldots, N_c
\]

(4) Security Constraints. These include the limits on the load bus voltage magnitudes and transmission line flows limits:

\[
V_{Li,\text{min}} \leq V_{Li} \leq V_{Li,\text{max}} \quad i = 1, \ldots, N_{pq}
\]

\[
|S_{L_i}| \leq S_{L_i,\text{max}} \quad i = 1, \ldots, N_f
\]

In this work, the penalty factor method is utilized for handling the inequality constraints. In this regard, each control vector which violates constraints will be fined by these penalty factors. Therefore, in the next step, this control vector will be deleted automatically.

3. Multiobjective Artificial Bee Colony Algorithm Based on K-Means Clustering

3.1. Canonical ABC Algorithm. The artificial bee colony (ABC) algorithm, proposed by Karaboga and Basturk [39] and further developed by Basturk and Akay [11, 40] for real-parameter optimization, which simulates the intelligent foraging behavior of a honeybee swarm, is one of the most recently introduced swarm-based optimization techniques.

The entire bee colony contains three groups of bees: employed bees, onlookers, and scouts. Employed bees explore...
the specific food sources; meanwhile they pass the food information to onlooker bees. The number of employed bees is equal to that of food sources; in other words, each food source owns only one employed bee. Then onlooker bees choose good food sources based on the received information and then further exploit the food near their selected food source. The food source with higher quality would have a larger opportunity to be selected by onlookers. There is a control parameter called “limit” in the canonical ABC algorithm. If a food source is not improved anymore when limit is exceeded, it is assumed to be abandoned by its employed bee and the employed bee associated with that food source becomes a scout to search for a new food source randomly. The fundamental mathematic representations are listed as follows [41].

Step 1 (initialization phase). In initialization phase, a group of food sources are generated randomly in the search space using the following equation:

\[ x_{ij} = x_j^{\text{min}} + \text{rand}(0, 1) \left( x_j^{\text{max}} - x_j^{\text{min}} \right), \quad (14) \]

where \( i = 1, 2, \ldots, \text{SN}, \ j = 1, 2, \ldots, D \). SN is the number of food sources. D is the number of variables, that is, problem dimension. \( x_j^{\text{min}} \) and \( x_j^{\text{max}} \) are the lower and upper bounds of the \( j \)th variable, respectively.

Step 2 (employed bees’ phase). In the employed bees’ phase, the neighbor food source (candidate solution) can be generated from the old food source of each employed bee in its memory using the following expression:

\[ v_{ij} = x_{ij} + \delta \left( x_{ij} - x_{kj} \right), \quad (15) \]

where \( k \) is a randomly selected food source and must be different from \( i \); \( j \) is randomly chosen indexes; \( \delta \) is a random number in range \([-1, 1]\].

Step 3 (onlooker bees’ phase). In the onlooker bees’ phase, an onlooker bee selects a food source depending on the probability value associated with that food source and \( P_i \) can be calculated as follows:

\[ P_i = \frac{\text{fitness}_j}{\sum_{j=1}^{\text{SN}} \text{fitness}_j}, \quad (16) \]

where fitness \( j \) is the fitness value of the \( j \)th solution.

Step 4 (scout bees’ phase). In the scout bees’ phase, if a food source cannot be improved further through a predetermined cycle (called “limit” in ABC), the food source is supposed to be abandoned. The employed bee subsequently becomes a scout. A new food source will be produced randomly in the search space using (14).

The employed, onlooker, and scout bees’ phase will recycle until the termination condition is met.

3.2. CMOABC for Optimization. This section presents the detailed description of the proposed multiobjective artificial bee colony algorithm based on K-means clustering (named CMOABC). The proposed algorithm makes improvements on two aspects: (1) modifying the method of information communication in employed bees’ phase and (2) partitioning the population using K-means clustering. The number of the clusters will be chosen from the predefined set \( G \). After each specific iteration, when the defined criteria are met, parts of individuals in clusters will be removed and an equal number of new individuals are regenerated randomly. Then the number of clusters will change into a new value of the set \( G \) and the population will be reclustered based on K-means clustering.

3.2.1. Modified Employed Bees’ Phase of ABC. To increase the exploitation of ABC algorithm and fasten the convergence rate, the employed bees’ phase is modified. The new food source (candidate solution) is generated using the following way:

\[ v_{ij} = x_{ij} + \phi \left( x_{ij} - x_{kj} \right) \quad \text{rand} \geq \text{Cr}, \quad (17) \]

\[ v_{ij} = x_{ij} + \varphi \left( x_{\text{best,ij}} - x_{ij} \right) \quad \text{rand} < \text{Cr}, \quad (18) \]

where \( x_{\text{best,ij}} \) is the position of the global best solution and the term \( x_{\text{best,ij}} \) can drive the new candidate solution to the global best solution, iter is the current iteration, and \( \text{iter}_\text{max} \) is the maximum iteration.

The parameter \( \text{Cr} \) in (18) plays an important role in balancing the exploration and exploitation of the candidate solution search. If \( \text{Cr} = 0 \), (18) is identical to (15). With the increase of \( \text{Cr} \), the probability of the candidate solution learning to the best solution increases correspondingly. In this way, in the beginning of optimization process, the proposed algorithm operates as the canonical artificial bee colony algorithm, which can well keep the population diversity; in the end of optimization process, the modified ABC algorithm has a considerable improvement on both convergence rate and local search.

Furthermore, in (17), the different learning intensities, which are determined by \( \phi \) and \( \varphi \), are adopted. We adopt the beta probability distribution [42] to tune these two parameters, which is based on the similar strategy of tuning parameters in [43]. The beta distribution is flexible for modeling data that are measured in a continuous scale on a truncated interval in range \([0, 1]\), since its density is a versatile way to represent different shapes depending on the values of the two parameters that index the distribution. The probability density (\( \text{Bd}_{\text{pd}} \)) of the beta distribution in the range \([0, 1]\) is calculated as follows:

\[ \text{Bd}_{\text{pd}} = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \quad (19) \]

where \( B \) is the beta function.
The mean (\(E\)) and variance (Var) of the beta distribution are calculated as follows:

\[
E(X) = \frac{\alpha}{\alpha + \beta},
\]

\[
\text{Var}(X) = \frac{\alpha \beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)}.
\] (20)

The values of \(\phi\) and \(\varphi\) are tuned by MATLAB\textsuperscript*\ script (see (21)). An example of \(\phi\) and \(\varphi\) values generated using beta distribution is shown in Figure I. Consider

\[
\alpha = \text{rand},
\beta = (0.7 \ast (\frac{\text{iter}}{\text{iter}_{\text{max}}}) + 0.2) \ast \text{rand},
\]

\[
\phi = \text{betarnd}(a, b, [1 1]),
\alpha = \text{rand},
\beta = (0.6 \ast (\frac{\text{iter}}{\text{iter}_{\text{max}}}) + 0.1) \ast \text{rand},
\varphi = \text{betarnd}(a, b, [1 1]),
\]

where \(\text{iter}_{\text{max}}\) denotes the max iteration and \(\text{iter}\) is the current iteration.

3.2.2. Cluster Setting. The K-means clustering method is employed to partition the population into subpopulations. The basic concepts of K-means clustering are presented firstly, and then we will give a detailed description of its application in our proposed algorithm.

(I) Basic Parameters of K-Means. The cluster centers are substituted for center positions of food sources and the formula of computing the centers is shown in (22). If the \(i\)th cluster contains \(n_i\) members and the members are denoted by \(x_1^i, x_2^i, \ldots, x_{n_i}^i\), then the center (\(\text{center}_i\)) is determined as

\[
\text{center}_i = \frac{\sum_{j=1}^{n_i} x_j}{n_i}.
\] (22)

The radius (\(R\)) of a cluster is defined as the mean distance (Euclidean) of the members from the center of the cluster. Thus, \(R\) can be written as

\[
R_i = \frac{\sum_{p=1}^{n_i} \| x_p - \text{center}_i \|}{n_i}.
\] (23)

(II) Clusters in the Proposed Algorithm CMOABC. In the proposed algorithm, the stochastically generated population is partitioned into \(n\) subpopulations based on the widely adopted K-means cluster method (Algorithm 1) [44]. The number of clusters is determined by the predefined set \(G = \{g_1, g_2, \ldots, g_m\}\), where \(g_1 > g_2 > \cdots > g_m\). Every cluster operates as the modified ABC introduced in the above section. During optimization, it may happen that two or more clusters come close to each other or get overlapped to a high degree. Then, they will practically search the same domain of the functional landscape. To avoid this scenario, the distances between each two clusters are calculated as follows:

\[
\text{Dis}_{\text{cluster}_i} = \| \text{center}_i - \text{Nei}_i \|,
\]

\[
\text{DIS}_m = 0.2 \ast \min(R_i, R_{i,\text{neighbor}}),
\]

where \(\text{Dis}_{\text{cluster}_i}\) is the distance between one cluster and its neighbor and \(\text{Nei}_i\) is the center of the \(i\)th cluster's neighbor. \(\text{center}_i\) is the center of the \(i\)th cluster.

If the distance between one cluster and its neighbors is smaller than the specific distance \(\text{DIS}_m\), one of the clusters will be removed and its nondomination solutions are stored:

\[
\text{TI} = \begin{cases} 
\text{floor}(0.03 \ast \text{iter}_{\text{max}}) & \text{if } \text{iter} \leq 0.5 \ast \text{iter}_{\text{max}}, \\
\text{floor}(0.06 \ast \text{iter}_{\text{max}}) & \text{if } \text{iter} > 0.5 \ast \text{iter}_{\text{max}}
\end{cases}
\] (26)

where \(\text{iter}_{\text{max}}\) is the maximum iterations; \(\text{iter}\) is the current iteration.

After each \(\text{TI}\) iteration, a certain number of individuals in each cluster should be regenerated according to this cluster’s contribution to the external archive that is used to store the nondomination solutions. For the \(j\)th cluster, the number of solutions updating to the external archive during each \(\text{TI}\) iteration is recorded in \(\text{Num}_{\text{Update}}(j)\). Then, according to its position in the sort of \(\text{Num}_{\text{Update}}\), the number of individuals needed to regenerate in the \(j\)th cluster is calculated in (27). The individuals that will be removed in cluster \(j\) are determined by nondomination sort:

\[
\text{Num}_{\text{regenerate}}(j) = \frac{\text{Sort}_{\text{Update}}(j)}{g_i} \ast \frac{\text{Num}_{\text{ind}}(j)}{2},
\]

where \(\text{Num}_{\text{ind}}(j)\) is the number of individuals in the \(j\)th cluster; \(\text{Sort}_{\text{Update}}(j)\) indicates the \(j\)th cluster’s position in
Figure 1: (a) $F$ and (b) CR values for a run generated using beta distribution in the beta distribution.

Algorithm 1: Pseudocode of CMOABC.

```plaintext
// Step 1. Initialization
Generalize a population of NP individuals in the search region randomly;
Set the numbers of clusters $G_l$; Create the external archive EA; initial the current iteration iter = 1;

// Step 2. Loop
while stopping criteria are not satisfied do
  Calculate $TI$ according to (26)
  Decide the number $g_i$ of clusters according to $G$
  Partition the whole population based on $K$-means clustering
  For all cluster
    Compute $Dis_{cluster}$ according to (24)
    Decide whether some clusters need to be removed according to $DIS_m$
    Select $x_{up}$ individuals based on non-domination.
  End for
  For $iter = 1:TI$
    Update the individuals' position in each cluster according to (17)
    Update the external archive and update $Num_Update$
    $iter = iter + 1$
  End for
  Calculate the number of individuals in each cluster needing to be regenerated, according to (27)
  Generate a certain number of the new individuals in every cluster;
End while
```

the sort of $Num_{Update}$; and $g_i$ is the current number of clusters.

Additionally, to prevent the size of a cluster ($Num_{ind}$) from becoming too large, we set a limit $C_{max}$ as the maximum of cluster size. If the number of individuals in a cluster exceeds the limit, the best $C_{max}$ solutions are reserved based on nondomination and others are removed to other clusters stochastically. Setting formula of $C_{max}$ is shown in

$$C_{max} = 1.5 \star \left( \frac{NP}{g_i} \right),$$

(28)

where NP is the number of individuals in the population and $g_i$ is the number of clusters in current.

3.3. External Archive and Crowding Distance

3.3.1. External Archive. In multiobjective optimization, in the absence of preference information, none of the solutions can be said to be better than the others. Therefore, in order to keep the best solutions, a fixed-sized external archive is used by the CMOABC algorithm. The external archive is used to keep a historical record of the nondominated vectors found along the search process [6, 45].
3.3.2. Crowding Distance. In order to prevent the number of nondominated solutions from exceeding the size of EA, the crowding distance [4] is used to remove the crowded solutions. But in the proposed algorithm, the rules of removing the crowded solutions are modified. In each iteration, the distances between each nondominant solution and adjacent nondominant solution are calculated first, whether the size of external archive reaches the defined size or not. The detailed improvement is described as follows.

1. The size of external archive exceeds the defined size of EA: calculate the crowding distance and remove the crowded solutions [4].

2. The size of external archive does not outnumber the defined size: the crowding distance will be compared with a metric DIS. DIS is defined in (29); those nondominant solutions in the EA whose distances are all less than DIS will be removed from the EA [44]. Consider

\[
\text{DIS} = \frac{\sum_{i=1}^{n-1} |f_i - f_{i-1}|}{n + 1}.
\]

(29)

3.3.3. Constraints Handling Rules. To solve constraints, this paper uses a modified way to handle constraints which is introduced by Mohamed and Sabry [45]. Based on the initial constraints handling methods [46], three modified rules are as follows.

1. The trial vector is feasible and the target vector is not.

2. The trial vector and target vector are both feasible, but the trial vector has fitness value smaller than or equal to the corresponding target vector.

3. The trial vector and target vector are both infeasible, but the trial vector has overall constraint violation smaller than or equal to the corresponding target vector.

The above selection procedure allows the trial vector to be entered in the new population if it has the same amount of constraint violation or objective function value as the target vector. Therefore, this simple modification can help the algorithm to spread out and pass through the search space, so the algorithm can escape from stagnation.

3.4. Benchmark Test

3.4.1. Test Function. To fully evaluate the performance of the CMOABC algorithm without a biased conclusion towards some chosen problems, in this paper, six commonly recognized benchmark functions have been used. Four of these test problems ZDT1, ZDT2, ZDT3, and ZDT6 [32] are of two objectives, while the other two, DLTZ2 and DLTZ6 [33], are of three objectives.

3.4.2. Performance Measures. In order to facilitate the quantitative assessment of the performance of a multiobjective optimization algorithm, two performance metrics are taken into consideration: (1) convergence metric \( \lambda \) and (2) diversity metric \( \eta \) [4].

Convergence Metric. This metric measures the extent of convergence to a known set of Pareto-optimal solutions:

\[
\lambda = \frac{\sum_{i=1}^{N} d_i}{N},
\]

(30)

where \( N \) is the number of nondominated solutions obtained with an algorithm and \( d_i \) is the Euclidean distance between each of the nondominated solutions and the nearest member of the true Pareto-optimal front. To calculate this metric, we find a set of \( H = 500 \) uniformly spaced solutions from the true Pareto-optimal front in the objective space. For each solution obtained with an algorithm, we compute the minimum Euclidean distance of it from \( H \) chosen solutions on the Pareto-optimal front. The average of these distances is used as the convergence metric \( \lambda \). Figure 2 shows the calculation procedure of this metric.

Diversity Metric. This metric measures the extent of spread achieved among the obtained solutions. Here, we are interested in getting a set of solutions that spans the entire Pareto-optimal region. This metric is defined as

\[
\eta = \frac{d_f + d_t + \sum_{i=1}^{N-1} d_i - \overline{d}}{d_f + d_t + (N - 1)\overline{d}},
\]

(31)

where \( d_f \) is the Euclidean distance between consecutive solutions in the obtained nondominated set of solutions and \( N \) is the number of nondominated solutions obtained by an algorithm, \( \overline{d} \) is the average value of these distances, \( d_f \) and \( d_t \) are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained nondominated set, as depicted in Figure 3.
3.4.3. Experimental Setting. The effectiveness of CMOABC will be demonstrated on various standard test problems in this section. In addition, the proposed algorithm will be compared with the nondominated sorting genetic algorithm II (NSGA-II) [4], the multiobjective particle swarm optimization (MOPSO) [47], and the multiobjective artificial bee colony algorithm (MOABC) [13]. In order to compare the different algorithms with a fair time measure, the number of function evaluations (Fes = 10,000) is used for the termination criterion.

For two test examples presented in this study, the CMOABC algorithm parameters are set as follows: the population size is 500 and the set \( G \) is defined as \{50, 20, 10, 5\}. For NSGA-II, the population size, crossover, and mutation probabilities are selected as 500, 0.8, and 0.2, respectively, for the two test examples. For MOPSO, the population size, mutation rate, and divisions for the adaptive grid are selected as 500, 0.5, and 30. For MOABC, the population size is selected as 500, and for other parameter settings refer to [13].

3.4.4. Two Objective Functions' Results. The results reported in terms of the best, worst, average, median, and standard deviation of the performance measures are listed in Tables 1–4. Figures 4–7 show the optimal front obtained by four algorithms for each two objective functions, respectively. The continuous lines represent the Pareto-optimal front, while mark spots represent found nondominated solutions by the algorithms.

On ZDT1 function, when given 10,000 function evaluations for four algorithms, Figure 4 shows that CMOABC, MOPSO, NSGA-II, and MOABC have found a well-distributed and diverse solution set in the entire Pareto-optimal region. However, NSGA-II and MOABC cannot achieve the true Pareto front for ZDT1. Table 1 shows that the performance of CMOABC in convergence metric is of three orders of magnitude better than NSGA-II, two orders better than MOABC, and one order better than MOPSO. For diversity metric, it can be observed that all algorithms have almost the same performance on ZDT1.

On ZDT2 function, Figure 5 shows that MOPSO and NSGA-II produce poor results on this test function and both of them cannot achieve the true Pareto front, and the results of the performance measures show that CMOABC and MOABC have better performance compared to the MOPSO and NSGA-II. From Table 3, it can be noticed that the performance of CMOABC and MOABC in convergence metric and diversity metric is better than that of MOPSO and NSGA-II. Moreover, CMOABC outperforms MOABC by one order of magnitude in terms of diversity metric.

On ZDT3 function, from Figure 6, it can be noticed that NSGA-II has difficulties in converging the true Pareto front, and other algorithms find a better converged set of nondominated solutions in ZDT3 compared to NSGA-II. Further, Table 3 also shows that NSGA-II is worse than CMOABC, MOPSO, and MOABC in convergence metric and diversity metric.

On ZDT6 function, Figure 7 and Table 4 show that the algorithms achieve similar performance as on ZDT3.

3.4.5. Three Objective Functions. The optimization results of CMOABC, MOPSO, MOABC, and NSGA-II algorithms for three objective DTLZ series problems are shown in Tables 5 and 6. The Pareto-optimal region and all nondominated solutions obtained by the four algorithms are shown in Figures 8–11.

From Figure 9, it is observed that all algorithms obtain similar performance on DTLZ2. However, Table 5 shows the performance of NSGA-II is worse than that of the other three algorithms in convergence, while the performance of
Table 1: Comparison of performance on ZDT1.

<table>
<thead>
<tr>
<th>ZDT1</th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
<th>Convergence metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>8.4932e−004</td>
<td>1.3879e−002</td>
<td>2.3012e−001</td>
<td>1.6892e−003</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>8.2398e−004</td>
<td>1.9086e−002</td>
<td>1.4321e−001</td>
<td>1.4309e−003</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>7.5656e−004</td>
<td>9.8867e−003</td>
<td>7.2362e−002</td>
<td>8.4798e−004</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>1.1021e−003</td>
<td>3.6609e−002</td>
<td>8.8031e−001</td>
<td>2.1979e−003</td>
</tr>
<tr>
<td></td>
<td>Std.</td>
<td>8.0988e−005</td>
<td>1.0157e−002</td>
<td>2.3715e−001</td>
<td>6.7121e−004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Diversity metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Median</td>
</tr>
<tr>
<td></td>
<td>Min</td>
</tr>
<tr>
<td></td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>Std.</td>
</tr>
</tbody>
</table>

Table 2: Comparison of performance on ZDT2.

<table>
<thead>
<tr>
<th>ZDT2</th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
<th>Convergence metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>2.0306e−004</td>
<td>1.0151e−003</td>
<td>8.1218e−002</td>
<td>1.5201e−001</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>3.3727e−004</td>
<td>9.4901e−004</td>
<td>9.5231e−002</td>
<td>7.9931e−002</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.2466e−005</td>
<td>6.9883e−004</td>
<td>2.1834e−002</td>
<td>6.8641e−004</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>1.8276e−003</td>
<td>1.0924e−003</td>
<td>1.1632e−001</td>
<td>6.1073e−001</td>
</tr>
<tr>
<td></td>
<td>Std.</td>
<td>6.0919e−004</td>
<td>2.5061e−004</td>
<td>3.3876e−002</td>
<td>6.5663e−001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Diversity metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Median</td>
</tr>
<tr>
<td></td>
<td>Min</td>
</tr>
<tr>
<td></td>
<td>Max</td>
</tr>
<tr>
<td></td>
<td>Std.</td>
</tr>
</tbody>
</table>

Figure 5: Pareto fronts obtained by CMOABC, MOPSO, MOABC, and NSGA-II on ZDT2.

Figure 6: Pareto fronts obtained by CMOABC, MOPSO, MOABC, and NSGA-II on ZDT3.
### Table 3: Comparison of performance on ZDT3.

<table>
<thead>
<tr>
<th>ZDT3</th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence metric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>4.3256e – 004</td>
<td>1.0012e – 003</td>
<td>2.9476e + 000</td>
<td>1.6421e – 002</td>
</tr>
<tr>
<td>Median</td>
<td>1.5908e – 004</td>
<td>9.7597e – 004</td>
<td>2.8105e + 000</td>
<td>2.9831e – 003</td>
</tr>
<tr>
<td>Min</td>
<td>9.3664e – 005</td>
<td>8.9187e – 004</td>
<td>5.0668e – 001</td>
<td>1.8277e – 003</td>
</tr>
<tr>
<td>Max</td>
<td>3.8939e – 003</td>
<td>2.5609e – 003</td>
<td>4.8218e + 000</td>
<td>5.2114e – 002</td>
</tr>
<tr>
<td>Std.</td>
<td>1.2977e – 003</td>
<td>2.4481e – 004</td>
<td>1.6288e + 000</td>
<td>2.6671e – 002</td>
</tr>
<tr>
<td></td>
<td>Diversity metric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>7.5592e – 002</td>
<td>6.4352e – 001</td>
<td>9.9802e – 001</td>
<td>8.0000e – 001</td>
</tr>
<tr>
<td>Median</td>
<td>6.9825e – 002</td>
<td>6.4185e – 001</td>
<td>8.9974e – 001</td>
<td>6.9871e – 001</td>
</tr>
<tr>
<td>Min</td>
<td>2.1118e – 002</td>
<td>5.9631e – 001</td>
<td>4.6617e – 001</td>
<td>5.0805e – 001</td>
</tr>
<tr>
<td>Max</td>
<td>6.8032e – 001</td>
<td>6.8111e – 001</td>
<td>1.0542e + 000</td>
<td>8.9392e – 001</td>
</tr>
<tr>
<td>Std.</td>
<td>2.2677e – 001</td>
<td>1.2147e – 002</td>
<td>2.3567e – 002</td>
<td>3.3288e – 002</td>
</tr>
</tbody>
</table>

### Table 4: Comparison of performance on ZDT6.

<table>
<thead>
<tr>
<th>ZDT6</th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence metric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3.0221e – 004</td>
<td>2.8623e – 003</td>
<td>3.1403e – 001</td>
<td>4.9315e – 001</td>
</tr>
<tr>
<td>Median</td>
<td>3.6059e – 004</td>
<td>9.2567e – 004</td>
<td>2.4214e – 001</td>
<td>8.6061e – 004</td>
</tr>
<tr>
<td>Min</td>
<td>9.2468e – 005</td>
<td>1.2073e – 005</td>
<td>6.6306e – 002</td>
<td>6.6709e – 004</td>
</tr>
<tr>
<td>Max</td>
<td>2.7199e – 003</td>
<td>1.9542e – 002</td>
<td>8.7220e – 001</td>
<td>6.2114e – 002</td>
</tr>
<tr>
<td>Std.</td>
<td>9.0662e – 004</td>
<td>7.8526e – 003</td>
<td>2.8452e – 001</td>
<td>6.6571e – 003</td>
</tr>
<tr>
<td></td>
<td>Diversity metric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>6.8407e – 002</td>
<td>5.9752e – 001</td>
<td>7.5905e – 001</td>
<td>6.5902e – 001</td>
</tr>
<tr>
<td>Median</td>
<td>6.7381e – 002</td>
<td>4.0185e – 001</td>
<td>7.4812e – 001</td>
<td>6.6971e – 001</td>
</tr>
<tr>
<td>Min</td>
<td>6.2059e – 002</td>
<td>3.9851e – 001</td>
<td>5.7135e – 001</td>
<td>5.8895e – 001</td>
</tr>
<tr>
<td>Max</td>
<td>6.1567e – 001</td>
<td>1.1455e + 000</td>
<td>9.4103e – 001</td>
<td>7.0702e – 001</td>
</tr>
<tr>
<td>Std.</td>
<td>2.0522e – 001</td>
<td>2.2147e – 001</td>
<td>9.7500e – 002</td>
<td>4.4958e – 002</td>
</tr>
</tbody>
</table>

![Figure 7: Pareto fronts obtained by CMOABC, MOPSO, MOABC, and NSGA-II on ZDT6.](image1)

![Figure 8: The true Pareto-optimal front on DTLZ2.](image2)

CMOABC is better than that of the other three algorithms in diversity.
Table 5: Comparison of performance on DTLZ2.

<table>
<thead>
<tr>
<th>DTLZ2</th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence metric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3.0895e−004</td>
<td>8.0023e−003</td>
<td>8.7896e−002</td>
<td>3.8684e−003</td>
</tr>
<tr>
<td>Median</td>
<td>3.6059e−004</td>
<td>6.5537e−003</td>
<td>8.0436e−002</td>
<td>3.8431e−003</td>
</tr>
<tr>
<td>Min</td>
<td>9.801e−005</td>
<td>5.2183e−003</td>
<td>6.1409e−002</td>
<td>3.7631e−003</td>
</tr>
<tr>
<td>Max</td>
<td>1.9765e−003</td>
<td>1.5690e−002</td>
<td>1.0582e−001</td>
<td>3.0772e−003</td>
</tr>
<tr>
<td>Std.</td>
<td>9.7699e−004</td>
<td>2.5380e−003</td>
<td>1.6798e−002</td>
<td>8.6924e−005</td>
</tr>
<tr>
<td>Diversity metric</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>4.4906e−002</td>
<td>3.9896e−001</td>
<td>6.1302e−001</td>
<td>4.2637e−001</td>
</tr>
<tr>
<td>Median</td>
<td>6.7381e−002</td>
<td>3.9685e−001</td>
<td>5.2999e−001</td>
<td>4.1521e−001</td>
</tr>
<tr>
<td>Min</td>
<td>6.2059e−002</td>
<td>2.9572e−001</td>
<td>3.6617e−001</td>
<td>3.8790e−001</td>
</tr>
<tr>
<td>Max</td>
<td>1.4906e−001</td>
<td>5.2561e−001</td>
<td>7.2542e−001</td>
<td>4.5618e−001</td>
</tr>
<tr>
<td>Std.</td>
<td>1.8649e−002</td>
<td>3.2234e−002</td>
<td>1.7567e−002</td>
<td>2.1692e−002</td>
</tr>
</tbody>
</table>

From Figure 11, it can be seen that CMOABC and MOABC have better performance on DTLZ6. The CMOABC especially has attained the nondominated set properly. The front obtained from the other two algorithms cannot be uniformly distributed on the true Pareto-optimal front on DTLZ6.

Table 6 also shows that MOABC has obtained better convergence and diversity than NSGA-II and MOPSO on DTLZ6, but it is found to be second to the CMOABC algorithm.

On the whole, for two and three objective problems, CMOABC has a more competitive performance than MOPSO, MOABC, and NSGA-II on both diversity metric and convergence metric. This is due to the fact that, in the proposed CMOABC algorithm, the modified employed bees’ phase with a self-adaptive strength of information communication effectively enhances its abilities of exploration in the early stage and exploitation in the late stage. Moreover, benefiting from the swarm decomposition operation with a dynamic K-means clustering method, CMOABC could ensure more sufficient exploitation around the potential solutions in the less crowded regions. When the number of clusters is large, the probability of the individuals with similar performances allocated into the same subcomponent increases, which improves the ability of global search.

When the number of clusters becomes small, the individuals trapped into local optimal may move toward the current best individual of their cluster, which effectively avoids the inefficiency of local search. In addition, the regeneration operation also enriches the diversity of external archive. Therefore, CMOABC can reasonably accommodate a considerable potential for solving more complex multiobjective optimization problems.

4. Multiobjective Optimal Power Flow Based on CMOABC

In order to prove the numerical correctness, efficiency, and validation of CMOABC algorithm for the model, in this section, the multiobjective OPF problem will be used to implement simulation experiment. The proposed algorithm will be applied to two objectives and three objectives of OPF problem and it is compared with NSGA-II, MOPSO, and MOABC to solve such problem.

4.1. Implementation of the CMOABC Algorithm for the OPF Problem

Step 1. Input the system parameters and the minimum and maximum limits of control variables.
Step 2. Input parameters of the CMOABC algorithm and the lower and upper limits of each variable.

Step 3. Produce the initial population.

Step 4. Cluster the population by K-means and compute the fitness.

Step 5. Store nondominated solutions in the memory table.

Step 6. Optimize every cluster individually.

Step 7. Handle the constraints.

Step 8. Compute the fitness of the solutions and store non-dominated solutions into EA.

Step 9. Update each EA of each hive according to greedy selecting strategy, sort the EA based on nondomination, and select the nondomination solutions to stay in EA. If the number of nondominated solutions exceeds the allocated size of EA, apply crowding distance to remove the crowded members.

Step 10. Judge the condition of reclustering. If it meets the condition, the number of clusters can be reset and go to Step 4; if not, go to Step 6.

4.2. Best Compromise Solution Using Fuzzy Membership Approach. In this work, a fuzzy-based mechanism is employed to extract the best compromise solution over the trade-off curve and assist the decision maker to adjust the generation levels efficiently [48]. Due to imprecise nature of the decision maker’s judgment, each objective function of the $i$th solution is represented by a membership function $\mu_i$ defined as follows:

\[
\mu_i = \begin{cases} 
1 & F_i \leq \min(F_i), \\
\frac{\max(F_i) - F_i}{\max(F_i) - \min(F_i)} & \min(F_i) \leq F_i \leq \max(F_i) \\
0 & F_i \geq \max(F_i),
\end{cases}
\]  

where $\min(F_i)$ and $\max(F_i)$ are lower and upper bounds of the $i$th objective function. The higher the values of the membership function are, the greater the solution satisfaction is.
For each nondominated solution, the normalized membership function \( \mu^k \) is calculated as

\[
\mu^k = \frac{\sum_{i=1}^{N_{\text{obj}}} \mu^k_i}{\sum_{k=1}^{M} \sum_{i=1}^{N_{\text{obj}}} \mu^k_i},
\]

(33)

where \( M \) is the number of nondominated solutions and \( N_{\text{obj}} \) is the number of objects. The best compromise solution is the one having the maximum of \( \mu^k \).

4.3. Simulation Results. In order to validate the robustness of the proposed CMOABC method, a standard IEEE 30-bus system has been used as the test system [28]. The system represents a portion of the American Electric Power System (in the Midwestern USA) as of December 1961, which consists of six generators, 41 transmission lines, and 4 transformers with off-nominal tap ratio in lines 6–9, 6–10, 4–12, and 27–28. The system data with minimum and maximum limits of control variables are shown in Table 7 [49]. The limits of generator buses and load buses are between 0.95 and 1.1 p.u. and 0.9 and 1.05 p.u., respectively. The lower and upper limits of transformer taps are 0.9 p.u. and 1.05 p.u., respectively, and the step size is 0.01 p.u. The parameter settings for these four algorithms are the same as in "performance setting" in Section 3.4.3. In the following section, there are two cases of the test system considered.

(1) Two objectives: emission cost, emission loss, and loss cost are considered, respectively.

(2) Three objectives: they are emission, cost, and loss.

4.3.1. Case I: Two-Objective OPF Optimization. In this case, the results obtained from the proposed CMOABC have been compared with MOPSO, MOABC, and NSGA-II in three cases: minimum cost, minimum emission, and minimum loss. Figures 12–14 show the convergence characteristics of the four algorithms on three two-objective OPF problems, respectively. For the case of emission cost, Table 8 gives the best solutions for minimum cost and minimum emission; the data of emission loss and loss cost are shown in Tables 10 and 12. Finally, the best compromise solutions for each objective in the two-dimensional Pareto front are shown in Tables 9, 11, and 13.
### Table 7: Characteristics of the generation units.

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{C_{\text{max}}}$ (MW)</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>$P_{C_{\text{min}}}$ (MW)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

#### Generator limits

- **Cost coefficients**
  - $a$: 10, 10, 20, 10, 20, 10
  - $b$: 200, 150, 180, 100, 180, 100
  - $c$: 100, 120, 40, 60, 40, 100

#### Emission coefficients

- $\alpha$: 4.091, 2.543, 4.258, 5.326, 4.258, 6.131
- $\beta$: -5.554, -6.407, -5.094, -3.550, -5.094, -5.555
- $\gamma$: 6.490, 5.638, 4.586, 3.380, 4.586, 5.151
- $\delta$: $2.0 \times 10^{-4}$, $5.0 \times 10^{-4}$, $1.0 \times 10^{-6}$, $2.0 \times 10^{-3}$, $1.0 \times 10^{-6}$, $1.0 \times 10^{-5}$
- $\epsilon$: 2.857, 3.333, 8.000, 2.000, 8.000, 6.667

### Table 8: The best solutions for cost and emission from the Pareto front based on four multiobjective algorithms.

<table>
<thead>
<tr>
<th></th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_1$</td>
<td>$f_2$</td>
<td>$f_1$</td>
<td>$f_2$</td>
</tr>
<tr>
<td>PG1</td>
<td>12.36</td>
<td>41.30</td>
<td>11.09</td>
<td>7.92</td>
</tr>
<tr>
<td>PG2</td>
<td>29.99</td>
<td>45.41</td>
<td>30.16</td>
<td>42.08</td>
</tr>
<tr>
<td>PG3</td>
<td>55.40</td>
<td>55.31</td>
<td>57.50</td>
<td>60.89</td>
</tr>
<tr>
<td>PG4</td>
<td>105.16</td>
<td>43.40</td>
<td>105.41</td>
<td>101.06</td>
</tr>
<tr>
<td>PG5</td>
<td>45.26</td>
<td>52.79</td>
<td>45.39</td>
<td>42.61</td>
</tr>
<tr>
<td>PG6</td>
<td>38.01</td>
<td>52.01</td>
<td>37.81</td>
<td>33.88</td>
</tr>
<tr>
<td>$f_1$ (cost)</td>
<td>606.52</td>
<td>639.09</td>
<td>607.18</td>
<td>645.01</td>
</tr>
<tr>
<td>$f_2$ (emission)</td>
<td>0.2255</td>
<td>0.1931</td>
<td>0.2259</td>
<td>0.2009</td>
</tr>
</tbody>
</table>

### Table 9: The best compromise solutions for emission and cost using different multiobjective algorithms.

<table>
<thead>
<tr>
<th></th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_1$</td>
<td>$f_3$</td>
<td>$f_1$</td>
<td>$f_3$</td>
</tr>
<tr>
<td>PG1</td>
<td>24.490</td>
<td>36.201</td>
<td>24.313</td>
<td>32.99</td>
</tr>
<tr>
<td>PG2</td>
<td>38.99</td>
<td>45.41</td>
<td>30.16</td>
<td>42.08</td>
</tr>
<tr>
<td>PG3</td>
<td>55.40</td>
<td>55.31</td>
<td>57.50</td>
<td>60.89</td>
</tr>
<tr>
<td>PG4</td>
<td>105.16</td>
<td>43.40</td>
<td>105.41</td>
<td>101.06</td>
</tr>
<tr>
<td>PG5</td>
<td>45.26</td>
<td>52.79</td>
<td>45.39</td>
<td>42.61</td>
</tr>
<tr>
<td>PG6</td>
<td>38.01</td>
<td>52.01</td>
<td>37.81</td>
<td>33.88</td>
</tr>
<tr>
<td>$f_1$ (cost)</td>
<td>616.790</td>
<td>627.011</td>
<td>55.40</td>
<td>67.53</td>
</tr>
<tr>
<td>$f_2$ (emission)</td>
<td>0.2015</td>
<td>0.2033</td>
<td>0.1992</td>
<td>0.2590</td>
</tr>
</tbody>
</table>

### Table 10: The best solutions for cost and loss from the Pareto front based on four multiobjective algorithms.

<table>
<thead>
<tr>
<th></th>
<th>CMOABC</th>
<th>MOABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_1$</td>
<td>$f_3$</td>
<td>$f_1$</td>
<td>$f_3$</td>
</tr>
<tr>
<td>PG1</td>
<td>15.13</td>
<td>3.69</td>
<td>9.62</td>
<td>4.02</td>
</tr>
<tr>
<td>PG2</td>
<td>29.92</td>
<td>27.41</td>
<td>32.16</td>
<td>11.24</td>
</tr>
<tr>
<td>PG3</td>
<td>55.89</td>
<td>105.93</td>
<td>57.99</td>
<td>109.92</td>
</tr>
<tr>
<td>PG4</td>
<td>112.66</td>
<td>57.01</td>
<td>95.41</td>
<td>67.53</td>
</tr>
<tr>
<td>PG5</td>
<td>31.23</td>
<td>5.23</td>
<td>44.38</td>
<td>11.64</td>
</tr>
<tr>
<td>PG6</td>
<td>38.95</td>
<td>87.56</td>
<td>39.85</td>
<td>80.63</td>
</tr>
<tr>
<td>$f_1$ (loss)</td>
<td>2.4514</td>
<td>1.596</td>
<td>2.4643</td>
<td>1.5489</td>
</tr>
</tbody>
</table>
First, two competing objectives, fuel cost and emission, are considered. From the Pareto-optimal solution shown in Figure 12, it is clear that the proposed CMOABC method is able to give well distributed solutions compared to the other three algorithms. From Table 8, it is clear that the minimum fuel cost obtained by CMOABC is $606.52$/h, which is better than the results optimized by the other three algorithms. And when the best fuel cost effect case is considered, CMOABC method can provide smaller loss. The best compromise solution is $616.7902$/hand $0.2015$ ton/h of CMOABC which can be observed in Table 9. It can once again prove that the proposed method has obtained better solutions.

The cost loss objective functions are shown in Figure 13 and Table 10. From the figure, we can see that the proposed algorithm obtains better results than the other three algorithms, though the obtained Pareto fronts are not evenly distributed. From Table 10, it is easily observed that CMOABC method can provide lower economic fuel cost when the minimum loss case is considered and smaller loss when the best fuel cost effect case is considered.

For emission loss objective functions, we can observe that the algorithms achieve similar performance ranking as for fuel cost emission objective functions.

4.3.2. Case II: Three-Objective OPF Optimization. In this case, the three objectives are optimized simultaneously by the four algorithms and the corresponding best solutions are given in Table 14. Figure 15 also shows the result values of three competing objectives.

As seen in Figure 15, compared to the other three algorithms, it can be observed that the Pareto-optimal solutions obtained by the proposed CMOABC are better distributed on the front with good diversity. Among the other three algorithms, the Pareto-optimal solutions obtained by the standard MOABC are also well distributed and the diversification of them is not as good as the ones obtained by the proposed CMOABC.

Furthermore, from Table 14, it can be also concluded that the CMOABC is able to discover a well distributed and diverse solution set for this three-objective problem.
However, the other three algorithms cannot archive the true Pareto front for the three-objective OPF problem.

The compromise solutions are shown in Table 15. From this table we can see the proposed algorithm obtains the best emission, cost, though the system loss obtained by CMOABC is 2.1601 MW, which is a little more than 2.1598 MW obtained by MOABC.

5. Conclusions

An improved multiobjective ABC algorithm based on K-means cluster, called CMOABC, is proposed in this paper. In the proposed algorithm, we first modify the employed bees' phase of the original ABC to improve its convergence rate. The proposed algorithm adopts K-means clustering...
method to partition the population into many clusters and the number of the clusters is changing to implement information exchange among the different clusters. With testing against a set of 6 mathematical benchmark functions (including both two- and three-objective cases), CMOABC has a better performance than the other MOPSO, MOABC, and NSGA-II.

Then, the proposed algorithm is used to handle multi-objective OPF problem, and 30-bus IEEE test system is adopted to test the proposed algorithm. By comparing the simulation results of CMOABC, MOABC, MOPSO, and NSGA-II, the proposed method is able to give well distributed Pareto-optimal solutions compared to the other three methods for OPF problem with different objectives.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**Acknowledgment**

This research is partially supported by the Key Research Program of the Chinese Academy of Sciences Grant no. KGZD-EW-302.

**References**


