Hindawi
Complexity
Volume 2020, Article ID 5980504, 23 pages
https://doi.org/10.1155/2020/5980504

Research Article

A Multiobjective Particle Swarm Optimization Algorithm Based on Competition Mechanism and Gaussian Variation

Hongli Yu,[1] Yuelin Gao ,[2] and Jincheng Wang[3]

[1]*School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China*
[2]*Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China*
[3]*Department of Basic, Yinchuan University, Yinchuan 750105, China*

Correspondence should be addressed to Yuelin Gao; gaoyuelin@263.net

Received 1 August 2020; Revised 10 October 2020; Accepted 27 October 2020; Published 1 December 2020

Academic Editor: Zhile Yang

In order to solve the shortcomings of particle swarm optimization (PSO) in solving multiobjective optimization problems, an improved multiobjective particle swarm optimization (IMOPSO) algorithm is proposed. In this study, the competitive strategy was introduced into the construction process of Pareto external archives to speed up the search process of nondominated solutions, thereby increasing the speed of the establishment of Pareto external archives. In addition, the descending order of crowding distance method is used to limit the size of external archives and dynamically adjust particle parameters; in order to solve the problem of insufficient population diversity in the later stage of algorithm iteration, time-varying Gaussian mutation strategy is used to mutate the particles in external archives to improve diversity. The simulation experiment results show that the improved algorithm has better convergence and stability than the other compared algorithms.

1. Introduction

In many engineering problems, the problems are composed of multiple goals that influence and conflict with each other. In solving practical issues, people often encounter multiple objectives that need to obtain the best optimal solution at the same time, that is, multiobjective optimization problems. The optimization problem has more than one optimization objective and needs to be processed at the same time, which becomes a multiobjective optimization problem (MOP). Usually, the optimal solution of the multiobjective optimization problem obtained after analyzing the objective function is the Pareto optimal solution set [1]. Therefore, in solving the multiobjective optimization problem [2], the following three key elements need to be solved: (1) The solution set is as close to the Pareto front as possible. (2) Keep the good diversity of the population as much as possible. (3) Make the particles effectively and uniformly distributed in the solution space. In recent years, in order to effectively solve MOP, multiobjective optimization algorithms based on different optimization theories have been continuously proposed. Deb et al. [3] proposed a multiobjective evolutionary computing algorithm based on nondominated sorting, called NSGA-II, and introduced a new selection operator into the algorithm to reduce the complexity of the algorithm. Zitzler et al. [4] proposed SPEA-II using the idea of Pareto domination. Coello et al. [5] proposed an improved multiobjective particle swarm algorithm that uses the concept of Pareto dominance to determine the flight direction of a particle and it maintains previously found nondominated vectors in a global repository that is later used by other particles to guide their own flight. This algorithm improves the efficiency of solving multiobjective problems. Tsai et al. [6] proposed an improved multiobjective particle swarm optimizer with proportional distribution and jump improved operation, named PDJI-MOPSO, for dealing with multiobjective problems. PDJI-MOPSO maintains diversity of newly found nondominated solutions via proportional distribution and combines advantages of wide-ranged exploration and

extensive exploitations of PSO in the external repository with the jump improved operation to enhance the solution searching abilities of particles. Introduction of cluster and disturbance allows the proposed method to sift through representative nondominated solutions from the external repository and prevent solutions from falling into local optimum. Mnif et al. [7] introduced a new approach called multiobjective firework algorithm (MFWA). Mellouli et al. [8], in order to solve the two-dimensional cutting stock problem, combined genetic algorithm with linear programming model to estimate the best Pareto frontier for these two goals. The Pareto front provided by this algorithm is very close to the optimal front. Ali et al. [9] proposed a modified variant of Differential Evolution (DE) algorithm for solving multiobjective optimization problems. The proposed algorithm, named Multiobjective Differential Evolution Algorithm (MODEA), utilizes the advantages of Opposition-Based Learning for generating an initial population of potential candidates and the concept of random localization in mutation step. Zhou et al. [10] proposed a generic transformation strategy that can be referred to as the Mucard strategy, which converts an MSCCOP into a low-dimensional multiobjective optimization problem (MOP) to simultaneously obtain all the (near-) optima of the constrained optimization problems in a single algorithmic run. Vargas et al. [11] studied the performance of the combination of adaptive penalty technology called APM and GDE3 algorithm. Sun et al. [12] proposed a novel multiobjective particle swarm optimization algorithm based on Gaussian mutation and an improved learning strategy. This method uses Gaussian mutation strategy to improve the consistency of external archives and current population. In order to improve the global optimal solution, different learning strategies are proposed for nondominant and dominant solutions. An indicator is proposed to measure the distribution width of nondominated solution sets generated by various algorithms. Coello et al. [13] proposed that the external set could be used to retain the nondominated solutions found in the iterative process, which improved the efficiency of the algorithm. Tsai et al. [6] used the specific global optimal value to replace the individual optimal value fully using the guiding role of the global optimal value. Zhang [14] proposed an improved MOPSO algorithm with a mutation operator that can maintain the diversity of optimal solutions and has good convergence. Zhang [15] proposed a MOPSO algorithm based on fuzzy dominance, and the experimental results showed the effectiveness of the proposed algorithm. Tao [16] proposed a multiobjective optimization algorithm combining PSO and difference algorithms. By generating common new particles and updating the particle velocity formula, the search efficiency of the algorithm was effectively improved. Li [17] proposed an improved MOPSO algorithm that updates the optimal position of all particles through the Pareto dominance relationship. The experiment shows that the proposed algorithm can obtain a better noninferior solution. Ni [18] proposed an adaptive dynamic recombinant PSO algorithm that adopts a high-level clustering algorithm. The experimental results show that this algorithm can improve the convergence speed and evolutionary ability of the algorithm.

More and more improved algorithms and strategies are used to solve various multiobjective optimization problems. However, few researchers have improved the algorithm from the perspective of the balance between local search capability (exploitation) and global search capability (exploration) to solve multiobjective optimization problems. Related research has pointed out that the effective balance between the exploration and exploitation of intelligent algorithms will have a vital impact on the optimization performance of the algorithm [19]. Different from other study results, in this study, we proposed a multiobjective particle swarm optimization algorithm based on competition mechanism strategy and Gaussian mutation to balance the exploration and exploitation of the algorithm and enable the algorithm to search the optimal location and converge to the Pareto front more quickly. Finally, through the simulation test of the multiobjective test functions and the comparison with other multiobjective optimization algorithms, the results show that the algorithm proposed in this paper is superior to the compared algorithm in terms of convergence and population distribution.

The rest of the paper is organized as follows. Section 2 describes the multiobjective optimization problem and the basic concept. Section 3 presents the PSO algorithm. In Section 4, an improved multiobjective particle swarm optimization algorithm is introduced. In Section 5, the evaluation indicators and test functions used are introduced. The numerical experiment results and data analysis are described in Section 6. Finally, conclusions and respects are presented in Section 7.

2. Multiobjective Optimization Problem and the Basic Concept

2.1. Formal Definition of Multiobjective Optimization Problem. Generally, a multiobjective optimization problem (MOP) includes a set of objective functions and some constraints\enleadertwodots. Without loss of generality, an MOP with m objective functions and n decision variables can be described as follows [20, 21]:

$$\min f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$
$$\text{s.t.} \begin{cases} g_i(x) \le 0, & i = 1, 2, \ldots, p \\ h_j(x) = 0, & j = 1, 2, \ldots, q \\ x \in R^n, \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n \in R^n)$ are the n-dimensional decision variables. $f = (f_1, f_2, \ldots, f_m)$ is the objective function, which contains one or more objective functions; and $f = (f_1, f_2, \ldots, f_m)$ is the multiobjective function $(m \ge 2)$; $f_k(x)$: $R^n \longrightarrow R$, $(k = 1, 2, \ldots, m)$ is the $k-$th objective function; g_i: $R^n \longrightarrow R$, $(i = 1, 2, \ldots, p)$ is the $i-$th inequality constraint; h_j: $R^n \longrightarrow R$, $(j = 1, 2, \ldots, q)$ represents the $j-$th equality constraint condition.

$X = \{x \in R^n | g_i \le 0, \quad i = 1, 2, \ldots, p; h_j = 0, j = 1, 2, \ldots, q\}$ is the feasible domain of decision variables.

2.2. Pareto Optimal Solution Concepts. The Pareto optimal solution was discovered by the Italian economist Pareto. It was originally limited to the field of economics; this rule has gradually extended to various fields of social life and was deeply recognized by people as well. The strict Pareto optimal solution can be described by multiobjective mathematical programming. Assuming that there are several objectives at the same time, these objectives are independent of each other and cannot be weighed and summed. New optimization theories are needed to solve such problems. Generally, if one subobjective is improved, other subobjectives will be sacrificed. So, it is impossible to improve all subobjectives at the same time. Pareto optimal solution is also called nondominated solution. In a multiobjective optimization problem, due to factors such as conflict and incomparability among various subobjectives, a solution is often the best in one subobjective and may be the worst in other subobjectives; if there is one solution, improving any subobjective function will inevitably weaken at least one other subobjective function, which is called nondominated solution or Pareto optimal solution. All Pareto optimal solutions constitute the Pareto optimal solution set, and these solutions are mapped by the objective function to form the Pareto optimal front. Pareto proposed the concept of nondominated set of multiobjective solutions in 1986, which is defined as follows: assuming any two solutions S_1 and S_2 for all objectives, if S_1 is better than S_2, then we say that S_1 dominates S_2, and if S_1 is not dominated by other solutions, then S_1 is called a nondominated solution, also called a Pareto optimal solution.

2.3. Related Definitions of Multiobjective Optimization Problem. The following introduces some basic concepts in the multiobjective optimization problem [22, 23].

Definition 1. (feasible domain). In the decision space, the feasible domain is represented by X, and its expression is

$$X = \{x \in R^n | g_i \le 0, i = 1, 2, \ldots, p; h_j = 0, j = 1, 2, \ldots, q\}. \tag{2}$$

Definition 2. (feasible solution). For the point x in the decision space, if $x \in X$, x is the feasible solution.

Definition 3. (Pareto dominates). For any vector x_1, x_2, if and only if

$$\begin{aligned} \forall i \in R^n, \quad & f_i(x_1) \le f_i(x_2), \\ \wedge \exists j \in R^n, \quad & f_j(x_1) < f_j(x_2), \end{aligned} \tag{3}$$

then x_1 dominates x_2, so $x_1 \prec x_2$.

Definition 4. (Pareto optimal solution). The Pareto optimal solution is also known as the nondominant solution. For a solution x^* in the feasible domain, if x^* is not dominated by any other solution in the feasible domain, x^* is called the Pareto optimal solution, and its definition is as follows:

$$\exists x \in X: x \prec x^*. \tag{4}$$

Definition 5. (Pareto optimal solution set, PS). The set of all nondominated solutions in the feasible domain is called the Pareto optimal solution set and can be defined as follows:

$$\mathrm{PS} = \{x^* | \exists x \in X: x \prec x^*\}. \tag{5}$$

Definition 6. (Pareto optimal frontier, PF). The target vector set corresponding to the Pareto optimal solution set is the Pareto optimal frontier, also known as the Pareto optimal front end or Pareto equilibrium surface, which is defined as follows:

$$\mathrm{PF} = \{F(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_d(x^*)) | x^* \in \mathrm{PS}\}. \tag{6}$$

3. Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) is a heuristic swarm intelligence algorithm that solves optimization problems by imitating the swarm behavior of birds [24]. The algorithm has high stability and good adaptability. It is an intelligent global optimization algorithm that has attracted the attention of many scholars in recent years. Each particle in the PSO algorithm is equivalent to a bird in the population, so each particle has its own speed and position. Through self-learning and social learning, the particles move in the solution space to obtain the global optimum solution. Assuming that the population size of the particle is N and the dimension of the space is D, the update formula for the velocity and position of the particle are as follows:

$$\begin{aligned} v_{ij}^{t+1} &= \omega v_{ij}^t + c_1 r_1^k \left(\mathrm{pbest}_{ij}^t - x_i^t\right) + c_2 r_2^k \left(\mathrm{gbest}_j^t - x_i^t\right), \\ x_{ij}^{t+1} &= x_{ij}^t + v_{ij}^{t+1}, \end{aligned} \tag{7}$$

where i ($i \le N$) is the $i-$th particle, j ($j \le D$) is the velocity (position) of the $j-$th dimension, t is the current iteration number, velocity and position are limited in a certain range, ω is the inertia weight, c_1 and c_2 are two positive constants in $[0, 4]$ which represent the learning factors, r_1 and r_2 are two random numbers in $[0, 1]$, pbest_{ij} represents a coordinate that defines a particle pbest_i in the $j-$th dimension of the individual best position, and gbest_j represents a coordinate that defines the global best position in the $j-$th dimension of the population. ω is determined by the following formula:

$$\omega = \omega_{\max} - t * \frac{(\omega_{\max} - \omega_{\min})}{T_{\max}}, \tag{8}$$

where $\omega_{\max}, \omega_{\min}$ are the maximum and minimum values of inertia weight. Generally the value is set as 0.9 and 0.4, t represents the number of current iterations, and $T_{\max}$ is the maximum number of iterations.

4. An Improved Multiobjective Particle Swarm Optimization Algorithm

4.1. Competition Mechanism. The competition mechanism of this paper can quickly search the disposal solution set and build the external archives set in the MOPSO algorithm. First, select a particle x from the population s, and generally select the first particle in the population. Then, let $s = s - \{x\}$ and compare each particle in the x population on the basis of the objective function value of Pareto dominance relations. If $x \prec y$, particle y will be removed from the population s; otherwise, let $x = y$. Finally, let $n = n \cup \{x\}$ until $s = \phi$, at which time n is the nondominant solution set to be solved. This method is also adopted when the nondominant solution set enters the external archive set. As more and more particles are removed, the number of times the algorithm runs is less, which can effectively reduce complexity of the algorithm and improve search speed of the algorithm. The pseudocode of the algorithm is shown in Table 1.

4.2. External Archives Maintenance Strategy. In solving a MOPSO algorithm, each iteration produces a set of Pareto solutions. Therefore, external archives are needed to store the nondominated solutions produced by each iteration, and the solution set forms the Pareto front [25]. After each iteration, the Pareto front is also updated. However, as the number of iterations increases, the size of external archives will increase, and the complexity of the algorithm will also greatly increase. Therefore, it is necessary to limit the size of external archives to reduce the complexity of the algorithm. This paper uses the descending order of crowded distance to limit the size of the external archives set.

4.3. Selecting pbest and gbest. In a MOPSO algorithm, selecting the gbest is very important, which directly affects the convergence speed and capabilities. In IMOPSO, the size of the particle population is fixed, and the particles will not be deleted from the population, but the position of the particles in the population needs to be adjusted to update pbest and gbest. In multiobjective conditions, gbest normally exists in a group of noninferior solutions and is not a single gbest position. When gbest and pbest are nondominated, each particle may have more than one pbest. Therefore, it is necessary to choose pbest and gbest by the appropriate method.

4.3.1. Selecting pbest. The specific process is as follows: pbest_i^t is used to record the individual position and save the nondominated solution of the particles in the evolution process. The updating formula of pbest_i^t for the $t-$th generation particles is as follows:

$$\text{pbest}_i^t \begin{cases} \text{pbest}_i^t, & \text{if}\left(f\left(\text{pbest}_i^t\right) \prec f\left(x_i^t\right)\right), \\ x_i^t, & \text{if}\left(f\left(\text{pbest}_i^t\right) \succ f\left(x_i^t\right)\right), \\ \text{randselect}\left(\text{pbest}_i^t, x_i^t\right), & \text{if (otherwise)}, \end{cases} \tag{9}$$

Table 1: Constructing nondominated set n by competition mechanism.

```
Call function: n = find_nondominated_set (s)
(1) n = ∅
(2) while (|s| > 1){
(3) x = first (s); //Assign the first particle
(4)                 //in population s to x
(5) s = s − {x};
(6) for (each y ∈ s) {
(7) if (x dominates y) s = s − {y};
(8) else {x} = {y};
(9) } //end for
(10) n = n + {x};
(11) }//end while
```

where pbest_i^t is the optimal position of the $i-$th particle in the previous t generation and x_i^t is the position of the $i-$th particle in the $t-$th generation.

4.3.2. Selecting gbest. The specific process is as follows: In the selection process of gbest, this paper adopts the Pareto principle, also known as the asymmetry principle or the 80/20 law; that is, 80% of the results in practical issues are produced by 20% of key factors [26]. Therefore, in this study, the global optimal value is randomly selected from the top 20% nondominated solutions of the external archive.

4.4. Parameter Improvement Strategy. Inertia weight ω and the learning factors c_1 and c_2 in the PSO algorithm have a considerable influence on the searching ability of the population in the target region. However, the traditional linear adjustment strategies for these two types of parameters cannot effectively reflect the search process of the algorithm. Therefore, this article adopts a nonlinear dynamic adjustment strategy to more accurately reflect the search process of the algorithm and to more effectively balance the exploitation and exploration.

4.4.1. Inertia Weight. The inertia weight ω determines the influence of the velocity of the previous generation particle on the current velocity. The appropriate adjustment rule of ω can effectively balance the exploitation and exploration of the algorithm. If $\omega = 0$, the particle speed is determined by the current position, and the particle has no inheritance to the speed of the previous generation, so the algorithm is easy to fall into the local optimum. If $\omega \neq 0$, the larger inertial weight can strengthen the global search capability, and the smaller inertial weight can strengthen the local search capability. In order to effectively balance the exploration and exploitation and improve the search performance of the algorithm, this paper adopts a new strategy for adjustment; the updated formula is as follows:

$$\omega(t) = \omega_{\max} * \frac{1}{1 + \left(t/p_1\right)^{p_2}} + \omega_{\min}, \tag{10}$$

where $\omega(t)$ changes with the number of iterations. In this paper, p_1 is one-third of the maximum number of iterations.

p_2 is 10. $\omega_{\max}$ is 0.5. $\omega_{\min}$ is 0.4. t is the number of current iterations. Figure 1(a) shows the change curve of the new inertia weight.

4.4.2. Learning Factor. The learning factor determines the influence of self-learning ability and social-learning ability on particle motion during the search process, which reflects the state of information exchange between particles. In recent years, many scholars have revised the learning factors. Some scholars have proposed asynchronous learning factors; that is, in the initial search phase of the algorithm, the particles have greater self-learning ability and smaller social-learning ability; in the later phase of the algorithm search, the asynchronous learning factors can enhance the ability of particles to move to the global optimal position and obtain high-quality particles, which makes the algorithm have a higher probability of converging to the global optimal solution. For the learning factors, the nonlinear function of the new inertia weight mentioned above [27] is adopted in this paper, and its updated formula is as follows:

$$\begin{aligned} c_1(t) &= 1.167 \times \omega(t)^2 - 0.1167 \times \omega(t) + 0.66, \\ c_2(t) &= 3 - c_1(t), \end{aligned} \tag{11}$$

where, through the above formula, it is known that the learning factor also adjusts dynamically with the adjustment of inertia weight. Figure 1(b) shows the change curve of the learning factor.

4.5. Time-Varying Gaussian Mutation. In solving the problem of multiobjective optimization, the fast convergence of the PSO may not be advantageous. Instead, it may make the population prematurely gather around certain particles or a certain area and lose diversity, which is easy to make the algorithm show premature phenomena. Therefore, to enhance the population diversity and avoid the algorithm falling into local optimum, based on literature [13], this paper designs a kind of time-varying particle mutation to produce new solutions through mutation operators to external archives. The perturbation formula is as follows:

$$\begin{aligned} x_i^{t+1} &\begin{cases} x_i^t + r_g * \text{mut}_r\text{ange}, & P_m \geq \text{rand}, \\ x_i^t, & P_m < \text{rand}, \end{cases} \\ P_m &= \left(1 - \frac{t}{t_{\max}}\right)^{5/M_r}, \\ \text{mut}_r\text{ange} &= (ub(j) - lb(j)) * P_m, \end{aligned} \tag{12}$$

where P_m is the mutation probability. r_g is subject to a Gaussian distribution with a mean of 0 and a variance of 1. mut_range represents the scope of action of variation. $ub(j)$ and $lb(j)$ are the upper bound and lower bound of decision variables of the $j-$th dimension, respectively. M_r is the mutation parameter, and the value in this paper is 0.5.

4.6. The Specific Steps of the IMOPSO Algorithm

Step 1. Initialize the group's position and speed. Set the iteration times, population size, and algorithm parameters.

Step 2. Calculate the fitness value of each particle. Generate a nondominating solution set according to dominating relation.

Step 3. Update the external archive set.

Step 4. Arrange each particle of the external archival set in descending order according to the crowding distance and determine whether the set size number is exceeded. If it is exceeded, clear the nondominant solution beyond the size.

Step 5. Update the individual optimal position; if it is the first generation, directly select the initial position of each particle as the individual optimal value; otherwise, update according to the Pareto dominance relationship.

Step 6. Randomly select the global optimal location from the external archive set ranked in the top 20% nondominant solution.

Step 7. Update the speed formula. If the particle velocity $v_i > v_{\max}$, then let $v_i = v_{\max}$; if the particle velocity $v_i < v_{\min}$, then let $v_i = v_{\min}$.

Step 8. Update the position of the next generation of each particle and carry out time-varying Gaussian variation on the particles in each external archive according to the probability P to avoid premature precocity of the algorithm.

Step 9. Determine whether the condition (the maximum number of iterations) is met, and if so, end the loop. Otherwise, return to Step 2 to continue the iteration.

5. Algorithm Performance Evaluation Indexes and Test Functions

5.1. Performance Measurement. The quality evaluation mainly focuses on the distance between the solution produced by the algorithms and the Pareto optimal solution for MOPs and the extent covered by the solution produced by the algorithms. In this paper, two performance indicators are adopted.

5.1.1. Generational Distance. Generational Distance (GD) is used to evaluate the convergence performance of multiobjective algorithms. It is used to calculate the average minimum Euclidean distance from each point in the solution set n to the reference set n^*. The calculation formula is defined as follows [28]:

$$\text{GD}(n, n^*) = \frac{\sqrt{\sum_{y \in n} \min_{x \in n^*} \text{dis}(x, y)^2}}{|n|}, \tag{13}$$

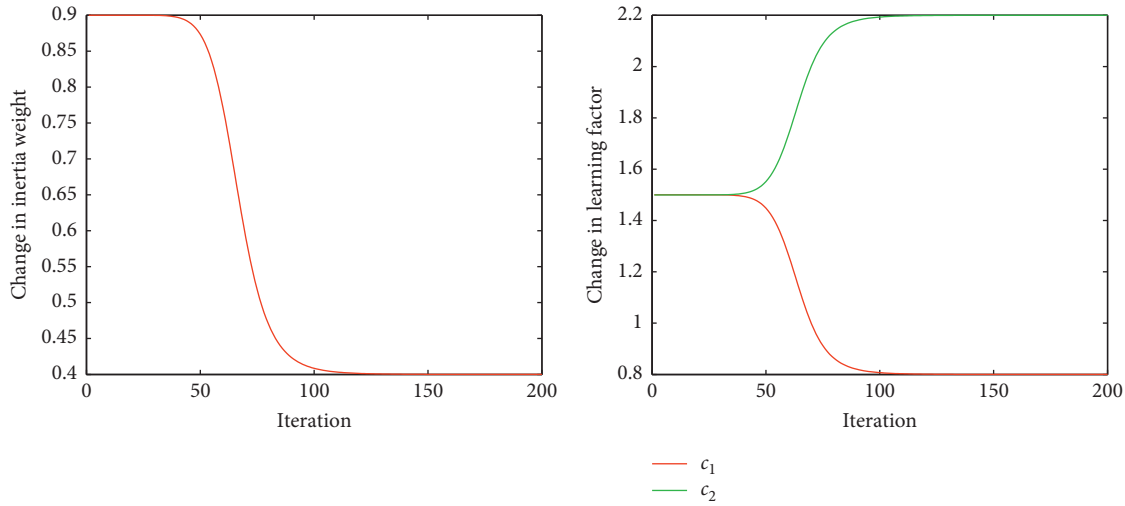

Figure 1: The curves of (a) new inertia weight and (b) learning factor.

where n is the solution set obtained by the algorithm, n^* is a set of uniformly distributed reference points sampled from true Pareto front (PF), and $\mathrm{dis}(x, y)$ is the Euclidean distance between the point y in the solution set n and the point x in the reference set n^*. The smaller GD is, the closer the Pareto optimal solution set obtained by the algorithm is to the true Pareto front, and the better convergence of the algorithm is. The ideal value of GD is 0; that is, the Pareto optimal solution obtained by the algorithm is on the true Pareto front.

5.1.2. Spacing. Spacing (SP) is a parameter used to evaluate the performance of the Pareto optimal solution obtained by the algorithm. The mathematical expression is [29, 30]

$$\mathrm{SP} = \sqrt{\frac{1}{|n| - 1} \sum_{i=1}^{|n|} \left(\overline{d} - d_i\right)^2}, \tag{14}$$

where n represents the number of nondominated solutions found; $\overline{d}$ is the average of all d_i, and d_i is calculated as follows:

$$d_i = \min_{j=1,\ldots,|n|} \left\{ \sum_{m=1}^{k} \left| f_m(x_i) - f_m\left(x_j\right) \right| \right\}, \tag{15}$$

where k represents the number of objective functions. The smaller SP is, the better the distribution of the solution obtained by the algorithm is. The ideal value of SP is 0; that is, the Pareto optimal solution obtained by the algorithm is evenly distributed in the target space.

5.1.3. Inverted Generational Distance. GD can only evaluate the convergence of the algorithm. In order to further evaluate the comprehensive performance of the algorithm, Inverted Generational Distance (IGD) was proposed. IGD represents the average value of the Euclidean distance between each reference point in the reference set n^* and the closest solution to n. The closer the IGD value is to 0, the better the overall performance of the algorithm is. The calculation formula of IGD is as follows:

$$\mathrm{IGD}\left(n, n^*\right) = \frac{\sum_{x \in n^*} \min_{y \in n^*} \mathrm{dis}(x, y)^2}{\left|n^*\right|}, \tag{16}$$

where n is the solution set obtained by the algorithm, n^* is a set of uniformly distributed reference points sampled from true PF, and $\mathrm{dis}(x, y)$ is the Euclidean distance between the point x in the reference set n^* and the point y in the solution set n.

5.2. Test Functions. To test the performance of the IMOPSO algorithm, this paper selects the classic multiobjective test functions for the simulation test [27, 31]. The test functions are expressed in Tables 2 and 3.

6. Experimental Analysis and Comparison

Through experiments, the IMOPSO algorithm in this paper was compared with the experimental results of the NSGA-II, SPEA-II, MOPSO, and NSGA-III [32] algorithms. The population size was set as 100, the maximum number of iterations was set as 10,000, and the size of the external archive was set as 100. The specific parameters of each algorithm were set as shown in Table 3. Among them, pc is the crossover probability, pm is the mutation probability, mu is the mutation rate, sep is the variable step length, ng is the grid control maximal particle number, cg is the cross probability, aph is the expansion probability, and t_1 and t_2 are mutation parameters. The NSGA-II, SPEA-II and MOPSO, and NSGA-III and IMOPSO independent algorithms were ran 30 times on each test function. Tables 4–6, respectively, show the convergence, distribution, and comprehensive performance of the algorithms. The test was

Table 2: Test functions: ZDT.

Name	Objective functions	Dimension	Variable bounds
ZDT1	$\begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = g(x)(1 - \sqrt{(x_1/g(x))}) \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/(n-1) \end{cases}$	30	$x_i \in [0,1]$
ZDT2	$\begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = g(x)(1 - (x_1/g(x))^2) \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/(n-1) \end{cases}$	30	$x_i \in [0,1]$
ZDT3	$\begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = g(x)(1 - \sqrt{x_1/g(x)} - (x_1/g(x))\sin(10\pi x_1)) g(x) = 1 + 9\sum_{i=2}^{n} x_i/(n-1) \end{cases}$	30	$x_i \in [0,1]$
ZDT4	$\begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = g(x)\left\{1 - \sqrt{x_1/g(x)}\right\} \\ g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(10\pi x_i)) \end{cases}$	10	$x_i \in [0,1]\, x_i \in [-5,5]$
ZDT6	$\begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = g(x)\left\{1 - (x_1/g(x))^2\right\} \\ g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(10\pi x_i)) \end{cases}$	10	$x_1 \in [0,1]$

Table 3: Test functions: DTLZ.

Name	Objective functions	Dimension	Variable bounds
DTLZ1	$\begin{cases} f_1(x) = 0.5(1 + g(x_M))\prod_{i=1}^{m-1} x_i \\ f_{m=2;M-1}(x) = 0.5(1 + g(x_M))(1 - x_{M-m+1})\prod_{i=1}^{m-1} x_i \\ f_M(x) = (1 + g(x_M))\sin(x_i\pi/2) \\ g(x) = (100\lvert x_M\rvert + \sum_{x_i \in M}(X - 0.5)^2 - \cos(20\pi(x - 0.5))) \end{cases}$	30	$x_i \in [0,1]$
DTLZ2	$\begin{cases} \min f_1(x) = 0.5(1 + g(x_M))\prod_{i=1}^{M-1}\cos(x_i\pi/2) \\ \min f_{m=2;M-1}(x) = 0.5(1 + g(x_M)) \\ \sin(x_{M-m+1}\pi/2)\prod_{i=1}^{M-1}\cos(x_i\pi/2) \\ \min f_M(x) = (1 + g(x_M))\sin x_i\pi/2 \\ g(x_M) = \sum_{x_i \in M}(x_i - 0.5)^2 \end{cases}$	30	$x_i = \in [0,1]$
DTLZ3	$\begin{cases} f_1 = 0.5(1 + g(x_M))\prod_{i=1}^{M} -1\cos(\text{frac}x_i 2) \\ f_{m=2;M-1}(x) = 0.5(1 + g(x_M))\sin(\pi x_{M-m+1}/2) \\ f_M(x) = 100(\lvert x_M\rvert + \sum_{x_i \in M}(x - 0.5)^2 - \cos(20\pi(x_i - 0.5))) \\ g(x) = \sum_{x_i \in M}(x - 0.5)^2 \end{cases}$	30	$x_i \in [0,1]$
DTLZ4	$\begin{cases} f_1(x) = 0.5(1 + g(x_M))\prod_{i=1}^{M} -1\cos(x_i\pi/2) \\ f_{m=2;M-1}(x) = 0.5(1 + g(x_M)) \\ \sin(\pi x_{M-m+1}/2)\prod_{i=1}^{M} -1\cos(\pi x_i/2) \\ f_M(x) = (1 + g(x_M))\sin(x_1^a\pi) \\ g(x) = \sum_{x_i \in M}(X - 0.5)^2, a = 100 \end{cases}$	30	$x_i \in [0,1]$
DTLZ5	$\begin{cases} f_1(x) = 0.5(1 + g(x_M))\prod_{i=1}^{M} -1\cos(\pi(1 + 2g(x_M)x_1)/2(2 + (1 + g(x_M)))) \\ f_{m=2;M-1}(x) = 0.5(1 + g(x_M))\cdot \\ \sin(\pi x_{M-m+1}/2)\prod_{i=1}^{M} -m\cos(\pi(1 + 2g(x_M)x_1)/2(2 + (1 + g(x_M)))) \\ f_M(x) = (1 + g(x_M))\sin(\pi x_i/2) \\ g(x) = \sum_{x_i \in M}(x - 0.5)^2 \end{cases}$	30	$x_i \in [0,1]$

Table 3: Continued.

Name	Objective functions	Dimension	Variable bounds
DTLZ6	$\begin{cases} f_1(x) = 0.5(1+g(x_M))\prod_{i=1}^{M} -1\cos(\pi(1+2g(x_M)x_i)/2(2+(1+g(x_M)))), \\ f_{m=2;M-1}(x) = 0.5(1+g(x_M))\cdot \\ \sin(\pi x_{M-m+1}/2)\prod_{i=1}^{M} -m\cos(\pi(1+2g(x_M)x_i)/2(2+(1+g(x_M)))) \\ f_M(x) = (1+g(x_M))\sin(\pi x_i/2) \\ g(x) = \sum_{x_i \in M} x_i \end{cases}$	30	$x_i \in [0,1]$
DTLZ7	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = x_2 \\ f_3(x) = (1+g(x_k))h(f_1, f_2, g) \\ g(x_k) = 1 + 9/\lvert x_k \rvert \sum_{x_i \in x_k} x_i h(f_1, f_2, g) = k - \sum_{i=1}^{k-1}((f_i/1+g)(1+\sin(3\pi f_i))) \end{cases}$	30	$x_\epsilon [0,1]$

Table 4: Parameters setting.

Algorithms	Parameters
NSGA-II	$pc = 0.8,\ pm = 0.3,\ mu = 0.02,\ sep = 0.1$
SPEA-II	$pc = 0.5,\ pm = 0.5,\ ng = 7,\ cg = 0.15$
MOPSO	$c_1 = c_2 = 2,\ w = 0.5,\ wp = 0.9,\ ng = 7\ ,aph = 0.1$
NSGA-III	$t_1 = 20,\ t_2 = 20,\ pc = 0.8,\ mu = 0.3$
IMOPSO	$w_{\max} = 0.5,\ w_{\min} = 0.4$

Table 5: Statistical results of GD obtained by different methods for different test functions.

Test functions	GD	NSGA-II	SPEA-II	MOPSO	NSGA-III	IMOPSO
ZDT1	Average	$7.71E-3$	$2.09E-2$	$1.01E-2$	$2.84E-3$	**1.08E − 4**
	Std. dev.	$7.28E-4$	$2.31E-2$	$9.59E-4$	$5.95E-4$	**1.37E − 5**
ZDT2	Average	$7.69E-3$	$2.32E-2$	$2.53E-2$	**9.60E − 4**	$4.15E-3$
	Std. dev.	$8.98E-3$	$1.92E-3$	$4.81E-3$	**4.83E − 4**	$6.47E-3$
ZDT3	Average	$6.74E-4$	$1.15E-3$	$1.25E-1$	$1.69E-3$	**6.91E − 5**
	Std. dev.	$1.59E-4$	$5.85E-4$	$2.91E-2$	$4.51E-4$	**3.25E − 5**
ZDT4	Average	$2.34E-2$	$1.22E+0$	$6.41E-3$	$6.19E-3$	**1.24E − 3**
	Std. dev.	$2.60E-2$	$1.78E+0$	$7.63E-3$	$1.73E-3$	**1.07E − 3**
ZDT6	Average	$8.69E-3$	$5.37E-3$	$2.76E-1$	$6.53E-2$	**2.29E − 3**
	Std. dev.	$3.69E-3$	$5.43E-3$	$3.43E-1$	$3.29E-2$	**1.79E − 3**
DTLZ1	Average	$4.19E-2$	$1.94E-1$	$7.11E+0$	$4.14E-2$	**2.15E − 2**
	Std. dev.	$5.18E-2$	$2.78E-1$	$1.92E+0$	$3.36E-2$	**2.09E − 2**
DTLZ2	Average	$1.31E-3$	$1.47E-3$	$8.90E-3$	**6.08E − 5**	$1.68E-3$
	Std. dev.	$6.54E-3$	$2.00E-4$	$2.70E-3$	**4.65E − 5**	$3.73E-4$
DTLZ3	Average	$1.48E+0$	$1.14E+1$	$2.91E+1$	$2.89E+1$	**1.09E + 0**
	Std. dev.	$3.74E+0$	$8.16E+0$	$1.03E+1$	$5.39E+1$	**3.87E − 1**
DTLZ4	Average	$1.98E-3$	$2.09E-3$	$3.96E-2$	**5.76E − 4**	$1.56E-3$
	Std. dev.	$2.30E-4$	$1.36E-4$	$2.59E-2$	**1.39E − 4**	$2.13E-4$
DTLZ5	Average	$2.70E-4$	$3.19E-4$	$1.58E-3$	$3.25E-4$	**1.88E − 4**
	Std. dev.	$7.01E-5$	$8.19E-5$	$1.06E-3$	$8.32E-5$	**6.59E − 5**
DTLZ6	Average	$1.31E-5$	$5.94E-3$	$7.11E-2$	$4.26E-1$	**5.19E − 6**
	Std. dev.	$4.53E-5$	$9.37E-2$	$4.28E-3$	$1.72E-2$	**2.97E − 7**
DTLZ7	Average	$7.06E-3$	$6.18E-3$	$5.97E-1$	$6.85E-3$	**3.13E − 3**
	Std. dev.	$1.56E-3$	$1.60E-3$	$4.03E-1$	$2.24E-3$	**1.06E − 3**

completed in MATLAB 2018a with a Windows 10 system, and the computer was configured with an Intel Core *i*7 3.40 GHz processor.

The numerical experiment results are shown in Tables 4–6, and the optimal results are marked in bold. Table 4 gives the numerical experiment result of the

Table 6: Statistical results of SP obtained by different methods for different test functions.

Test functions	SP	NSGA-II	SPEA-II	MOPSO	NSGA-III	IMOPSO
ZDT1	Average	$7.33E-3$	$8.97E-3$	$1.28E-2$	$1.08E-2$	**6.47E − 3**
	Std. dev.	$6.77E-4$	$1.91E-3$	$2.18E-3$	$1.65E-3$	**6.34E − 4**
ZDT2	Average	$7.85E-3$	$8.99E-3$	$7.88E-3$	$1.19E-2$	**7.79E − 3**
	Std. dev.	$2.34E-3$	$4.61E-3$	$5.74E-3$	$5.76E-3$	**1.75E − 3**
ZDT3	Average	$8.16E-3$	$1.05E-2$	$1.37E-2$	$1.06E-2$	**7.101E − 3**
	Std. dev.	$9.32E-4$	$2.06E-3$	$5.40E-3$	$1.46E-3$	**9.28E − 4**
ZDT4	Average	$2.76E-2$	$1.36E-1$	$3.70E-2$	$6.09E-3$	**1.81E − 2**
	Std. dev.	$2.79E-2$	$3.62E-1$	$3.93E-2$	$2.83E-2$	**5.27E − 2**
ZDT6	Average	$1.73E-2$	$1.61E-2$	$8.82E-2$	$4.23E-2$	**1.19E − 2**
	Std. dev.	$3.55E-2$	$3.80E-2$	$6.13E-2$	$2.08E-2$	**4.01E − 3**
DTLZ1	Average	**5.81E − 2**	$1.44E+0$	$5.51E+0$	$6.13E-2$	$3.93E-1$
	Std. dev.	**4.41E − 2**	$3.27E+0$	$1.45E+0$	$3.27E-2$	$4.01E-1$
DTLZ2	Average	$6.17E-2$	$5.20E-2$	$6.15E-2$	$5.78E-2$	**5.95E − 2**
	Std. dev.	$5.23E-3$	$5.61E-3$	$9.19E-3$	$1.79E-3$	**4.90E − 3**
DTLZ3	Average	$7.71E+0$	$2.56E+0$	$6.63E+1$	$1.70E+1$	**1.32E + 0**
	Std. dev.	$3.18E+0$	$5.06E+0$	$2.89E+1$	$5.32E-1$	**2.20E + 0**
DTLZ4	Average	$5.44E-2$	**4.426E − 2**	$4.97E-2$	$5.17E-2$	$6.12E-2$
	Std. dev.	$1.36E-2$	**4.76E − 3**	$3.12E-2$	$2.15E-2$	$4.87E-3$
DTLZ5	Average	$1.04E-2$	$1.32E-2$	$1.38E-2$	$1.52E-2$	**9.79E − 3**
	Std. dev.	$1.18E-3$	$2.12E-3$	$1.06E-3$	$2.91E-3$	**8.04E − 4**
DTLZ6	Average	**1.13E − 2**	$2.793E-1$	$1.38E-2$	$2.19E-2$	$1.22E-2$
	Std. dev.	**8.87E − 4**	$5.09E-2$	$1.01E-1$	$2.50E-2$	$1.25E-3$
DTLZ7	Average	$7.03E-2$	$4.55E-2$	$5.97E-1$	$6.48E-2$	**3.56E − 2**
	Std. dev.	$6.89E-3$	$1.63E-2$	$2.65E-2$	$8.90E-3$	**2.33E − 3**

Table 7: Statistical results of IGD obtained by different methods for different test functions.

Test functions	IGD	NSGA-II	SPEA-II	MOPSO	NSGA-III	IMOPSO
ZDT1	Average	$1.24E-2$	$3.82E-2$	$9.60E-1$	$2.84E-2$	**4.87E − 3**
	Std. dev.	$2.03E-3$	$4.08E-2$	$2.39E-1$	$6.75E-3$	**2.15E − 4**
ZDT2	Average	**2.02E − 2**	$9.61E-2$	$1.87E+0$	$5.26E-2$	$4.54E-2$
	Std. dev.	**1.63E − 2**	$7.66E-2$	$3.85E-1$	$3.32E-2$	$1.24E-1$
ZDT3	Average	$1.10E-2$	$3.70E-2$	$1.047E+0$	$2.34E-2$	**5.46E − 3**
	Std. dev.	$5.72E-3$	$2.70E-2$	$2.05E-1$	$5.57E-3$	**2.19E − 4**
ZDT4	Average	$1.85E-1$	**2.21E − 1**	$1.90E+1$	$6.04E-1$	$2.66E+0$
	Std. dev.	$1.25E-1$	**1.32E − 1**	$8.20E+0$	$2.33E-1$	$2.30E+0$
ZDT6	Average	$6.14E-2$	$2.68E-2$	$1.30E+0$	$2.82E-1$	**4.29E − 3**
	Std. dev.	$2.44E-2$	$9.81E-3$	$1.91E+0$	$1.20E-1$	**1.97E − 4**
DTLZ1	Average	$2.69E-1$	$1.65E+0$	$1.19E+1$	$2.51E-1$	**1.58E − 1**
	Std. dev.	$3.14E-1$	$1.33E+0$	$4.63E+0$	$2.10E-1$	**1.44E − 1**
DTLZ2	Average	$6.92E-2$	$6.72E-2$	**5.50E − 2**	$1.04E-1$	$7.2E-2$
	Std. dev.	$1.76E-3$	$3.72E-3$	**2.46E − 4**	$1.02E-2$	$7.2068E-2$
DTLZ3	Average	$8.33E+0$	$7.76E+1$	$1.76E+2.1$	$1.25E+1$	**7.42E + 0**
	Std. dev.	$5.65E+0$	$2.80E+1$	$5.11E+1$	$4.41E+0$	**5.20E + 0**
DTLZ4	Average	$9.97E-2$	$7.14E-2$	$3.75E-1$	$2.01E-1$	**6.55E − 2**
	Std. dev.	$1.20E-1$	$1.95E-3$	$1.86E-1$	$2.27E-1$	**1.73E − 3**
DTLZ5	Average	**5.98E − 3**	$1.23E-2$	$1.31E-2$	$1.52E-2$	$6.62E-3$
	Std. dev.	**3.12E − 4**	$2.24E-3$	$4.28E-3$	$1.71E-3$	$5.53E-4$
DTLZ6	Average	**6.39E − 3**	$1.76E-2$	$3.178E+0$	$5.56E-2$	$6.87E-3$
	Std. dev.	**2.32E − 3**	$2.29E-2$	$8.43E-13$	$1.33E-1$	$7.20E-4$
DTLZ7	Average	$2.06E-1$	$4.43E+0$	$1.23E-1$	$1.22E-1$	**9.78E − 2**
	Std. dev.	$2.69E-1$	$2.04E-1$	$1.21E+0$	$7.83E-2$	**1.05E − 2**

evaluation index GD, which is the convergence performance of the algorithm; Table 5 gives the numerical experiment result of the evaluation index SP, which is the algorithm distribution index; Table 6 gives the numerical experiment result of the evaluation index IGD, which represents the algorithm comprehensive performance. It can be seen from Table 4 that, compared with the other compared algorithms, in the test functions, ZDT, IMOPSO has obtained the best results 4 times, and NSGA-III has obtained the best results once; in the test functions, DTLZ, IMOPSO has obtained the

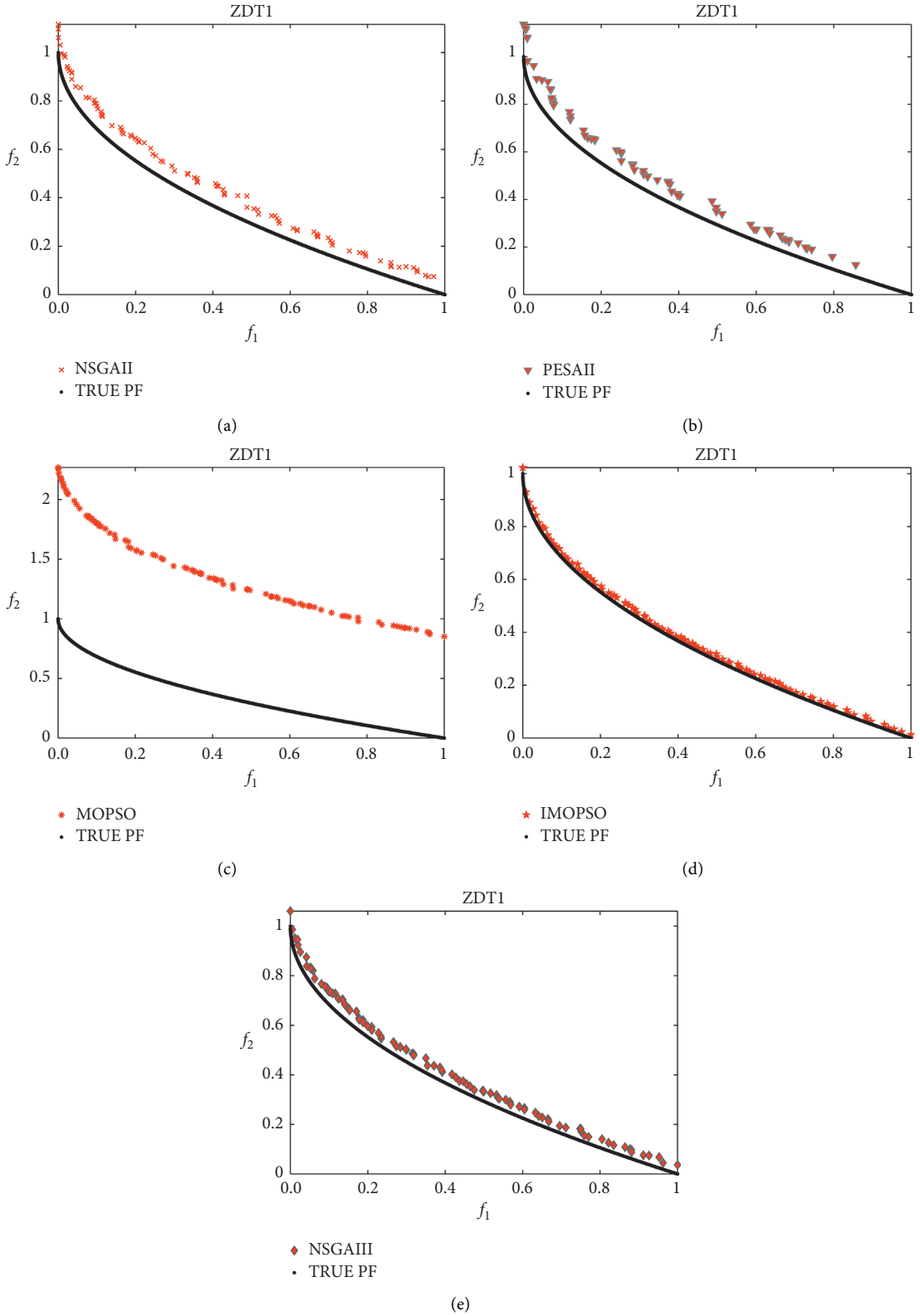

Figure 2: ZDT1 test results.

best results 5 times and NSGA-III has obtained the best results 2 times. It can be concluded that the convergence of IMOPSO is superior to those of the other algorithms used for comparison, and, at the same time, it can be further concluded that the learning factor adjustment rules adopted in this paper have effectively improved the convergence of

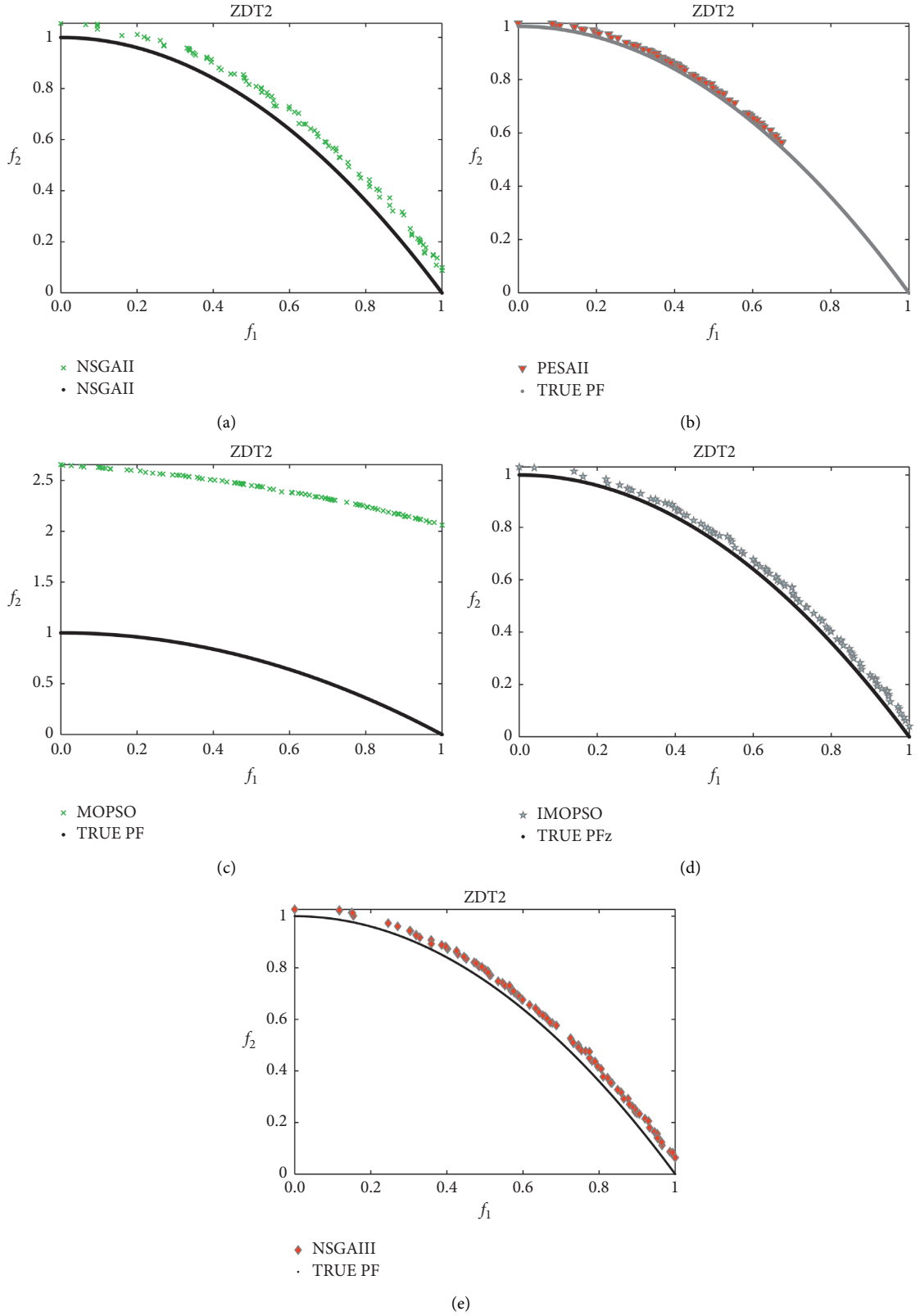

Figure 3: ZDT2 test results.

the IMOPSO. The numerical experiment results of the algorithms distribution are shown in Table 5. In the test functions, ZDT, IMOPSO obtained the best results 5 times; in the test functions, DTLZ, IMPSO obtained the best results 4 times, NSGA-II obtained the best results 2 times, and SPEA-II obtained the best result once. By analyzing the

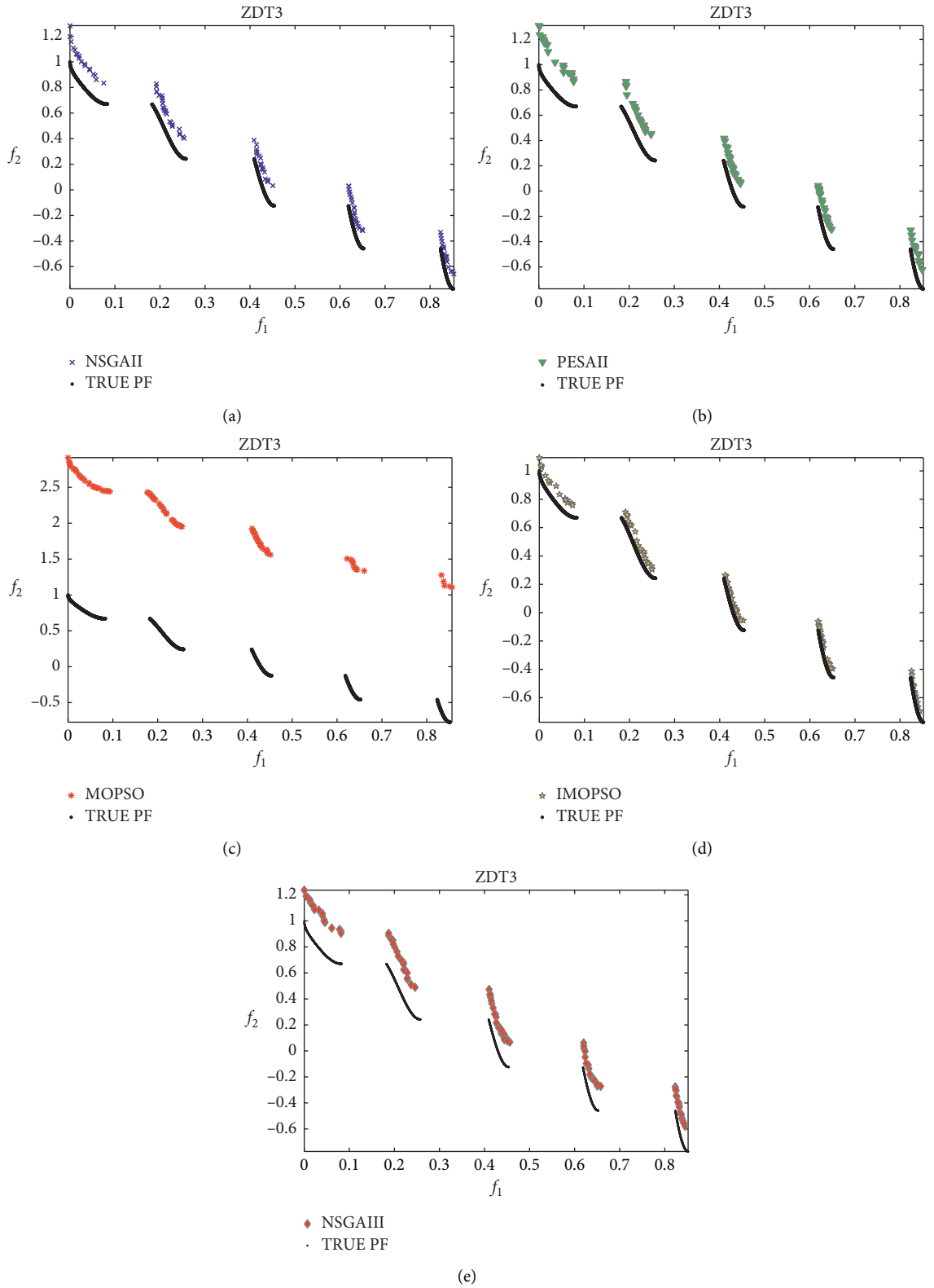

Figure 4: ZDT3 test results.

numerical results in Table 5, it can be seen that the distribution of the algorithm can be effectively improved by introducing time-varying Gaussian mutation strategy into IMOPSO. By analyzing Table 6, it can be seen that, in the test functions, ZDT, IMOPSO has obtained the best results 3 times, and NSGA-II and SPEA-II have obtained the best results once, respectively; in the test functions, DTLZ, IMOPSO has obtained the best results 4 times, and NSGA-II

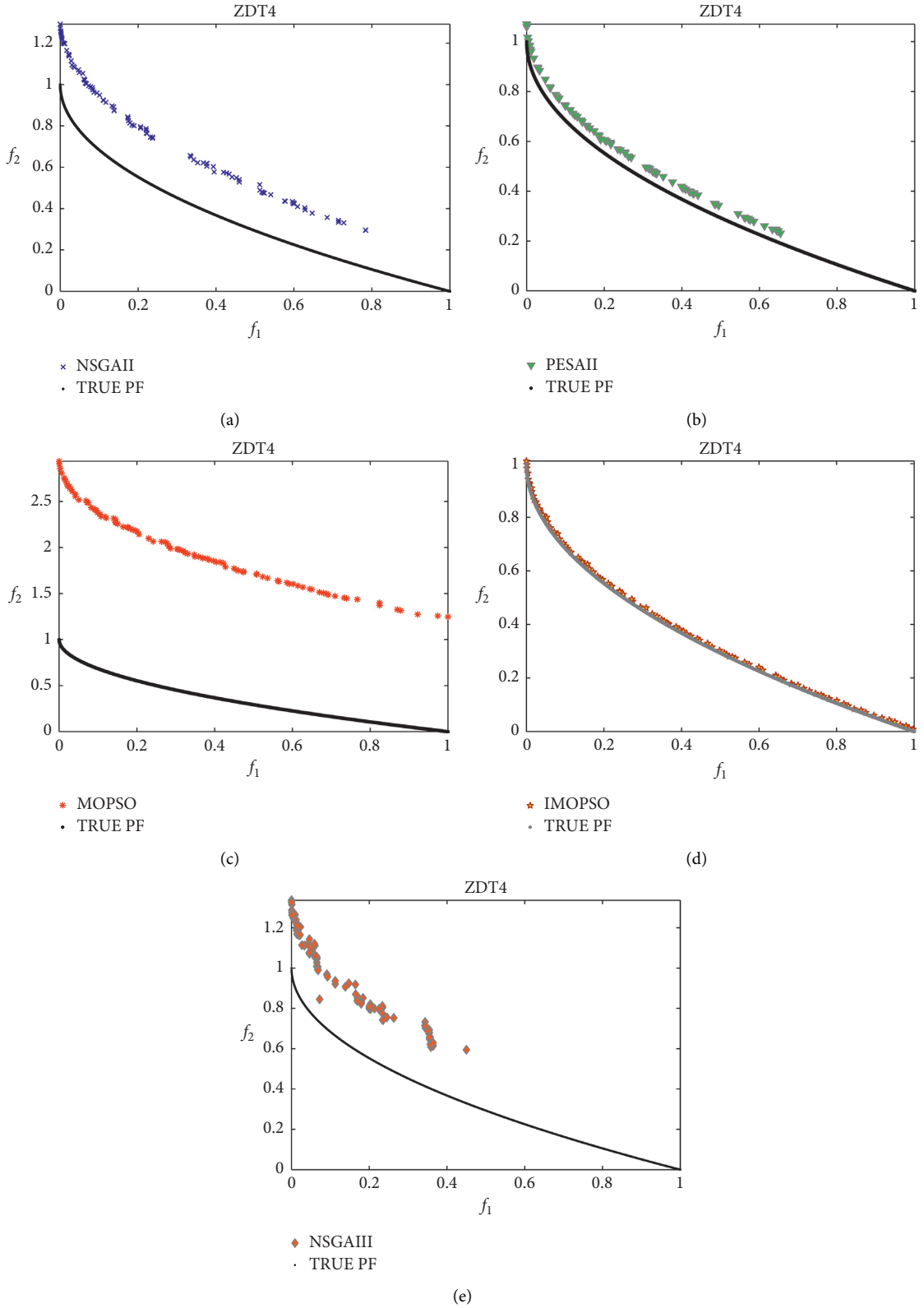

Figure 5: ZDT4 test results.

and MOPSO obtained the best results 2 times and once, respectively, so, on the whole, IMOPSO has better comprehensive performance (Table 7).

In order to more intuitively show the characteristics of each optimization problem in this article and the status of each algorithm in solving the optimization problems,

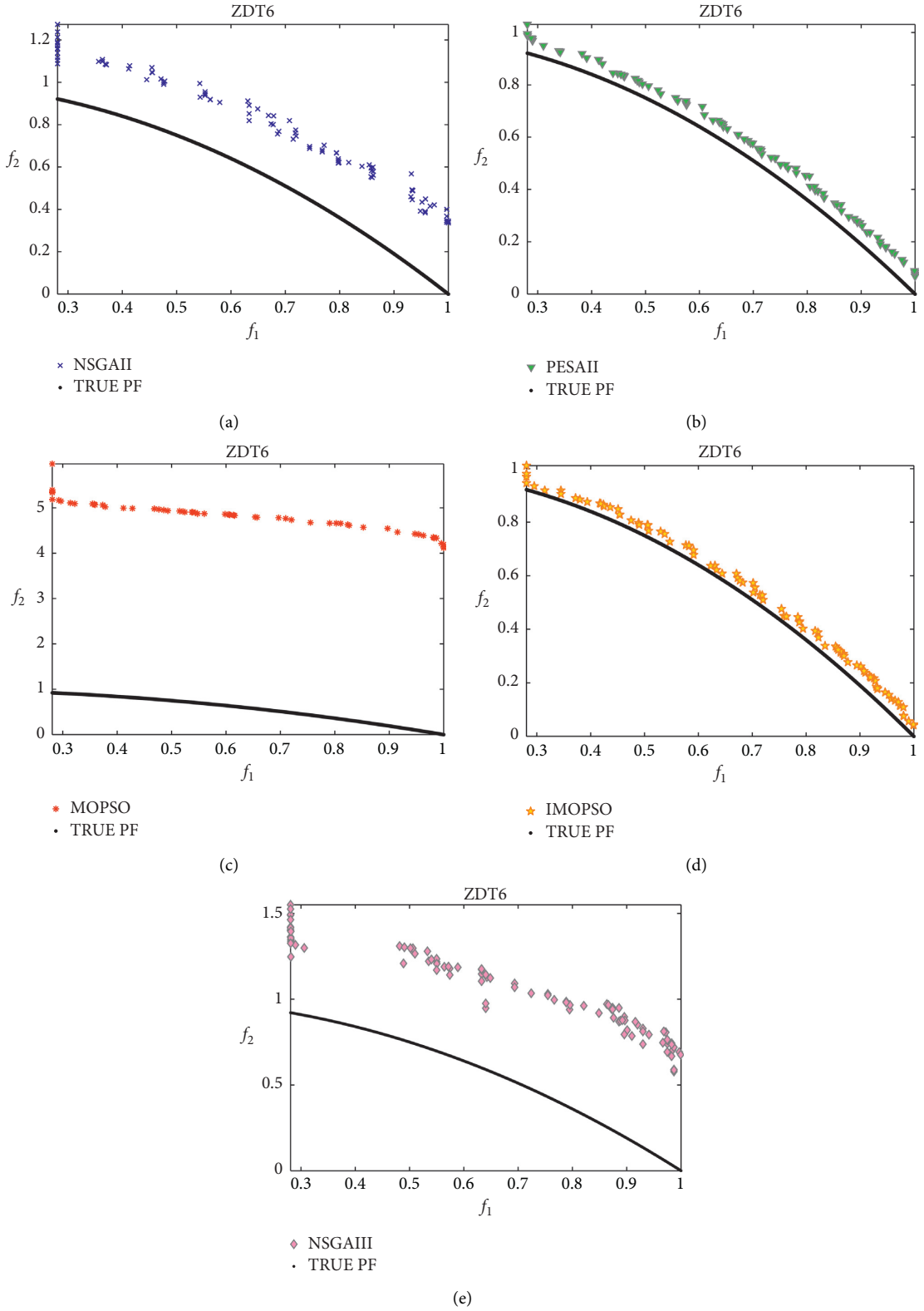

Figure 6: ZDT6 test results.

Figures 2–13 show the corresponding relationship between the optimal solutions obtained by each algorithm and the true PF. In Figures 2–6, we can intuitively draw the following conclusions: In the test functions, ZDT, except that the convergence of IMOPSO in ZDT3 is worse than that of NSGA-III, IMOPSO is better than the other compared algorithms in terms of convergence and distribution. In addition, in the test functions, DTLZ, the

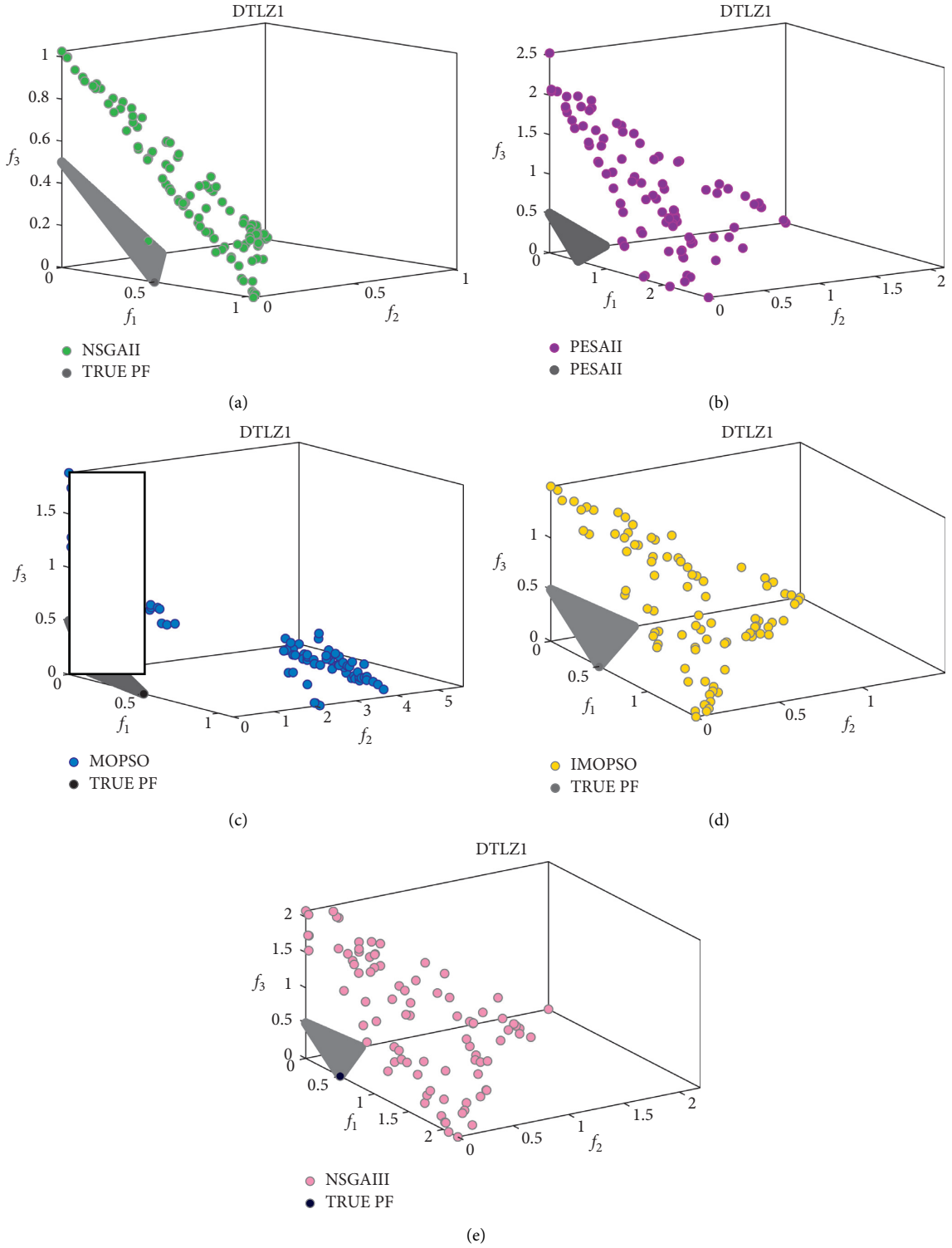

Figure 7: DTLZ1 test results.

convergence of IMOPSO in DTLZ2 and DTLZ4 is worse than that of NSGA-III, but it is better than the other compared algorithms in other test functions; in terms of SP, NSGA-II is better than IMOPSO in DTLZ1 and DTLZ6, and SPEA-II is better than IMOPSO in DTLZ6, but, in the remaining DTLZ test functions, the performance of IMOPSO is better than those of the other compared algorithms.

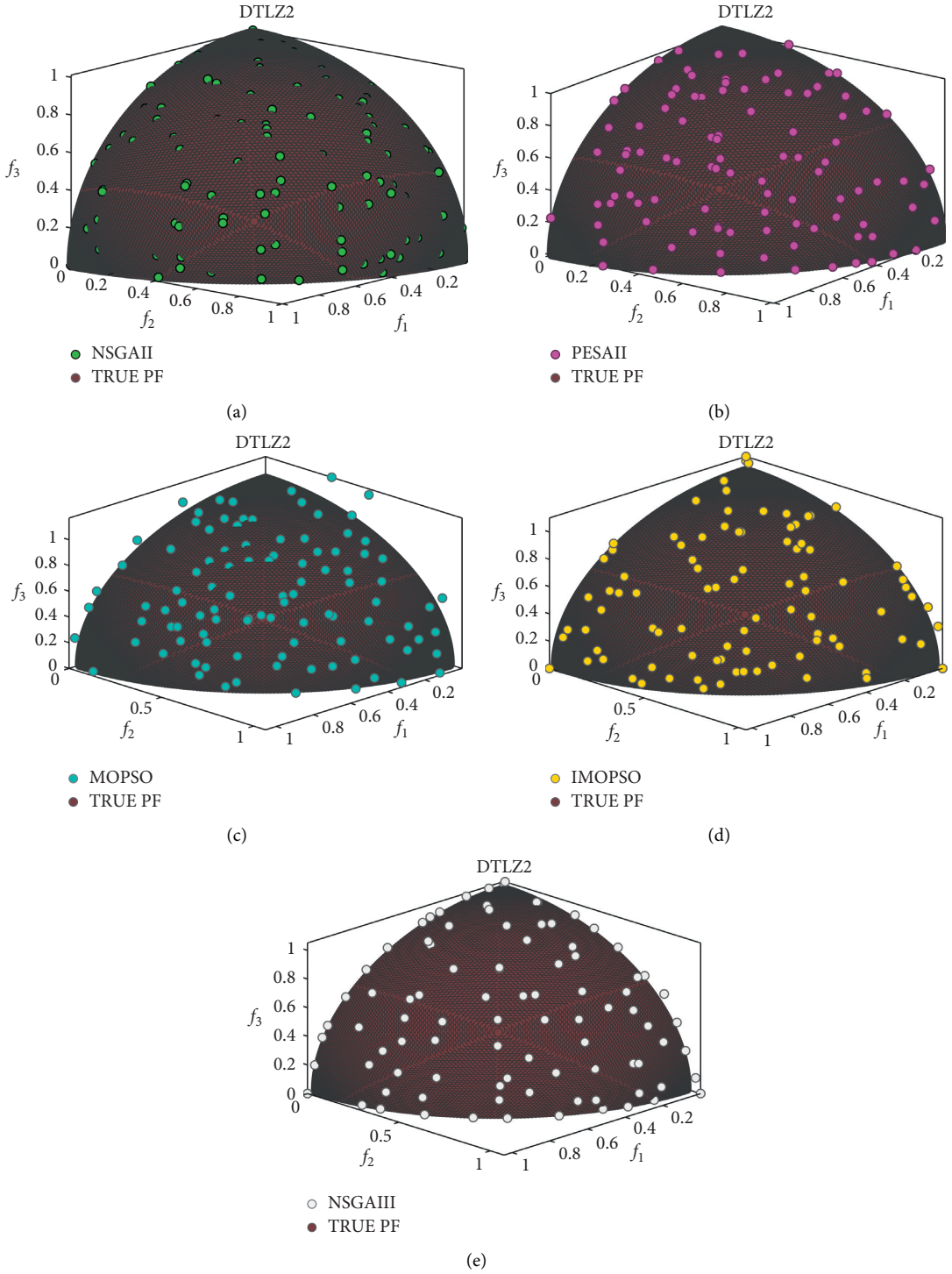

Figure 8: DTLZ2 test results.

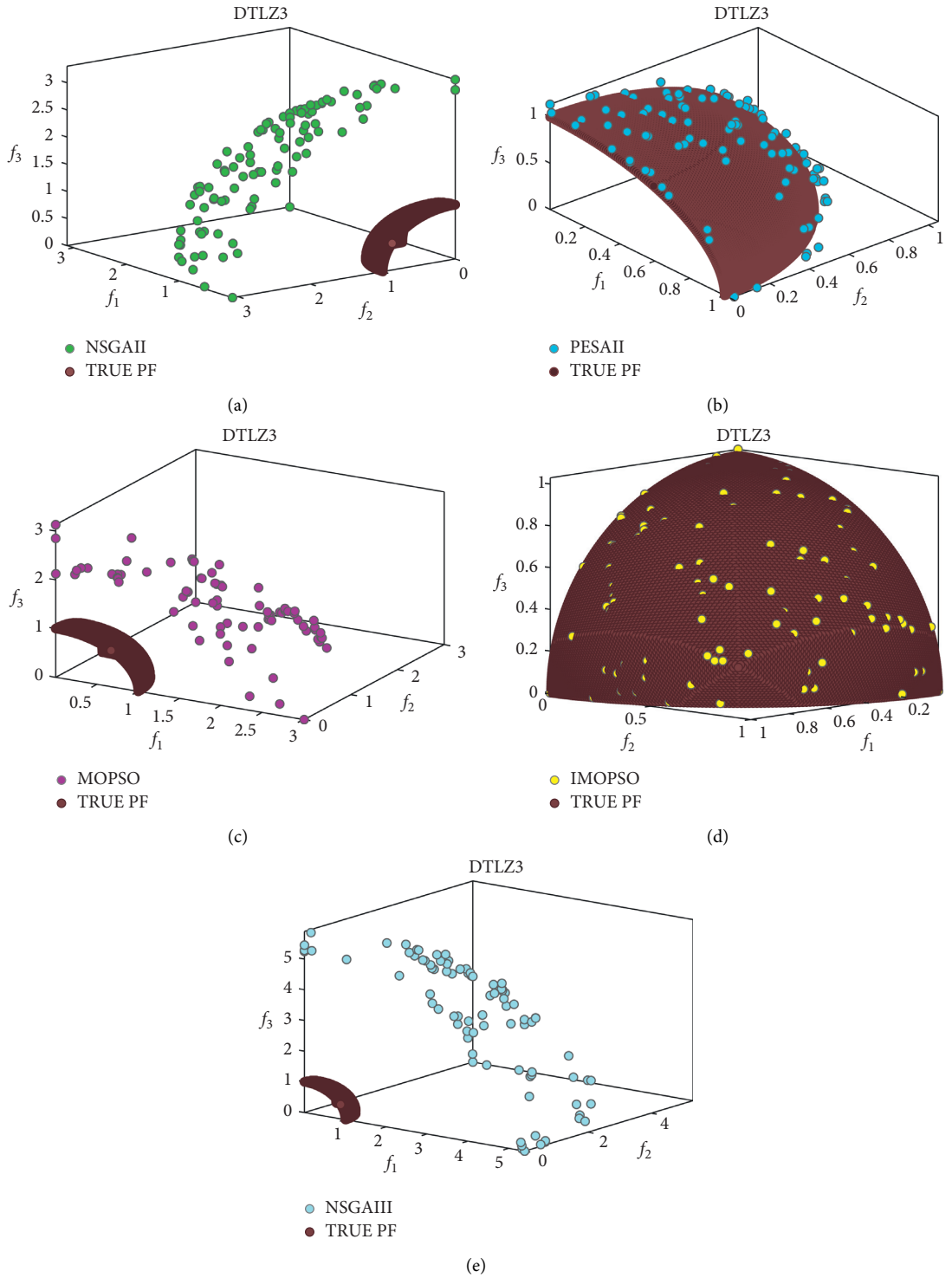

Figure 9: DTLZ3 test results.

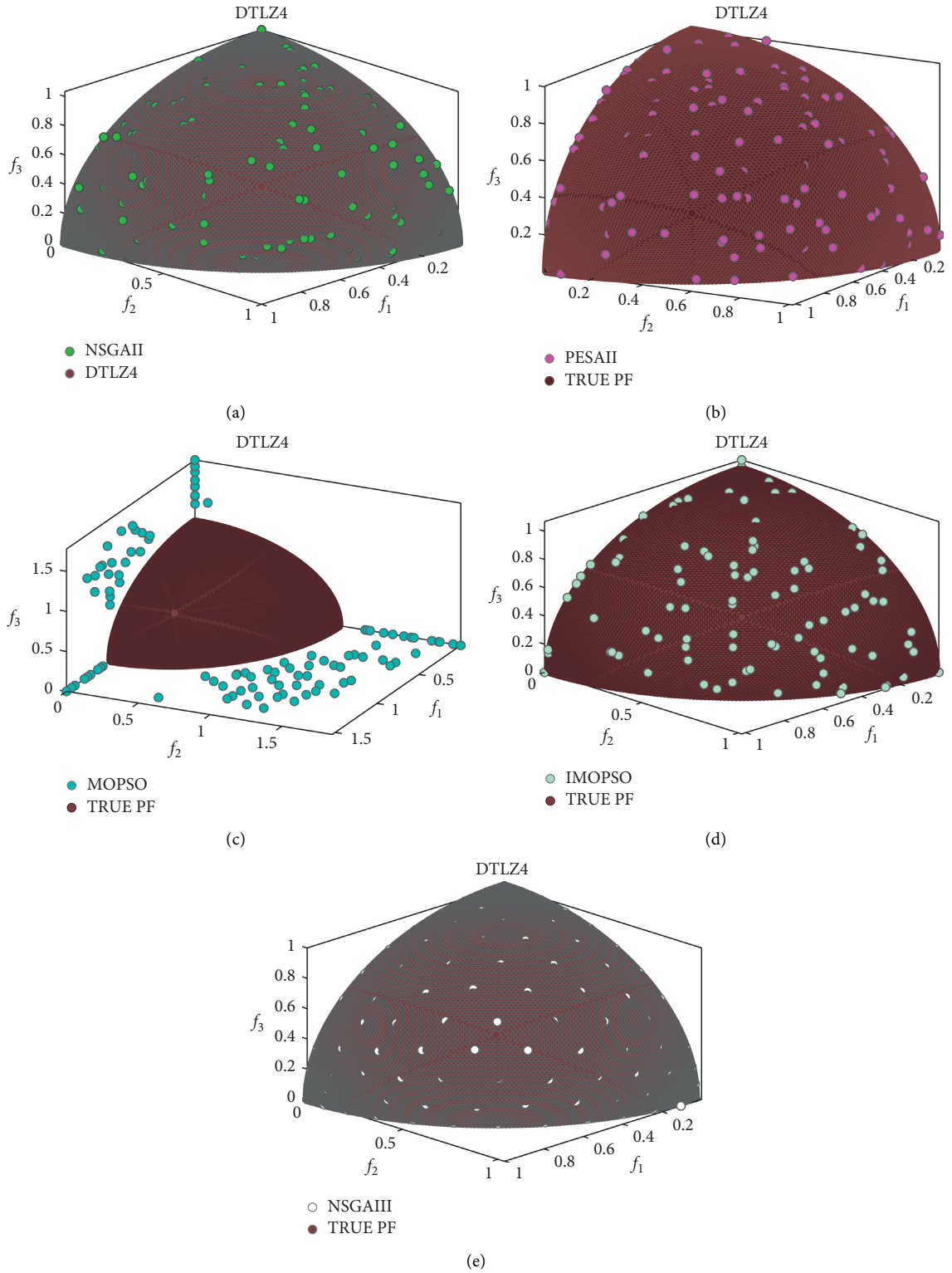

Figure 10: DTLZ4 test results.

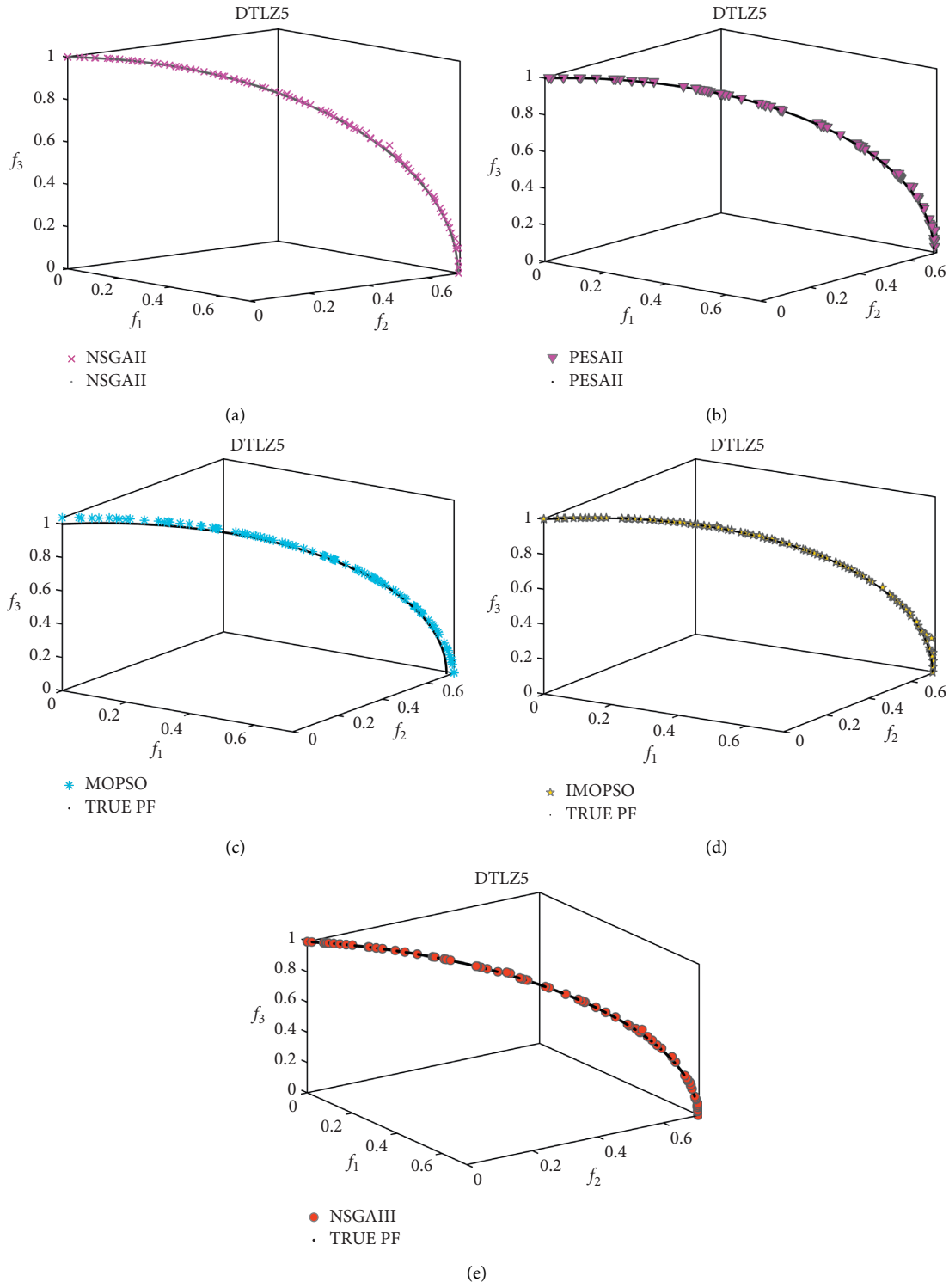

Figure 11: DTLZ5 test results.

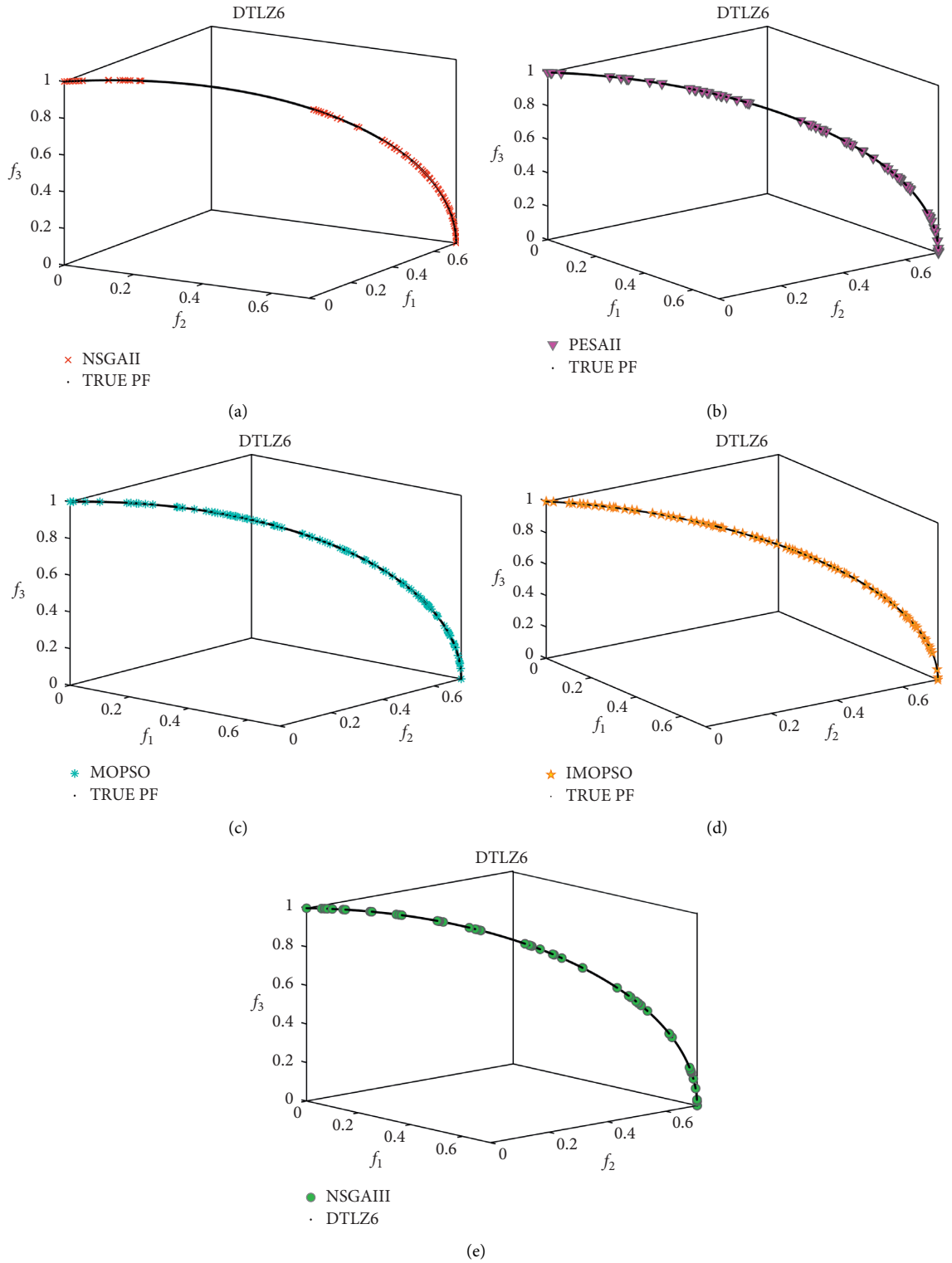

Figure 12: DTLZ6 test results.

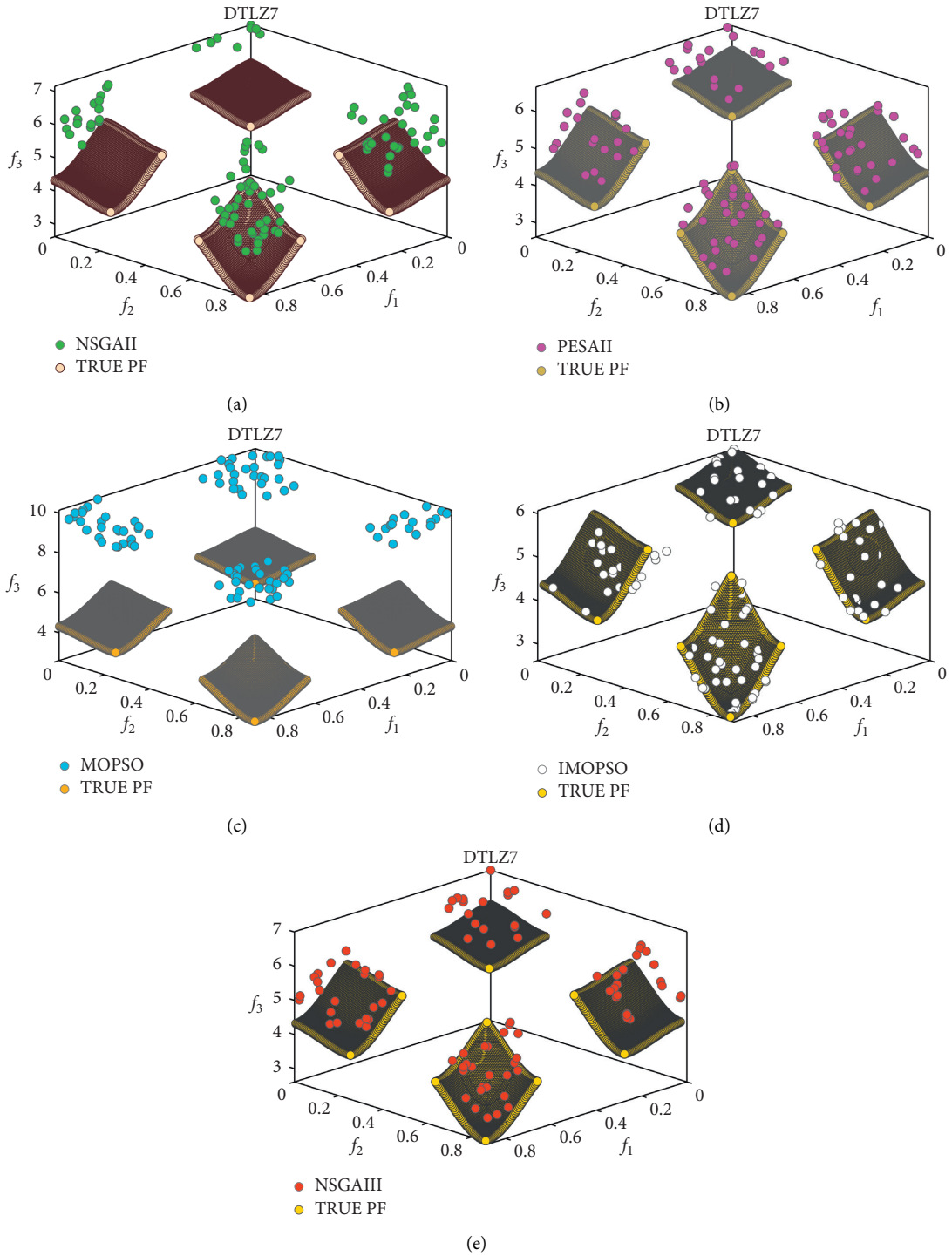

Figure 13: DTLZ7 test results.

7. Conclusions and Prospects

In order to improve MOPSO prone to premature convergence, poor distribution, and so forth, in solving multiobjective optimization problems, this paper proposes an improved multiobjective particle swarm optimization (IMOPSO) algorithm. In this paper, by introducing a competition mechanism strategy in IMOPSO and dynamically adjusting the inertia weight and learning factor of the algorithm, the convergence performance of the algorithm is effectively improved. In addition, in order to solve the shortcomings of insufficient distribution of the algorithm in the optimization process, time-varying Gaussian mutation is cited in the algorithm to increase the diversity of the algorithm and improve its distribution. Through the analysis of numerical experiment results, in the experimental results of the test functions, ZDT, and the test functions, DTLZ, as a whole, the convergence and distribution of the algorithm are better than those of the other compared algorithms. Finally, in the results of the comparison of the comprehensive performances of the algorithms, the comprehensive performance of the IMOPSO algorithm is also the best among the compared algorithms.

It should be pointed out that this article only conducts the numerical experiment analysis of the benchmark function, and its performance in actual multiobjective optimization problems needs further verification and analysis, such as multiobjective job shop scheduling problems and redundancy allocation problem with cold-standby strategy. This will be our next research works.

Data Availability

At present, these raw data cannot be shared because which forms part of an ongoing study.

Conflicts of Interest

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

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grants 11961001 and 61561001), the Construction Project of First-Class Subjects in Ningxia Higher Education (NXY LXK2017B09), and the major proprietary funded project of North Minzu University (ZDZX201901).

References

[1] L.-C. Chang and F.-J. Chang, "Multi-objective evolutionary algorithm for operating parallel reservoir system," *Journal of Hydrology*, vol. 377, no. 1-2, pp. 12–20, 2009.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Inc., New York, NY, USA, 1999.

[3] K. Deb, S. A. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm for multi-objective optimization," in *Proceedings of the Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems*, Berlin, Germany, 2001.

[5] C. A. C. Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, Honolulu, HI, USA, May 2002.

[6] S.-J. Tsai, T.-Y. Sun, C.-C. Liu, S.-T. Hsieh, W.-C. Wu, and S.-Y. Chiu, "An improved multi-objective particle swarm optimizer for multi-objective problems," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5872–5886, 2010.

[7] M. G. Mnif and S. Bouamama, "A new multi-objective firework algorithm to solve the multimodal planning network problem," *International Journal of Applied Metaheur Computing*, vol. 11, 2020.

[8] A. Mellouli, R. Mellouli, and F. Masmoudi, "An innovative genetic algorithm for a multi-objective optimization of two-dimensional cutting-stock problem," *Applied Artificial Intelligence*, vol. 33, no. 5–8, pp. 531–547, 2019.

[9] M. Ali, P. Siarry, and M. Pant, "An efficient differential evolution based algorithm for solving multi-objective optimization problems," *European Journal of Operational Research*, vol. 217, no. 2, pp. 404–416, 2018.

[10] X. Zhou, H. Wang, W. Peng et al., "Solving multi-scenario cardinality constrained optimization problems via multi-objective evolutionary algorithms," *Ence China Information Ences*, vol. 62, no. 9, 2019.

[11] D. E. C. Vargas, A. C. C. Lemonge, H. J. C. Barbosa, and H. S. Bernardino, "Differential evolution with the adaptive penalty method for structural multi-objective optimization," *Optimization and Engineering*, vol. 20, no. 1, pp. 65–88, 2019.

[12] Y. Sun and Y. Gao, "A multi-objective particle swarm optimization algorithm based on Gaussian mutation and an improved learning strategy," *Mathematics*, vol. 7, no. 2, 2019.

[13] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[14] L. Zhang, "Optimization design and application of improved multi-objective particle swarm optimization algorithm," *Journal of Radio Science*, vol. 26, no. 4, pp. 789–795, 2011.

[15] E. Zhang, "A class of interval multi-objective particle swarm optimization algorithm," *Control and Decision-Making*, vol. 29, no. 12, pp. 2171–2176, 2014.

[16] X. Tao, "A multi-objective optimization algorithm for combined particle swarm optimization and differential evolution," *Computer Simulation*, vol. 10, no. 4, pp. 313–316, 2013.

[17] W. Li and X. Zhang, "An improved multi-objective particle swarm optimization algorithm based on Pareto solution," *Computer Simulation*, vol. 12, no. 5, pp. 96–99, 2010.

[18] H. Ni, "Adaptive dynamic recombinant multi-objective particle swarm optimization algorithm," *Control and Decision*, vol. 30, no. 8, pp. 1417–1422, 2015.

[19] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys*, vol. 45, no. 3, p. 35, 2013.

[20] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multi-objective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[21] L. Deming, *Multi-Objective Intelligent Optimization Algorithm and Its Application*, Science Press, Beijing, China, 2009.
[22] V. Kumar and S. Minz, "Multi-objective particle swarm optimization: an introduction," *Smart Computing Review*, vol. 4, no. 5, pp. 335–353, 2014.
[23] M. J. Reddy and D. N. Kumar, "Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation," in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, Vancouver, Canada, June 2007.
[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, November 1995.
[25] Y. Yusoff, M. S. Ngadiman, and A. M. Zain, "Overview of NSGA-II for optimizing machining process parameters," *Procedia Engineering*, vol. 15, pp. 3978–3983, 2011.
[26] D. . Anna and H. Zidani, "Pareto front characterization for multiobjective optimal control problems using Hamilton-Jacobi approach," *SIAM Journal on Control and Optimization*, vol. 57, no. 6, pp. 3884–3910, 2019.
[27] M. Daneshyari and G. G. Yen, "Cultural-based multiobjective particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 2, p. 553, 2011.
[28] D. A. V. Veldhuizen and G. B. Lamont, *Evolutionary Computation and Convergence to a Pareto Front*, Stanford University California, Stanford, CA, USA, 1999.
[29] W. Shuai, L. Xiaohui, and H. Xiaomin, "Multi-objective optimization of reservoir OOD dispatch based on MOPSO algorithm," in *Proceedings of 8th International Conference on Natural Computation*, Sichuan, China, May 2012.
[30] A. Konak, D. Coit, and E. Smith, "Multi-objective optimization using genetic algorithms, a tutorial," *Reliability Engineering and System Safety*, vol. 91, no. 9, pp. 92–107, 2006.
[31] H. Wang, G. Grary, and X. Zhang, "Multi-objective particle swarm optimization based on paretoentropy," *Journal of Software*, vol. 25, no. 5, pp. 1025–1050, 2014.
[32] M. W. Mkaouer, M. Kessentini, and A. Shaout, "Many-objective software remodularization using NSGA-III," *ACM Transactions on Software Engineering and Methodology*, vol. 24, no. 3, pp. 1–45, 2015.