Biparty Multiobjective Optimal Power Flow: the Problem Definition and an Evolutionary Approach

Yatong Chang[a], Wenjian Luo[a,b,*], Xin Lin[c], Zhen Song[a], Carlos A. Coello Coello[d,e]

Email: 20s151150@stu.hit.edu.cn, luowenjian@hit.edu.cn, xlin@nuist.edu.cn, 21s151097@stu.hit.edu.cn, carlos.coellocoello@cinvestav.mx

[a]*Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, Guangdong, China*
[b]*Peng Cheng Laboratory, Shenzhen 518055, Guangdong, China*
[c]*School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing 210044, Jiangsu, China*
[d]*Departamento de Computación, CINVESTAV-IPN, Mexico City, México*
[e]*School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey, Nuevo León, México*

Abstract

The multiobjective optimal power flow (MOOPF) problem consists of adjusting the generator power and the voltage state of each node within the feasible range in the process of power transmission and, finally of achieving the objectives of optimizing the cost, loss and stability, etc. In the MOOPF, two key decision makers are usually involved, which are the power generation sector and the transmission sector. Thus, it is more suitable to model an MOOPF as a biparty multiobjective optimal power flow (BPMOOPF) problem. However, so far, there is no work on treating and solving the MOOPF problem from the perspective of biparty multiobjective optimization. In this paper, we propose the definition of the BPMOOPF problem as well as a novel evolutionary biparty multiobjective optimization algorithm for solving the BPMOOPF problem, which we call BPMOOPF-EA. Our experimental results show that, compared two state-of-the-art algorithms (C-MOEA/D and A-NSGA-III), our proposed BPMOOPF-EA has a better performance when solving the BPMOOPF problem.

*Corresponding author

Keywords: Multiobjective optimization, multiparty multiobjective optimization, multiobjective evolutionary algorithm based on decomposition, optimal power flow

Abbreviations

The following summarizes the meanings of abbreviations and acronyms used throughout the paper.

A-NSGA-III	Adaptive-NSGA-III
BPMOOPF	Biparty Multiobjective Optimal Power Flow
BPMOP	Biparty Multiobjective Optimization Problem
C-MOEA/D	Constraint-MOEA/D
DE	Differential Evolution
DM	Decision Maker
EA	Evolutionary Algorithm
ESDE-MC	Enhanced Self-adaptive Differential Evolution with Mixed Crossover
HV	Hypervolume Indicator
IUDE	Improved Unified Differential Evolution
MOEA/D	Multiobjective Evolutionary Algorithm based on Decomposition
MOOPF	Multiobjective Optimal Power Flow
MOP	Multiobjective Optimization Problem
MPHV	Multiparty Hypervolume Indicator
MPMOP	Multiparty Multiobjective Optimization Problem
OPF	Optimal Power Flow
MOOPF	Multiobjective Optimal Power Flow
SBX	Simulated Binary Crossover
SLFA	Shuffle Frog Leaping Algorithm

1. Introduction

Optimal power flow (OPF) is a widely studied problem in the field of power systems [1]. Under conditions that include the limits of the active and reactive power of the generator and line flow, the algorithms for OPF ofen seek to find solutions to minimize objectives such as: generation costs, transmission losses, and voltage deviation [2]. Thus, an OPF problem usually involves multiple objectives which should be optimized simultaneously and this is often regarded as a multiobjective optimization problem (MOP).

Many evolutionary algorithms (EAs) have been applied to solve multi-objective optimal power flow (MOOPF) problems [1]. In [3], an improved version of the Shuffle Frog Leaping Algorithm (SLFA), which combines a mutation srategy, is proposed to solve MOOPF problems. Pulluri *et al.* [4] proposed an Enhanced Self-adaptive Differential Evolution with Mixed Crossover (ESDE-MC) approach for solving MOOPF problems, which showed its effectiveness in the IEEE 30-bus, the IEEE 57-bus and the Algerian 59-bus systems. By combining Pareto ordering and fuzzy computation, Shaheen *et al.* [5] improved the Differential Evolution (DE) operator, so that it could speed up the population's convergence, and showed its effectiveness in solving MOOPF problems. Biswas *et al.* [6] applied a multiobjective evolutionary algorithm based on decomposition (MOEA/D) to solve MOOPF problems. Naderi *et al.* [7] proposed a hybrid self-adaptive heuristic algorithm to address the MOOPF problem. Karthik *et al.* [8] proposed a MOOPF solution using a new heuristic optimization algorithm. Li *et al.* [9] formulated the optimal power flow with stochastic wind and solar energy as a multi-objective optimization problem and presented a multi-objective evolutionary algorithm based on non-dominated sorting with the constraint handling technique to solve it. Kahraman *et al.* [10] developed a method using the Pareto archiving approach based on crowding distance to address the MOOPF problem. Ganesan *et al.* [11] provide a concise review on recent implementations of multiobjective and multilevel optimization on sustainable energy economic systems.

Existing work designs different algorithms to solve the OPF problem and achieves remarkable results. To the best of our knowledge, however, these algorithms have been designed from the perspective of a single decision maker (DM) but fail to take multi-decision maker scenarios. In reality, there are different decision makers (sectors) which focus on different objectives [12]. Thus, the OPF problem needs to be optimized from various perspectives, including power generation and transmission. For the power generation sector, the power generation cost should be the main concern. Meanwhile, because stochastic wind and solar power are often added to the power generation system, the uncertainty of the renewable power could lead to the instability of the power system, which is often accompanied with serious consequences. Therefore, voltage stability is an important objective that the power generation sector should also consider [13]. Additionally, how to reduce the losses in the transmission process, including active power loss [14] and reactive power loss [15], are two crucial objectives for the transmission sector.

Multiparty multiobjective optimization problems (MPMOPs) [12] are a particular class of multiobjective optimization problems (MOPs) [16] in which more than one decision maker (DM) is involved. In an MPMOP, each DM focuses on his/her preferred objectives but does not care about the preferences of the other DMs. Moreover, when solving MPMOPs, not only all the objectives should be considered, but also the objectives of each DM should be optimized (in a Pareto sense) as much as possible. For example, there is a MPMOP with two DMs, each of whom has two maximization objectives. Given two solutions x_1 and x_2, let's assume that $F_1(x_1) = (10, 10)$ and $F_1(x_2) = (100, 100)$ are objective values for the first DM on x_1 and x_2. Similarly, $F_2(x_1) = (1, 2)$ and $F_2(x_2) = (2, 1)$ are objective values for the second DM. If we consider optimizing all objectives, we find that $(10, 10, 1, 2)$ for x_1 and $(100, 100, 2, 1)$ for x_2 are equally good according to Pareto dominance relationship. But in fact, x_2 is significantly better than x_1 for the first DM, and this is where an MPMOP comes into play. When the number of DMs is two, MPMOPs are often called biparty multiobjective optimization problems (BPMOPs). Thus, an MOOPF problem is more suitable to be modeled as a BPMOP. So far, however, there has been little work on solving MOOPF problems from the perspective of biparty multiobjective optimization.

In this paper, to model an MOOPF as a BPMOP, two sectors are discussed: the power generation sector and transmission sector. Each of them focuses on different objectives, and all the objectives from these two sectors form the biparty multiobjective optimal power flow (BPMOOPF). To solve the BPMOOPF, it is not appropriate to directly apply traditional multiobjective optimization algorithms, because those algorithms ignore the existence of two or more DMs. State-of-the-art multiparty multibjective optimization algorithms (e.g., OptMPNDS [12], OptMPNDS2 [17] and MOEA/D-MP [18]), are also not suitable for directly solving BPMOOPF problems. The main reason is that they lack the ability to deal with a large number of constraints which is something common in real-world problems such as the BPMOOPF problem.

Evolutionary algorithms have been widely applied to solve MOPs, and most of them are based on either Pareto dominance, a quality indicator or decomposition [16, 19, 20]. These multiobjective evolutionary algorithms (MOEAs) have shown effectiveness and efficiency in solving complex MOPs having two and three objectives. However, when dealing with more than three objectives (the so-called many-objective optimization problems, or MaOPs [19]) the performance of Pareto dominance based MOEAs quickly degrades

as the number of objectives increases due to the exponential increase in the number of nondominated solutions [20]. Regarding indicator-based MOEAs, their computational cost significantly increases (normally in an exponential manner, if using the hypervolume indicator) with a higher number of objectives [20]. In contrast, decomposition-based MOEAs (e.g., MOEA/D [21]) keeps a good performance and a reasonable computational cost when increasing the number of objectives. That is the reason for which we selected MOEA/D as our baseline algorithm for developing our proposed approach called BPMOOPF-EA.

Due to the complexity of the BPMOOPF problem, BPMOOPF-EA adopts a different way to generate weight vectors from that provided in the original MOEA/D [21], and uses the combination of three DE operators in the Improved Unified Differential Evolution (IUDE) [22] to reproduce new candidates. In addition, the Superiority of Feasible Solutions (SFS) constraint handling rule [23] is used in BPMOOPF-EA to guide the population towards the feasible region. BPMOOPF-EA is compared with two state-of-the-art constrained multiobjective evolutionary optimization algorithms: Constraint-MOEA/D (C-MOEA/D) [24] and Adaptive-NSGA-III (A-NSGA-III) [24]. Our experimental results show that our proposed BPMOOPF-EA outperforms the other two approaches.

In summary, the contributions of this paper are the following:

- For the first time, the traditional multiobjective optimal power flow problem is solved from the perspective of biparty multiobjective optimization. In the BPMOOPF problem, there are two sectors: the power generation sector and the transmission sector. The power generation sector pays more attention to the overall cost of generating electricity and voltage quality, while the power transmission sector is more concerned about the line losses of the network.

- A novel biparty multiobjective optimization algorithm based on MOEA/D (BPMOOPF-EA), is proposed here to solve the BPMOOPF problem. In our proposed BPMOOPF-EA, in order to deal with both parties in the BPMOOPF problem, the weight vectors for decomposition are adjusted accordingly. Meanwhile, the reproduction operators in IUDE are adopted to efficiently generate new candidate solutions, and the SFS rule is adopted to handle multiple constraints. Experimental results show that our proposed BPMOOPF-EA has a better performance than two state-of-the-art algorithms.

The remainder of this paper is organized as follows. Section 2 introduces the previous related work. Section 3 introduces the BPMOOPF problem. Section 4 describes our proposed BPMOOPF-EA in detail. Section 5 contains our experimental results and their corresponding discussion. Finally, our conclusions and some possible paths for future research are provided in Section 6.

2. Related Work

2.1. Multiparty Multiobjective Optimization

Multiparty multiobjective optimization problems (MPMOPs) [12] are a particular type of multiobjective optimization problems (MOPs) [16] having multiple decision makers.

A MOP is formally defined as follows[1]:

$$\text{minimize } f(x) := [f_1(x), f_2(x), \ldots, f_k(x)] \tag{1}$$

subject to:

$$g_i(x) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(x) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $x = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$ are the objective functions and $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

A few additional definitions are required to introduce the notion of optimality used in multiobjective optimization:

Definition 1. Given two vectors $x, y \in \mathbb{R}^k$, we say that $x \leq y$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that x **dominates** y (denoted by $x \prec y$) if $x \leq y$ and $x \neq y$.

Definition 2. We say that a vector of decision variables $x \in \mathcal{X} \subset \mathbb{R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $x' \in \mathcal{X}$ such that $f(x') \prec f(x)$.

[1]Without loss of generality, we will assume only minimization problems.

Definition 3. We say that a vector of decision variables $x^* \in \mathcal{F} \subset \mathbb{R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is nondominated with respect to $\mathcal{F}$.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}$ is defined by:

$$\mathcal{P} = \{x \in \mathcal{F} | x \text{ is Pareto-optimal}\}$$

Definition 5. The **Pareto Front** $\mathcal{PF}$ is defined by:

$$\mathcal{PF} = \{f(x) \in \mathbb{R}^k | x \in \mathcal{P}\}$$

Therefore, our aim is to obtain the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3).

MPMOPs were recently proposed by Liu *et al.* [12]. An MPMOP usually involves multiple decision makers (DMs). Each DM represents a party, and at least one of the DMs focuses on multiple objectives, which conflict with each other. An MPMOP can be defined as follows [17]:

$$\begin{aligned} &Minimize\ E(x) = (F_1(x), \ldots, F_M(x)), \\ &where \begin{cases} F_1(x) = (f_{1,1}(x), \ldots, f_{1,m_1}(x)), \\ \qquad\qquad \vdots \\ F_i(x) = (f_{i,1}(x), \ldots, f_{i,m_i}(x)), \\ \qquad\qquad \vdots \\ F_M(x) = (f_{M,1}(x), \ldots, f_{M,m_M}(x)), \end{cases} \end{aligned} \tag{4}$$

where $E(x)$ is a minimization MPMOP, for which the number of DMs is M. For the i^{th} DM who focuses on $F_i(x)$ ($i \in \{1, \ldots, M\}$), there are m_i objectives to be optimized simutaneously.

To find the best solutions for an MPMOP, it is required to make that the solutions reach the Pareto fronts of all parties as much as possible. If one solution x^{MP} meets the requirements of the best solutions for an MPMOP, $\nexists x \in \mathcal{F}$ satisfies the following two conditions simultaneously:

- For all parties, $x \prec x^{MP}$, or they do not dominate each other;
- For at least one party, $x \prec x^{MP}$.

2.2. The Optimal Power Flow Problem

In the optimal power flow (OPF) problem we seek to optimize the operation of the power system under the physical constraints of power law and engineering constraints [25]. Common optimization objectives include power generation cost, voltage stability, active power loss, and reactive power loss, which are listed as follows [26]:

$$Minimize\ f_1 = \sum_{i=1}^{N} \left(a_i + b_i P_{g,i} + c_i P_{g,i}^2\right) \tag{5}$$

$$Minimize\ f_2 = \sum_{i=1}^{N} \left(1 - |V_i|\right)^2 \tag{6}$$

$$Minimize\ f_3 = \sum_{i=1}^{N} \left(P_{g,i} - P_{l,i}\right) \tag{7}$$

$$Minimize\ f_4 = \sum_{i=1}^{N} \left(Q_{g,i} - Q_{l,i}\right) \tag{8}$$

where f_1 is the total cost of power generation, in which a_i, b_i and c_i are the cost factors for the i^{th} bus generator and $P_{g,i}$ is the output active power of the i^{th} bus generator. f_2 indicates the voltage deviation of each bus, which is an important measure of voltage stability. V_i is the bus voltage of the i^{th} bus, and N represents the total number of buses in the power system. f_3 and f_4 are the active power loss and reactive power loss, respectively. $P_i\,(= P_{g,i} - P_{l,i})$ represents the active injection power on the i^{th} bus, in which $P_{l,i}$ is the active power load of the i^{th} bus. $Q_i\,(= Q_{g,i} - Q_{l,i})$ represents the reactive injection power on the i^{th} bus, in which $Q_{g,i}$ and $Q_{l,i}$ are the output reactive power of the i^{th} bus and the reactive power load of the i^{th} bus, respectively.

In addition, the OPF problem has multiple constraints, including inequality constraints and equality constraints, as shown below:

$$P_{g,i} - P_{l,i} = V_i \sum_{j=1}^{N} V_j \left(G_{ij} \cos\left(\delta_i - \delta_j\right) + B_{ij} \sin\left(\delta_i - \delta_j\right)\right), i = 1, 2, \ldots, N \tag{9}$$

$$Q_{g,i} - Q_{l,i} = V_i \sum_{j=1}^{N} V_j \left(G_{ij} \sin\left(\delta_i - \delta_j\right) - B_{ij} \cos\left(\delta_i - \delta_j\right)\right), i = 1, 2, \ldots, N \tag{10}$$

$$V_{min} \leq V_k \leq V_{max}, k = 1, 2, \ldots, N \tag{11}$$

$$\delta_{min} \leq \delta_k \leq \delta_{max}, k = 1, 2, \ldots, N \tag{12}$$

$$P_{min} \leq P_{g,k} \leq P_{max}, k = 1, 2, \ldots, N \tag{13}$$

$$Q_{min} \leq Q_{g,k} \leq Q_{max}, k = 1, 2, \ldots, N \tag{14}$$

where $Ybus_{ij} (= G_{ij} + 1jB_{ij})$ is the ij^{th} element of the admittance matrix, in which G_{ij} and B_{ij} are the transfer conductance and susceptance between buses i and j; δ_i and δ_j are the voltage angles of buses i and j, respectively. Equations (9) and (10) are the power flow equations in the network, which are also the equality constraints that the problem must meet. Equations (11)-(14) are the ranges of each dimension of the decision variables, which are also the inequality constraints of the OPF problem.

2.3. MOEA/D and its Improvements

The multiobjective evolutionary algorithm based on decomposition (MOEA/D) [21], which was originally proposed by Zhang and Li, has been widely studied in recent years. MOEA/D decomposes an MOP into several subproblems using a set of uniformly distributed vectors, whose size equals the size of the population. Each of these vectors corresponds to one (or few) individual in the population. In addition, the concept of "neighbor" is emphasized in MOEA/D. Neighbors are determined by Euclidean distances between vectors. For a vector, T vectors (including itself) with the smallest Euclidean distances are selected as its neighbors. For each individual, its offspring is generated through the parents selected from its neighbors. The offspring will be compared with all the neighbors of the corresponding individual, and the best individuals will enter the next generation. The comparison relies on scalar g values, which are calculated by the corresponding vectors. For minimization MOPs, the smaller the g value, the more likely the individual will remain in the population.

Multiple decomposition approaches, including the weighted sum approach [27], the Tchebycheff approach [27] and the penalty-based boundary intersection (PBI) approach [21] have been proposed. The Tchebycheff decomposition approach, which is used in our proposed BPMOOPF-EA is illustrated in equation (15).

$$\begin{aligned} &Minimize\ g^{Te}(x|\boldsymbol{\lambda}, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i^*|\}, \\ &Subject\ to\ x \in \Omega, \end{aligned} \tag{15}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^T$ is a generated weight vector, which satisfies that $\sum_{i=1}^{m} \lambda_i = 1$, and m is the dimensionality of the objective space. The value of each dimension of the reference point $\boldsymbol{z}^*$ represents the minimum value of the objective value of all individuals in this dimension.

A wide variety of improved versions of MOEA/D have been proposed so far [28]. For example, in [29], Li *et al.* proposed MOEA/D-DE for MOPs having complicated PS shapes. The main difference between MOEA/D and MOEA/D-DE is in the mechanism used to generate offspring, where the DE operator replaces the simulated binary crossover (SBX) operator from the original MOEA/D. In [30], Qi *et al.* applied the adaptive weight adjustment strategy on MOEA/D and named it MOEA/D-AWA. After the population converges to a certain extent, MOEA/D-AWA replaces the subproblems which have low sparsity in the population with the subproblems having high sparsity in the external population to adjust the weight vectors. In [31], Wang *et al.* adopted both the SBX operator and the DE operator, and combined the adaptive strategy to improve the diversity when the population has converged to a certain extent. In [32], Chen *et al.* embedded a resource allocation strategy, a multioperator and multiparameter strategy and a bidirectional local search strategy into MOEA/D (this variant was called MOEA/D-RML) to solve scalable multi/many-objective problems.

3. Biparty Multiobjective Optimal Power Flow

As mentioned in Section 1, OPF problems with multiple/many objectives are often solved from the perspective of one decision maker, i.e., one party. However, an OPF problem usually involves more than one decision maker at the same time. Thus, an MOOPF problem is often suitable to be modeled as an MPMOP.

In this paper, we model the MOOPF with the four objectives listed in Section 2.2 as the biparty multiobjective optimal power flow (BPMOOPF). The BPMOOPF is defined as follows:

$$Minimize\ E_{OPF} = (F_1, F_2) \tag{16}$$

where

$$F_1 = (f_{1,1}, f_{1,2}) \tag{17}$$

$$F_2 = (f_{2,1}, f_{2,2}) \tag{18}$$

$$f_{1,1} = \sum_{i=1}^{N} \left(a_i + b_i P_{g,i} + c_i P_{g,i}^2\right) \tag{19}$$

$$f_{1,2} = \sum_{i=1}^{N} \left(1 - |V_i|\right)^2 \tag{20}$$

$$f_{2,1} = \sum_{i=1}^{N} \left(P_{g,i} - P_{l,i}\right) \tag{21}$$

$$f_{2,2} = \sum_{i=1}^{N} \left(Q_{g,i} - Q_{l,i}\right) \tag{22}$$

The above formulas mean that the BPMOOPF problem involves two decision makers, i.e., the power generation sector and the transmission sector. Each of them focuses on different objectives. The power generation sector focuses on $f_{1,1}$ and $f_{1,2}$, i.e., the total generation cost and voltage stability. In contrast, the transmission sector mainly focuses on $f_{2,1}$ and $f_{2,2}$, i.e., the active power loss and the reactive power loss. Additionally, the BPMOOPF problem also includes multiple constraints, which are those provided in equations (9)-(14).

To solve the BPMOOPF problem, we need to find solutions as close as possible to the Pareto front of all parties while satisfying the constraints. In this paper, a novel algorithm based on MOEA/D is proposed to solve this problem. This algorithm will be described in the next section.

4. Proposed Algorithm

In this section, in order to solve the BPMOOPF problem, a novel algorithm, called BPMOOPF-EA, which is an improved version of MOEA/D, is proposed.

4.1. Main Framework

Algorithm 1 shows the framework of BPMOOPF-EA, which is explained as follows.

(1) First, a population P with size N is randomly initialized.

(2) A set of $\sqrt[M]{N}$ weight vectors is generated for each decision maker. Here, M is the number of parties.

(3) A set of N new weight vectors W are generated, and T neighbors of each weight vector are determined according to Euclidean distances. The details will be provided in Section 4.2.

(4) The objective values and constraint violations of all individuals in the initial population P are calculated. Because BPMOOPF-EA adopts Tchebycheff decomposition, it is necessary to calculate the objective values of all individuals in advance, and take the minimum value of each objective of all individuals as the reference point.

(5) The external population EP is initialized to $\emptyset$, and gen, which is used to record the number of generations, is initialized to 0.

(6) For each DE operator, it is necessary to initialize its memory sets M_{CR} and M_F, and winning sets S_{CR} and S_F, which will be further explained in Section 4.4.2.

(7) For the main loop, while the termination condition is not met, the following steps are executed:

- First, we add 1 to gen, and the $Remove\_Vector$ operator, which is used to periodically delete poor individuals, is executed. The details of the $Remove\_Vector$ operator will be explained in Section 4.3.

- For each individual in the current population P, three offspring are generated according to three DE operators, and the best individual y_{best} is selected according to the SFS constraint rule [23]. The details of the operators will be provided in Section 4.4.1.

- The reference point is updated.

- Each of the three offspring is compared with its parent according to the SFS constraint rules, and its winning sets S_{CR} and S_F are updated.

- Then y_{best} is compared with each neighbor of x using the $Comparator$ operator. The details of the $Comparator$ operator will be depicted in Section 4.4.4.

- If the size of EP exceeds the threshold, the individuals in EP will be randomly deleted until the number of individuals in EP meets the requirement.

- The memory sets M_{CR} and M_F of each DE operator are updated.

4.2. Weight Vector Generation

MOEA/D [21] has been widely used to solve MOPs, but it cannot be directly applied to solve BPMOPs. In order to solve BPMOPs using MOEA/D, it is necessary to improve its weight vector generation strategy. Here, a new weight vector generation strategy is proposed to make the generated subproblem more suitable for BPMOPs.

In MOEA/D, a set of weight vectors uniformly distributed in the objective space will be generated first, and the dimensionality of the vector is the same as the number of objectives. This set of weight vectors decomposes the multiobjective problem into several subproblems, and each subproblem assigns different weights to each objective through the weight vector.

Let's take the minimization MOP $F(x) = (f_1(x), \ldots, f_m(x))$ as an example, where m represents the number of objectives. According to the vector generation strategy of the original MOEA/D, and assuming that the population size is N, the generated set of vectors $\{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N\}$ contains the vector $\boldsymbol{\lambda}_1 = (1, 0, \ldots, 0)^T$, that is, only the weighted value of the first objective is 1, and the weighted values of other objectives are all zero. Thus, the subproblem corresponding to $\boldsymbol{\lambda}_1$ only focuses on the first objective. In the BPMOP, it is necessary to consider the requirements of both parties at the same time. Therefore, if the original vector generation strategy is directly used, many subproblems of weight vectors could not obtain the optimal solution of the BPMOP. For example, the above $\boldsymbol{\lambda}_1$ only considers the first objective of the first party, but does not consider any objectives of the second party.

Given that, a new weight vector generation strategy is proposed. First, a set of uniformly distributed weight vectors is generated for each party, and then the following operations are repeated: take each weight vector from the weight vector set of each party set in turn, and the vectors obtained each time are spliced together according to the order of the decision makers to form a new weight vector, which is used as the vector to decompose the BPMOP.

The following example illustrates the new vector generation strategy. Let's assume that $E_{BP} = (F_1, F_2)$ is a minimization BPMOP with two decision makers, $F_1 = (f_{1,1}, f_{1,2})$ indicates that the first party pays attention to the two objectives $f_{1,1}$ and $f_{1,2}$, and $F_2 = (f_{2,1}, f_{2,2})$ indicates that the second party pays attention to the two objectives $f_{2,1}$ and $f_{2,2}$. To simplify the description, let's assume that each party generates only 3 weight vectors,

Algorithm 1 BPMOOPF-EA

Input: population size N, BPMOOPF E_{OPF}
Output: Final population P

Randomly initialize the population P with size N;
Generate $\sqrt[M]{N}$ evenly distributed weight vectors for each DM, where M is the number of DMs;
Generate a set of N weight vectors W, and determine T neighbors of each vector;
Calculate the objective values and constraint violations of all individuals in P and generate the reference point;
Initialize external population $EP = \emptyset$ and variable gen=0;
for $i = \{1, 2, 3\}$ **do**
    Initialize $M_{CR,i}$, $M_{F,i}$, $S_{CR,i}$ and $S_{F,i}$;
end for
while the termination is **not** satisfied **do**
    $gen = gen$+1;
    [P, W, EP, N] = Remove_Vector(gen, N, P, W, EP);
    for each individual x **do**
        Generate offspring and select y_{best};
        Update the reference point;
        for $i = \{1, 2, 3\}$ **do**
            Compare the i^{th} offspring with x using the SFS rule;
            Update $S_{CR,i}$ and $S_{F,i}$;
        end for
        for each neighbor x_T of x **do**
            [P, EP] = Comparator(y_{best}, x_T, $\boldsymbol{\lambda}_{x_T}$, P, EP);
        end for
    end for
    while the size of EP exceeds the threshold **do**
        Randomly select an individual of EP and delete;
    end while
    for $i = \{1, 2, 3\}$ **do**
        Update memory set $M_{CR,i}$ and $M_{F,i}$;
    end for
end while

which are $\boldsymbol{\lambda}_1 = (1, 0)^T$, $\boldsymbol{\lambda}_2 = (0.5, 0.5)^T$ and $\boldsymbol{\lambda}_3 = (0, 1)^T$. Then, the vectors are recombined, and finally a set of non-repeated weight vectors containing 9 vectors is obtained, which are:

- $(1, 0, 1, 0)^T$, $(0.5, 0.5, 1, 0)^T$, $(0, 1, 1, 0)^T$,
- $(1, 0, 0.5, 0.5)^T$, $(0.5, 0.5, 0.5, 0.5)^T$, $(0, 1, 0.5, 0.5)^T$,
- $(1, 0, 0, 1)^T$, $(0.5, 0.5, 0, 1)^T$, $(0, 1, 0, 1)^T$.

Then, according to the generated vectors, the BPMOP is decomposed in a proper way.

4.3. Adaptive Weight Vector Removal

In BPMOOPF-EA, the vectors are widely distributed in objective space and guide the evolution of the population. However, some vectors could seriously deviate from the feasible region. Obviously, allocating computing resources to such vectors is a waste of time. Therefore, it is necessary to delete the above vectors dynamically during the calculation process.

In [33], Zhang *et al.* introduced a strategy for allocating computer resources to different subproblems, which is based on the decrease rate of the fitness values. Here, the decrease rate of the fitness value is calculated as follows.

$$\Delta^i = \frac{old\ fitness\ value - new\ fitness\ value}{old\ fitness\ value}, \tag{23}$$

where $new\ fitness\ value$ represents the current objective value of the individual i, and $old\ fitness\ value$ represents its objective value before several generations. Then, the value of π^i is updated according to equation (24), and π^i can be used as a reference for the priority of assigning the computational resources to the i^{th} individual. The larger the value of π^i, the higher the rate at which the individual's objective value has descended at the previous generations. This also means that the subproblem is more likely to find an optimal solution and should be prioritized in allocating computer resources.

$$\pi^i = \begin{cases} 1 & \text{if } \Delta^i > 0.001 \\ \left(0.95 + 0.05 * \frac{\Delta^i}{0.001}\right) \pi^i & \text{otherwise} \end{cases} \tag{24}$$

We made minor changes to the above formulas, and used them as a reference for dynamically deleting vectors. The *new fitness value* and *old fitness value* in equation (23) are replaced by *new constraint violation* and *old constraint violation*, respectively, as shown in equation (25). Therefore, π_{CV}^{i} in equation (26) represents the decrease rate of constraint violations during several generations. An individual with large Δ_{CV} can be considered to have a greater chance of evolving into the feasible region and needs to be preserved. In addition, Δ_{CV} of an individual whose constraint violation is reduced to 0 should always be set to 1. Obviously, π_{CV} of such an individual will always be 1. π_{CV}^{i} is used as a reference for the priority that the i^{th} individual is retained, and the initial value is set to 1. The larger the value of π_{CV}^{i}, the more likely is that the i^{th} subproblem may find a feasible solution.

$$\Delta_{CV}^{i} = \frac{old\ constraint\ violation - new\ constraint\ violation}{old\ constraint\ violation} \tag{25}$$

$$\pi_{CV}^{i} = \begin{cases} 1 & \text{if } \Delta_{CV}^{i} > 0.001 \\ \left(0.95 + 0.05 * \frac{\Delta_{CV}^{i}}{0.001}\right) \pi_{CV}^{i} & \text{otherwise} \end{cases} \tag{26}$$

Based on the above, the $Remove\_Vector$ operator is designed to periodically remove vectors in BPMOOPF-EA. Algorithm 2 shows the process of $Remove\_Vector\_Operator$. Both the Δ_{CV} and the π_{CV} values of each individual are updated after each time interval T_{itl}, and the calculation method is shown in equations (25) and (26). Then, after the population has converged to a certain extent, all individuals are sorted in descending order according to their π_{CV} values. The latter N_{del} individuals in population P and their corresponding vectors in the set of weight vectors W will be removed. Removed individuals are then added to the external population EP. After repeating this operation several times, if the population size decreases to N_{final}, it will not be executed again.

4.4. Offspring Generation

4.4.1. Generating Offspring by Combining Multiple DE Operators

The original MOEA/D adopted simulated binary crossover (SBX) and polynomial-based mutation [21] to generate new candidate solutions. In [29], Li *et al.* pointed out some shortcomings of using SBX to solve complex optimization problems, including that it is easy to make the population converge

Algorithm 2 Remove_Vector

Input: gen, N, P, W, EP
Output: P, W, EP, N

if gen is an integer multiple of T_{itl} and $N > N_{final}$ **then**
    Update Δ_{CV} and π_{CV} of each individual;
    if $gen > G_{remove}$ **then**
        Sort all individuals' π_{CV} values in descending order;
        Remove the last N_{del} individuals in P;
        Remove the corresponding vectors in W;
        Add the removed N_{del} individuals to EP;
        $N = N\text{-}N_{del}$;
        Recalculate the nearest neighbors of each remaining vector;
    end if
end if

prematurely at the early stages of the evolutionary process, since SBX tends to produce a loss of diversity and also tends to generate low-quality solutions. Therefore, Li *et al.* proposed to adopt the differential evolution (DE) operator to replace the SBX operator, and achieved good results in solving problems with complex Pareto solution sets.

In [22], the IUDE algorithm proposed by Trivedi *et al.* obtained good results in solving single-objective constrained optimization problems. The reproduction strategy of IUDE consists in generating three offspring by using three different DE operators, then selecting the optimal individual by the SFS constraint rule, and comparing the selected optimal offspring with the parent by the εC constraint rule [34]. Finally, the best individual is retained in the population. The three DE operators are: DE/rand/1 mutation strategy with binomial crossover operator, DE/current-to-rand/1 mutation strategy without any crossover operator and DE/current-to-pbest/1 mutation strategy with binomial crossover operator. The first two DE operators could better maintain the population diversity and make the search space wider, while the latter DE operator shows better convergence. At different stages of the evolutionary process, the three DE operators could better guide the population to evolve towards the optimal solutions.

Because DE is free of restrictions on problem characteristics [35][36] and has the excellent properties described above. Our proposed BPMOOPF-EA

improves the original MOEA/D by referring to the reproduction strategy of IUDE. The process is shown in Figure 1, where DE1, DE2 and DE3 are the three DE operators mentioned above, and PM means polynomial-based mutation.

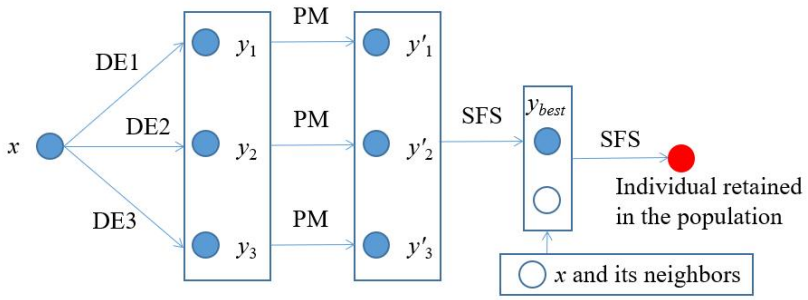

Figure 1: Improved offspring generation strategy

When an individual x generates offspring, three DE operators are used to generate three individuals y_1, y_2 and y_3, and polynomial-based mutation is applied on each of them. The formula of polynomial-based mutation [29, 37] is shown below.

$$u_i = \begin{cases} v_i + \sigma_i \times (x_i^{max} - x_i^{min}), & r_1 \leq p_m \\ v_i, & r_1 > p_m \end{cases} \tag{27}$$

with

$$\sigma_i = \begin{cases} [2r_2 + (1-2r_2)(1-\delta_1)^{\eta+1}]^{\frac{1}{\eta+1}} - 1, & r_2 \leq 0.5 \\ 1 - [2(1-r_2) + 2(r_2-0.5)(1-\delta_2)^{\eta+1}]^{\frac{1}{\eta+1}}, & r_2 > 0.5 \end{cases} \tag{28}$$

and

$$\delta_1 = \frac{v_i - x_i^{min}}{x_i^{max} - x_i^{min}}, \delta_2 = \frac{x_i^{max} - v_i}{x_i^{max} - x_i^{min}} \tag{29}$$

where $\mathbf{v} = (v_1, \ldots, v_d)$ is the offspring of the DE operator, v_i is the value of the i^{th} dimension of the offspring. $\mathbf{u} = (u_1, \ldots, u_d)$ is the offspring generated after applying polynomial-based mutation. x_i^{max} and x_i^{min} are the upper and lower bounds of x_i in decision space, respectively. The parameter η is called the distribution index, and p_m is the mutation rate. Both r_1 and r_2 are

randomly generated between 0 and 1. In addition, as shown in equations (28) and (29), the generation of σ_i refers to [37], which is different from [29]. In order to prevent the offspring from exceeding the boundary of decision space after applying polynomial-based mutation, the corresponding repair rules are set up. When a dimension of an individual exceeds the boundary, the dimension is repaired to the corresponding boundary value.

After applying polynomial-based mutation, the SFS constraint handling rule is used to select the optimal individual y_{best} from among y'_1, y'_2 and y'_3. Then, the selected optimal individual y_{best} is compared with respect to its T nearest neighbors using the SFS constraint rule. The winner will be retained in the population.

4.4.2. Adaptive Parameters

The DE operator that combines adaptive CR and F parameters was introduced in [38, 39]. For each individual, first a combination of CR_i and F_i is randomly selected from the memory sets of CR and F, i.e., M_{CR} and M_F. Then, CR_i and F_i work as the mean of the Normal distribution and the Cauchy distribution, where the standard deviation is 0.1, in order to generate random values for CR'_i and F'_i, respectively. CR'_i and F'_i will be used in the DE operator. If the offspring generated by the DE operator with CR'_i and F'_i is superior to the parent, CR'_i and F'_i are saved into S_{CR} and S_F, respectively. S_{CR} and S_F are the winning set of CR and F values. After all individuals have been generated and evaluated in one generation, the memory sets of CR and F (i.e., M_{CR} and M_F) are updated as follows:

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR}) & \text{if } S_{CR} \neq \emptyset \\ M_{CR,k,G} & \text{otherwise} \end{cases}, \tag{30}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_F) & \text{if } S_F \neq \emptyset \\ M_{F,k,G} & \text{otherwise} \end{cases}, \tag{31}$$

$$mean_{WA}(S_{CR}) = \sum_{t=1}^{|S_{CR}|} w_t \cdot S_{CR,t}, \tag{32}$$

$$mean_{WL}(S_F) = \frac{\sum_{t_1=1}^{|S_F|} w_{t_1} \cdot S_{F,t_1}^2}{\sum_{t_2=1}^{|S_F|} w_{t_2} \cdot S_{F,t_2}}, \tag{33}$$

$$w_t = \frac{\Delta f_t}{\sum_{u=1}^{|S_{CR}|} \Delta f_u}, \tag{34}$$

$$\Delta f_t = |f(\mathbf{y}^t) - f(\mathbf{x}^t)|, \tag{35}$$

where M_{CR} and M_F refer to the memory set of CR and F, respectively. The parameter k in the above formulas refers to a certain position in the memory set. During each generation, only the CR and F of the k^{th} position in the memory set will be updated. f_t is the function value or constraint violation of the individual related to the parameters in the t^{th} position of the winning set. Specifically, if the offspring $\mathbf{y}^t$ is better than its parent $\mathbf{x}^t$ due to the function value, f_t is calculated as the function value. But if the offspring $\mathbf{y}^t$ is better than its parent $\mathbf{x}^t$ due to the constraint violation, f_t represents the constraint violation.

In order to improve the search ability of our algorithm in the whole search space, we apply the above method [38, 39] to our proposed BPMOOPF-EA. For multiple DE operators, the strategy of adaptive parameters can be implemented by setting memory sets for each operator, respectively. It is worth noting that for the DE operator without any crossover operator, it is not necessary to set the memory set and the winning set of parameter CR.

4.4.3. Probabilistic Replacement Strategy

MOEA/D-DE [29] improves MOEA/D by maintaining the population's diversity during the evolutionary process. In the process of replacing the offspring, the maximum number of replacements is specified. That is, the replacement occurs at most n_r $(0 < n_r < T)$ times in the process of comparing the offspring with its T nearest neighbors of the parent. n_r ensures that not all nor most of the population will be replaced by one excellent offspring in a round of comparison.

Inspired by this, our proposed algorithm adopts the probability replacement strategy to ensure the population diversity in the evolutionary process. First, at the beginning of the algorithm, the replacement probability p_{rep} is set. When the selected offspring is better than the parent, the algorithm does not directly replace the parent with the offspring, but with a probability of p_{rep}.

4.4.4. Comparator

Comparator is used to compare two individuals and retain the best one in our proposed BPMOOPF-EA. Algorithm 3 shows the way in which *Comparator* works.

In Algorithm 3, given two individuals x and y, the g values of x and y are calculated by using the vector $\boldsymbol{\lambda}_x$ of x. In addition, the constraint violations of x and y are also calculated. The value of g is used as the fitness value in the SFS constraint rule for comparison. Under the SFS constraint handling rule, if y is better than x, and the randomly generated number $rand$ is less than or equal to the replacement probability p_{rep}, x in the population P is replaced by y and added to the external population EP. Otherwise, y is added to EP.

It should be noticed that individuals in EP have the probability to generate offspring during the evolution of the population. Therefore, adding x or y to EP could better maintain the diversity of the population.

Algorithm 3 Comparator

Input: x, y, $\boldsymbol{\lambda}_x$, P, EP
Output: P, EP

1: Calculate the g value g_x of x, and the g value g_y of y using $\boldsymbol{\lambda}_x$ and the constraint violations of x and y;
2: **if** y is better than x according to the SFS rule **then**
3: **if** $rand \leq p_{rep}$ **then**
4: Replace x with y in the population P, and add x to EP;
5: **else**
6: Add y to EP
7: **end if**
8: **else**
9: Add y to EP;
10: **end if**

4.5. Computational Complexity

In this section, the computational complexity of the proposed algorithm is analyzed. These analyzes are based on the time-consuming operation in Algorithm 1. Specifically, in line 3 of the algorithm, the T neighbors of each individual are calculated, and the time complexity of this operation is

$O(n^2)$, where n is the population size. In the loop, the *Remove_Vector* on line 11 is designed to periodically remove vectors that deviate from the feasible domain, and its time complexity is $O(n^2)$. In lines 12 to 22, the *comparator* operation is performed on T neighbors of each individual to compare two individuals and keep the better one, and its time complexity is $O(n * T)$. In summary, the time complexity of BPMOOPF-EA is $O(MaxGen * n^2)$, where MaxGen is the maximum evolutionary generations allowed.

5. Experiment

5.1. Benchmark

The benchmark problem in this paper comes from the 46^{th} problem, i.e., RWMOP46, which is included in a group of real-world constrained multiobjective optimization problems proposed in [26]. RWMOP46 is a multiobjective optimal power flow problem, and it contains four objectives, which have been described in Section 2.2. The parameters of RWMOP46 are designed based on the IEEE 14-bus system, in which generators are set at bus 1, 2, 3, 6 and 8.

Based on RWMOP46, we divided the objectives into two parties and modeled RWMOP46 as a BPMOOPF problem. The details are as shown in Section 3. In the source code of BPMOOPF, the number of decision makers M is set to 2, which means that the BPMOOPF problem includes 2 decision mekers. The first decision maker is the power generation sector, and the power transmission sector is the second decision maker. As for the objectives in the source code of BPMOOPF, f_1 represents the power generation cost as shown in equation (5), and f_2 represents the voltage deviation as shown in equation (6). f_3 and f_4 in the source code of BPMOOPF are the active and the reactive power loss shown in equations (7) and (8), respectively. The source code of the BPMOOPF problem and our proposed BPMOOPF-EA can be obtained from `https://github.com/MiLabHITSZ/2022ChangBPMOOPF-EA`.

5.2. Performance Assessment

Until now, the most popular performance measurement adopted to evaluate multiparty multiobjective optimization algorithms has been the multiparty IGD proposed in [12]. However, this measurement is not applicable to algorithms designed to solve the BPMOOPF problem. This is because the Pareto front of the OPF model [26] is unknown and, therefore, we cannot obtain a proper reference set for computing the multiparty IGD.

The hypervolume indicator (HV) [40] is a popular performance measurement for assessing convergence and spread of MOEAs. One of its main advantages is that it does not require the true Pareto front to be computed. Therefore, in this paper, a new performance measurement called the multi-party Hypervolume indicator (MPHV), is proposed for evaluating the efficiency of the algorithms which are used to solve the BPMOOPF problem.

Diffenent from traditional HV, in order to calculate MPHV, the HV results of multiple objectives for each decision maker are calculated separately and then, they are accumulated. As a result, MPHV is able to measure the acceptance of a solution to multi-party decision makers. For a minimization MOP $F(x) = (f_1(x), \ldots, f_m(x))$, when calculating the HV, the objective values are normalized as shown in equation (36).

$$\widehat{f}_i(x) = \begin{cases} \frac{f_i(x) - f_i^{best}}{f_i^{worst} - f_i^{best}} & \text{if } f_i^{worst} \neq f_i^{best} \\ f_i(x) - f_i^{best} & \text{otherwise} \end{cases}, \tag{36}$$

where $f_i(x)$ represents the function value of x on the dimension i, f_i^{best} denotes the optimal value for the objective values of all the individuals on dimension i and f_i^{worst} denotes the worst of the objective values of all the individuals on dimension i. $\widehat{f}_i(x)$ denotes the normalized function value of x on dimension i.

After normalization, the function values range within $[0, 1]$, and the reference point $(1.1, 1.1, ..., 1.1)$ is selected to calculate the HV [41, 42]. When using the HV, the larger the result, the better the performance of the algorithm.

5.3. Compared Algorithms

Two state-of-the-art constrained optimization MOEAs (Constraint-MOEA/D (C-MOEA/D) [24] and Adaptive-NSGA-III (A-NSGA-III) [24]), were selected to compare results with respect to our proposed approach. For obtaining the source code of C-MOEA/D and A-NSGA-III, please refer to PlatEMO [43].

C-MOEA/D is improved on the basis of MOEA/D, by adding the SFS constraint handling rule to deal with constrained optimization problems. A-NSGA-III is improved on the basis of NSGA-III [24] by incorporating the rule of adaptively adding or deleting reference vectors. In addition, before each generation, the population will be sorted according to the constraint violations for constrained multiobjective optimization problems.

5.4. Parameters Settings

Parameters are set as shown in Table 1. The population size is set to 10000, and the maximum number of fitness evaluations is set to 1.6×10^7. For each algorithm, 15 independent experiments are carried out, and non-repeated random number seeds were set in each experiment. Regarding the algorithms used in our comparative study (i.e., C-MOEA/D [24] and A-NSGA-III [24]), the above conditions remained the same.

Table 1: Parameters of BPMOOPF-EA, C-MOEA/D and A-NSGA-III

	BPMOOPF-EA	C-MOEA/D	A-NSGAIII
Population size N	10000	10000	10000
maximum fitness evaluations $MFEs$	1.6×10^7	1.6×10^7	1.6×10^7
Run times	15	15	15

In addition, other parameters in BPMOOPF-EA are set as shown in Table 2. Parameter T, which indicates the number of neighbors, is set to 3. The size of EP is the same as the population size. The replacement probability p_{rep} in $Comparator$ is set to 0.5. In $Remove\_Vector$, G_{remove}, which decides when to start deleting weight vectors, is set to 200. T_{itl} is set to 50, and N_{del} is set to 700. The final output population size, i.e., N_{final}, is set to 200.

Table 2: Other parameters of BPMOOPF-EA

Parameter	Meaning	Value
T	number of neighbors	3
N_{EP}	size of EP	10000
p_{rep}	replacement probability	0.5
G_{remove}	generation to start deleting	200
T_{itl}	time interval	50
N_{del}	number of removed individuals	700
N_{final}	size of final population	200

5.5. Experimental Results

In this section, we first analyze some important parameters of our proposed BPMOOPF-EA. Then, we compare experimental results of BPMOOPF-EA with two state-of-the-art constrained MOEAs, namely, C-MOEA/D [24] and A-NSGA-III [24], in dealing with the BPMOOPF problem.

5.5.1. Population Size N

We performed an experiment to compare the performance of BPMOOPF-EA with different initial population sizes. To better analyze the influence of the initial population size, the number of the independent runs was increased to 30 for each different N. The other settings remain the same as those in Section 5.4.

It should be noted that when the population size is less than 200, the algorithm does not perform the deletion operation, and the final output population size is equal to the initial population size.

Table 3: Experimental results of BPMOOPF-EA with different N

N	10000(100*100)	900(30*30)	100(10*10)
Feasible times	30	25	5
Ratio	100%	83.3%	16.7%
avgMPHV	0.0207±0.0013	0.0151±0.0077	0.0035±0.0082

The experimental results of BPMOOPF-EA with different values of N are shown in Table 3, where "Feasible times" is the number of runs when at least one feasible solution is found and "avgMPHV" means the mean and standard deviation of MPHV in 30 independent run for each different value of N. Table 3 shows that, given a sufficiently large population size and a sufficiently large number of fitness evaluations, our proposed BPMOOPF-EA can guide the population to feasible solutions in each run. However, when the population size is small, BPMOOPF-EA cannot always find feasible solutions. Moreover, according to the avgMPHV values, the diversity of the solutions also decreases. Therefore, for our proposed BPMOOPF-EA, the initial population size should not be too small.

5.5.2. Replacement Probability p_{rep}

The replacement probability p_{rep} is related to the convergence rate of the population. The greater the replacement probability p_{rep} is, the greater the possibility of having excellent offspring replacing their parents, which could produce a faster convergence of the population. On the contrary, the smaller the p_{rep} is, the more likely is to maintain diversity in the evolutionary process.

Since the setting of the replacement probability p_{rep} has an influence on the convergence rate of the population, the analysis of the replacement probability p_{rep} is necessary. The results with different p_{rep} are as shown in

Table 4. For each different p_{rep}, the times of independent experiments are set to 30. Other settings are the same as those in Section 5.4.

Table 4: Experimental results of BPMOOPF-EA with different p_{rep} values

p_{rep}	0.2	0.4	0.6	0.8	1.0
Feasible times	28	30	30	30	30
Ratio	93.3%	100%	100%	100%	100%
avgMPHV	0.0176±0.0049	0.0202±0.0011	0.0203±0.0010	0.0208±0.0015	0.0208±0.0010

As shown in Table 4, when p_{rep} is small, BPMOOPF-EA cannot always find feasible solutions. But if p_{rep} is set to a relatively large value, the ability of BPMOOPF-EA to search for feasible solutions is improved. In addition, it can be seen from the change of avgMPHV that the diversity of the obtained solutions is improved when comparing it with respect to the use of small p_{rep} values. However, after reaching a certain value, the growth trend of avgMPHV becomes very modest.

5.5.3. Number of Neighbors T

Parameter T is also very important since it affects the convergence rate of the population. Therefore, we conducted comparative experiments with different numbers of neighbors T.

Table 5 shows our experimental results for our proposed BPMOOPF-EA using different values of T, where the number of independent runs for each different T was set to 30. The other settings were the same as those reported in Section 5.4.

Table 5: Experimental results of BPMOOPF-EA with different T values

T	3	5	7	9
Feasible times	30	27	30	29
Ratio	100%	90%	100%	96.7%
avgMPHV	0.0207±0.0013	0.0191±0.0069	0.0216±0.0039	0.0220±0.0068

Our experimental results show that the setting of T has a certain impact on the performance of our proposed BPMOOPF-EA. But in general, the rate for BPMOOPF-EA to find feasible solutions is above 90%. The change of the parameter T affects the convergence rate of the population and the diversity in the evolutionary process, leading to a failure to find feasible solutions in some runs. Therefore, T needs to be set to an appropriate value.

5.5.4. Comparison of Results

Table 6 shows the comparison results among the three algorithms previously indicated. "avgMinCV" refers to the average minimum constraint violation in the final population of 15 independent runs, and "avgMPHV" refers to the mean and standard deviation of MPHV in 15 independent runs. It can be seen that our proposed BPMOOPF-EA is much better than the other two approaches with respect to which it was compared, under the same conditions.

Table 6: Comparison of results among the three algorithms selected

	BPMOOPF-EA	C-MOEA/D	A-NSGAIII
avgMinCV	0	1.2479	1.3689
Feasible times	15	0	0
Ratio	100%	0	0
avgMPHV	0.0207±0.0013	0±0	0±0
Execution time	1.31e+05 ± 2.44e+03	9.27e+04 ± 2.31e+03	1.00e+06 ± 4.56e+04

Figure 2 shows that, although the value of MPHV changes slightly, it fluctuates above and below the average value. It can be seen that the performance of BPMOOPF-EA is relatively stable. In addition, the last row shows the execution time (in seconds) of the three algorithms. It can be seen that the execution efficiency of BPMOOPF-EA is a bit worse than C-MOEA/D, but significantly better than A-NSGAIII.

In general, BPMOOPF-EA finds feasible solutions in all rounds of experiments compared to the comparison algorithm in experiments. In addition, we can optimize the algorithm's ability to search for feasible solutions and convergence speed by properly adjusting the values of parameters N, p_{rep} and T. Therefore, it can be said that BPMOOPF-EA has better robustness.

6. Conclusions and Future Work

In this paper, we defined the biparty multiobjective optimal power flow problem (BPMOOPF). In order to solve the BPMOOPF problem, we proposed a novel algorithm called BPMOOPF-EA, which is an improved version of MOEA/D. Compared with two state-of-the-art constrained multiobjective evolutionary algorithms (C-MOEA/D and A-NSGA-III), our proposed BPMOOPF-EA showed significant advantages when solving the BPMOOPF

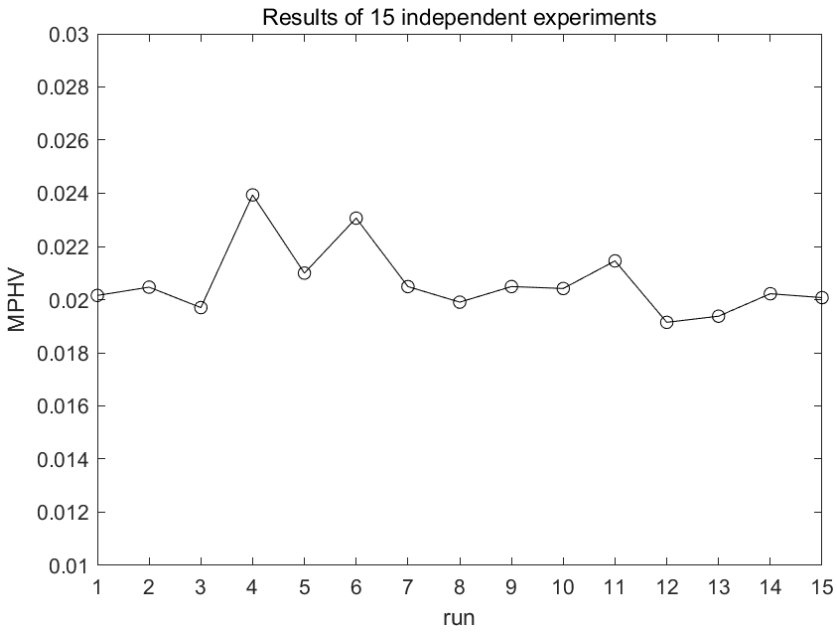

Figure 2: Experimental results of BPMOOPF-EA

problem. As part of our future work, we intend to continue to explore multi-party multiobjective optimal power flow problems with more decision makers and more optimization objectives.

Acknowledgments

This study is supported by Shenzhen Fundamental Research Program (Grant No. JCYJ20220818102414030), the Major Key Project of PCL (Grant No. PCL2022A03), Shenzhen Science and Technology Program (Grant No. ZDSYS20210623091809029), Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (Grant No. 2022B1212010005). Carlos A. Coello Coello gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (Investigación en Fronteras de la Ciencia 2016).

References

[1] M. B. Maskar, A. Thorat, I. Korachgaon, A review on optimal power flow problem and solution methodologies, in: Proceedings of the 2017

International Conference on Data Management, Analytics and Innovation, IEEE, 2017, pp. 64–70.

[2] Z. Qiu, G. Deconinck, R. Belmans, A literature survey of optimal power flow problems in the electricity market context, in: Proceedings of the 2009 IEEE/PES Power Systems Conference and Exposition, IEEE, 2009, pp. 1–6.

[3] T. Niknam, M. rasoul Narimani, M. Jabbari, A. R. Malekpour, A modified shuffle frog leaping algorithm for multi-objective optimal power flow, Energy 36 (11) (2011) 6420–6432.

[4] H. Pulluri, R. Naresh, V. Sharma, An enhanced self-adaptive differential evolution based solution methodology for multiobjective optimal power flow, Applied Soft Computing 54 (2017) 229–245.

[5] A. M. Shaheen, S. M. Farrag, R. A. El-Sehiemy, MOPF solution methodology, IET Generation, Transmission & Distribution 11 (2) (2017) 570–581.

[6] P. P. Biswas, P. N. Suganthan, R. Mallipeddi, G. A. Amaratunga, Multi-objective optimal power flow solutions using a constraint handling technique of evolutionary algorithms, Soft Computing 24 (4) (2020) 2999–3023.

[7] E. Naderi, M. Pourakbari-Kasmaei, F. V. Cerna, M. Lehtonen, A novel hybrid self-adaptive heuristic algorithm to handle single-and multi-objective optimal power flow problems, International Journal of Electrical Power & Energy Systems 125 (2021) 106492.

[8] N. Karthik, A. K. Parvathy, R. Arul, K. Padmanathan, Multi-objective optimal power flow using a new heuristic optimization algorithm with the incorporation of renewable energy sources, International Journal of Energy and Environmental Engineering 12 (2021) 641–678.

[9] S. Li, W. Gong, L. Wang, Q. Gu, Multi-objective optimal power flow with stochastic wind and solar power, Applied Soft Computing 114 (2022) 108045.

[10] H. T. Kahraman, M. Akbel, S. Duman, Optimization of optimal power flow problem using multi-objective manta ray foraging optimizer, Applied Soft Computing 116 (2022) 108334.

[11] T. Ganesan, I. Litvinchev, J. A. Marmolejo-Saucedo, J. J. Thomas, P. Vasant, Review on recent implementations of multiobjective and multilevel optimization in sustainable energy economics, in: Advances of Artificial Intelligence in a Green Energy Environment, Elsevier, 2022, pp. 245–277.

[12] W. Liu, W. Luo, X. Lin, M. Li, S. Yang, Evolutionary approach to multiparty multiobjective optimization problems with common pareto optimal solutions, in: Proceedings of the 2020 IEEE Congress on Evolutionary Computation, IEEE, 2020, pp. 1–9.

[13] P. P. Biswas, P. Suganthan, G. A. Amaratunga, Optimal power flow solutions incorporating stochastic wind and solar power, Energy Conversion and Management 148 (2017) 1194–1207.

[14] A. Kumar, B. K. Jha, S. Das, R. Mallipeddi, Power flow analysis of islanded microgrids: a differential evolution approach, IEEE Access 9 (2021) 61721–61738.

[15] R. Ramos, J. Vallejos, B. Barán, Multiobjective reactive power compensation with voltage security, in: Proceedings of 2004 IEEE/PES Transmision and Distribution Conference and Exposition: Latin America, IEEE, 2004, pp. 302–307.

[16] J. Zhang, L. Xing, A survey of multiobjective evolutionary algorithms, in: Proceedings of the IEEE International Conference on Computational Science and Engineering and IEEE International Conference on Embedded and Ubiquitous Computing, Vol. 1, IEEE, 2017, pp. 93–100.

[17] Z. She, W. Luo, Y. Chang, X. Lin, Y. Tan, A new evolutionary approach to multiparty multiobjective optimization, in: Proceedings of the International Conference on Swarm Intelligence, Springer, 2021, pp. 58–69.

[18] Y. Chang, W. Luo, X. Lin, Z. She, Y. Shi, Multiparty Multiobjective Optimization By MOEA/D, in: Proceedings of the 2022 IEEE Congress on Evolutionary Computation, IEEE, 2022, pp. 01–08.

[19] S. Mane, M. N. Rao, Many-objective optimization: Problems and evolutionary algorithms–a short review, International Journal of Applied Engineering Research 12 (20) (2017) 9774–9793.

[20] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, IEEE Transactions on Evolutionary Computation 21 (3) (2016) 440–462.

[21] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[22] A. Trivedi, D. Srinivasan, N. Biswas, An improved unified differential evolution algorithm for constrained optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2018, pp. 1–10.

[23] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2-4) (2000) 311–338.

[24] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach, IEEE Transactions on Evolutionary Computation 18 (4) (2013) 602–622.

[25] S. Frank, S. Rebennack, An introduction to optimal power flow: Theory, formulation, and examples, IIE Transactions 48 (12) (2016) 1172–1197.

[26] A. Kumar, G. Wu, M. Z. Ali, Q. Luo, R. Mallipeddi, P. N. Suganthan, S. Das, A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results, Swarm and Evolutionary Computation 67 (2021) 100961.

[27] K. Miettinen, Nonlinear multiobjective optimization, Vol. 12, Springer Science & Business Media, 2012.

[28] Q. Xu, Z. Xu, T. Ma, A survey of multiobjective evolutionary algorithms based on decomposition: variants, challenges and future directions, IEEE Access 8 (2020) 41588–41614.

[29] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation 13 (2) (2008) 284–302.

[30] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, J. Wu, MOEA/D with adaptive weight adjustment, Evolutionary Computation 22 (2) (2014) 231–264.

[31] W. Wang, K. Li, X. Tao, F. Gu, An improved MOEA/D algorithm with an adaptive evolutionary strategy, Information Sciences 539 (2020) 1–15.

[32] J. Chen, J. Ding, K. C. Tan, Q. Chen, A decomposition-based evolutionary algorithm for scalable multi/many-objective optimization, Memetic Computing 13 (3) (2021) 413–432.

[33] Q. Zhang, W. Liu, H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 203–208.

[34] T. Takahama, S. Sakai, Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–9.

[35] K.-D. Lu, Z.-G. Wu, Constrained-differential-evolution-based stealthy sparse cyber-attack and countermeasure in an ac smart grid, IEEE Transactions on Industrial Informatics 18 (8) (2021) 5275–5285.

[36] K.-D. Lu, Z.-G. Wu, T. Huang, Differential evolution-based three stage dynamic cyber-attack of cyber-physical power systems, IEEE/ASME Transactions on Mechatronics.

[37] K. Deb, S. Tiwari, Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization, European Journal of Operational Research 185 (3) (2008) 1062–1087.

[38] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: Proceedings of the 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1952–1959.

[39] R. Polakova, L-SHADE with competing strategies applied to constrained optimization, in: Proceedings of the 2017 IEEE congress on evolutionary computation, IEEE, 2017, pp. 1683–1689.

[40] E. Zitzler, D. Brockhoff, L. Thiele, The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration, in: Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 862–876.

[41] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes, IEEE Transactions on Evolutionary Computation 21 (2) (2016) 169–190.

[42] Y. Yuan, H. Xu, B. Wang, X. Yao, A new dominance relation-based evolutionary algorithm for many-objective optimization, IEEE Transactions on Evolutionary Computation 20 (1) (2015) 16–37.

[43] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum], IEEE Computational Intelligence Magazine 12 (4) (2017) 73–87.