Hindawi
Scientific Programming
Volume 2022, Article ID 9448419, 21 pages
https://doi.org/10.1155/2022/9448419

Research Article

A Novel Multiobjective Particle Swarm Optimization Combining Hypercube and Distance

Xiaoli Shu,[1] **Yanmin Liu**,[2] **Qian Zhang**,[3] **and Meilan Yang**[3]

[1]*School of Data Science and Information Engineering, Guizhou Minzu University, Guiyang Guizhou 550025, China*
[2]*School of Mathematics, Zunyi Normal College, Zunyi Guizhou 563002, China*
[3]*School of Mathematics and Statistics, Guizhou University, Guiyang Guizhou 550025, China*

Correspondence should be addressed to Yanmin Liu; yanmin7813@163.com

Received 21 December 2021; Revised 13 February 2022; Accepted 15 February 2022; Published 14 April 2022

Academic Editor: Hangjun Che

The multiobjective optimization problems are a common problem in various fields in the real society. Therefore, solving the multiobjective optimization problems are one of the important problems studied by many researchers in recent years. From the research in recent years, it can be seen that there is still a lot of room for development of particle swarm optimization in solving multiobjective optimization problems. This paper proposes a novel multiobjective particle swarm optimization combining hypercube and distance, called HDMOPSO. The particle velocity update part in this paper uses a combination of hypercube and distance. In order to prevent the algorithm from falling into the local optimum, the part also uses the nonlinear decreasing opposite mutation strategy, which enables the particles to explore a more area. Finally, a control strategy is used for external archive to improve the convergence and diversity of the algorithm. The algorithm has been simulated in 22 test problems and compared with multiobjective particle swarm optimization algorithms (MOPSOs) and multiobjective evolutionary algorithms (MOEAs). The results show that the HDMOPSO can effectively improve the convergence and diversity, so it is an effective improvement.

1. Introduction

In real life, many problems are composed of multiobjective that conflict and influence each other. When an objective is found to be the best solution, it cannot guarantee that other objectives are also the best solution at the same time, but may lead to degradation. Scientists usually try to make these objectives reach the best state in an enclosed area, which is multiobjective optimization problems (MOPs).

In recent years, it has become one of the important methods to integrate biological information into metaheuristic algorithms to solve multiobjective optimization problems in evolutionary algorithms. Typical multiobjective evolutionary algorithms (MOEAs) are genetic algorithm [1], multiobjective particle swarm optimization [2], multiobjective bee colony algorithm [3], multiobjective ant colony algorithm [4], and multiobjective differential algorithm [5] and so on. The particle swarm optimization (PSO) is similar to the genetic algorithm. It is also an algorithm invented based on the behavior of the population in nature. Because of its simple principle, high search efficiency, fast convergence and so on, it is widely used in various fields of industrial production.

When using PSO to deal with MOPs, we should not only consider the common difficulties in the process of traditional multiobjective optimization, but also consider the problems targeted by PSO when applied to MOPs. There are four main problems: (1) The optimal particle selection strategy (i.e., how to select the "leader" particle to lead the entire population to quickly approach the Pareto front while retaining some individual information); (2) Mechanisms for maintaining diversity (i.e., how to guide particles out of the local optimal solution); (3) Convergence improvement means (i.e., how to improve the search efficiency while maintaining the diversity of the population when the external archive set increases sharply, and strengthen the advantage of the

algorithm in terms of convergence velocity); and (4) The balance method of diversity and convergence (i.e., how to dynamically coordinate the relationship between development and local search at different stages of the optimization process to obtain the best optimization results).

In order to solve the above four problems, scientists have proposed various solutions. Among them, how to choose the "leader" particles? Qingling Zhu et al. [6] proposed to select a leader from an external archive set and decompose the multiobjective optimization problems into a single optimization problem. This method greatly reduces the computational complexity of the algorithm and also improves the convergence velocity of the algorithm. The fast convergence speed and high convergence accuracy of the algorithm often lead to poor diversity performance. Dividing the population into multiple subgroups [7] and various maintenance strategies for external archive [8] can effectively improve the diversity of the algorithm. For the method of balancing diversity and convergence, a combination of multiple strategies [9] was required. Generally, combining the selection of the "leader" with the update strategy of the external archive can effectively balance the convergence and diversity of the algorithm. Because a good "leader" selection strategy can more efficiently guide the population to converge to the Pareto front, and a good external archive control strategy can better maintain the diversity of the population. However, Xingyi Zhang et al. [10] proposed a multiobjective particle swarm optimization with a competitive mechanism. This method was different from most multiobjective particle swarm optimization algorithms (MOPSOs) because it does not use external archive. It was more competitive in balancing diversity and convergence compared with most MOPSOs. Means for improving convergence, although the strategy of choosing a good "leader" can effectively improve the convergence of the algorithm, as the dimensionality increases, the convergence of the algorithm will be affected. Since the inertia weight can effectively affect the convergence of the algorithm in the scientific field, some assumptions about the inertia weight can effectively improve the convergence of the algorithm. For example, Peng Guang et al. [11] proposed a dynamic learning factor. Of course, it is not only the above methods that solve the multiobjective optimization problem, but also more extensive exploration in the scientific field. Therefore, the main contributions of this paper are as follows:

(1) A control strategy for external archive is proposed. In the continuous iterative update of the algorithm, the nondominant solutions produced will gradually increase, which not only increases the complexity of the algorithm, but also affects the convergence and diversity of the algorithm. Therefore, when the total number of nondominant solutions and newly generated nondominant solutions in the external archive exceed the preset threshold, the hypercube technology is used in the external archive to control the nondominant solutions within the preset threshold. First, the hypercube is created in an external archive. Then, a hypercube is created in the densest hypercube. Finally, a nondominant solution is randomly deleted from it. This cycle continues until the sum of the number of nondominant solutions and the number of newly generated nondominant solutions in the external archive is within the preset threshold. This method can effectively maintain the diversity of solutions and improve the convergence of algorithm.

(2) A nonlinear decreasing opposite mutation is proposed. This strategy performs nonlinear decreasing opposite mutation on the particles of the nondominant solution generated after each iteration update, which can effectively prevent the algorithm from falling into the local optimal solution. Using the method of nonlinear declining can well balance the global exploration and local exploration capabilities of the algorithm, and is an effective method to balance diversity and convergence.

(3) The combination of hypercube and distance method is proposed to set the particle velocity to update the social part. First, the hypercube established based on the objective function value of the nondominant solution in the external archive. Second, calculating the average value of the nondominant solutions set in each hypercube obtained. We call this value the "generalized position". This value is the learning position of the social part of the group. Finally, the Euclidean distance method allows each particle to learn from the generalized position closest to itself. This strategy is an effective means to improve the diversity and convergence of algorithms.

The rest of this paper is organized as follows: In Section 2, the relevant background knowledge of this paper is briefly introduced. Section 3 gives the details of the HDMOPSO. Section 4 verifies the performance of HDMOPSO by comparing with existing MOPSOs and MOEAs. Finally, Section 5 introduces the conclusions and future work of this paper.

2. Background

2.1. MOPs. MOP is an optimization problem composed of n-dimensional decision variables, m objective functions, and $P + Q$ constraints. MOPs can generally be transformed into a minimum problem, so the mathematical form of MOPs is expressed as follows:

$$\begin{aligned} &\operatorname{Min} y = F(x) \\ &\qquad = [f_1(x), f_2(x), \ldots, f_m(x)], \\ &g_p(x) \le 0, \quad p = 1, 2, \ldots, P, \\ &h_q(x) = 0, \quad q = 1, 2, \ldots, Q, \\ &\quad L_i \le x_i \le U_i, \quad i = 1, 2, \ldots n, \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)^T \in X \in R^n$ is the n-dimensional decision variable; $y = (y_1, y_2, \ldots, y_m)^T \in Y \in R^m$ is the m-dimensional objective variable; P is the number of inequality

constraints; Q is the number of equality constraints; and $[L_i, U_i]$ is the boundary of the i-th dimension of the particle.

In MOPs, the definitions of Pareto dominance, Pareto optimal solution, and Pareto optimal solution set are as follows.

Definition 1 (Pareto dominance [12]). For two decision vectors $x_u, x_v \in X$, x_v dominates x_u which is expressed as $x_v \prec x_u$, if and only if

$$\{\forall i \in \{1, 2, \ldots, n\}, f_i(x_v) \leq f_i(x_u)\} \wedge \{\exists j \in \{1, 2, \ldots, n\}, f_j(x_v) < f_j(x_u)\}. \quad (2)$$

Definition 2 (Pareto optimal solution [12]). $x* \in X$ is the Pareto optimal solution on X, if and only if

$$\exists x \in X, x \prec x^*. \quad (3)$$

Definition 3 (Pareto optimal solution set [12]). The set of all Pareto optimal solutions becomes Pareto optimal set ($P*$). The mathematical definition is as follows:

$$P^* = \{x \in X | \exists x' \in X, f_j(x') \leq f_j(x), (j = 1, 2, \ldots, m)\}. \quad (4)$$

Definition 4. (Pareto front (PF) [12]). All objective functions corresponding to nondominant solutions constitute the nondominant optimal objective domain, also known as Pareto front (PF). The mathematical definition is as follows:

$$PF = \{F(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_m(x^*)) | x^* \in P^*\}. \quad (5)$$

2.2. PSO. PSO was first proposed by Eberhart and Kennedy [13] in 1995, and its concept originated from the study of bird flock foraging behavior. Ven den Bergh [14] analyzed and proved the stability and convergence of PSO from a theoretical perspective. In 2002, coello et al. [15] applied the PSO to solve MOPs, called multiobjective particle swarm optimization (MOPSO). Imagine a scene where a group of birds randomly search for food in an enclosed area. There is a lot of food in this area, but all the birds do not know where the food is. They only know how far the food is from the current location. Therefore, the simplest and effective strategy is needed to find the food quickly and efficiently. PSO uses a massless particle to simulate birds in a flock of birds. Particles have only two attributes, velocity and position. Velocity represents the speed of the particle's movement, and position represents the direction of the particle's movement. The particle uses the following equation to update its velocity and position:

$$v_i(t+1) = wv_i(t) + c_1 r_1 (pbest_i(t) - x_i(t)) + c_2 r_2 (gbest_i(t) - x_i(t)). \quad (6)$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (7)$$

The right side of equation (6) consists of three parts. The first part is the inheritance of the particle to the previous velocity, which represents the particle's trust in its own motion state, and is the inertial motion of the particle's previous velocity. Among them, w is called the inertia weight, and its value is non-negative. When the value is large, the global optimization ability is strong, and the local optimization ability is weak. When the value is small, the global optimization ability is weak, and the local optimization ability is strong. The second part is the self-cognition part of the particle, which means that the particle's thinking about itself is derived from the summary of the past experience so as to implement the next behavior decision. Here $pbest_i$ represents the optimal solution found by the i-th particle in the historical experience, which is called "individual extreme value". The third part is the social part, which represents the process of information sharing and mutual cooperation between particles. Particles can make appropriate adjustments to their flight directions by perceiving their experienced companions. In the standard PSO, $gbest_i$ represents the optimal solution found so far by the entire group (i.e., the global optimal). In addition, in equation (6) and (7), v_i represents the velocity of the i-th particle; r_1, r_2 are random numbers between (0, 1). x_i represents the current position of the i-th particle; c_1, c_2 represent acceleration factors.

2.3. The Existing MOPSOs. Since PSO was proposed to solve MOPs in 2002, many scholars have become more and more interested in it. In the past twenty years, there have been many MOPSOs. Below, we'll take a closer look at some of the existing MOPSOs.

Coello et al. [15] first proposed using PSO to solve MOPs. Pareto front was proposed in this algorithm to select the nondominant solution in the current solution and save it in the archive. The global optimal particle was then determined from the archive. Although compared with traditional non-dominated sorting genetic algorithm II (NSGAII) [16], this algorithm has certain advantages. But the diversity of solutions and convergence performance can be further improved. Nebro et al. [17] proposed a velocity constrained multiobjective particle swarm optimization. In this algorithm, the velocity contraction program was used to prevent

the group explosive in the process of particle movement. Literature [7, 8] proposed different methods for the selection of global optimal particles on the basis of [12]. In reference [7], the optimal solution was selected according to the way of using crowding distance in the nondominant solution in the external archive. In reference [8], the selection of the global optimal solution was based on the grid technology in the archive, and the optimal solution was selected according to the grid distance. Experimental results show that literature [7, 8] was superior to the algorithm proposed in literature [12] in both diversity and convergence.

Zapotecas Martinez S et al. [18] proposed a method that relied entirely on decomposition (dMOPSO). The decomposition method was similar to the algorithm proposed by Qingfu Zhang et al. [19], which was to decompose multiobjective optimization problems into several single-objective optimization problems. It can also be combined with other decomposition methods in dMOPSO. In the literature [18], a set of global optimal solutions can give the set value of the best value of all the sub-problems to update the position of the particle. Of course, the algorithm also has some limitations because this algorithm mainly solves continuous and unconstrained multiobjective optimization problems.

Researchers have also proposed other methods to improve the convergence and diversity of the algorithm. The biggest difference between the algorithm proposed by Xingyi Zhang et al. [10] and other MOPSOs was that it does not use external archives. A competition mechanism was used in the algorithm, which was composed of three parts: nondominant solution selection, pair competition and particle learning. Experimental results showed that this algorithm was more competitive than the existing MOPSO algorithm.

The above algorithms in the paper were almost aimed at some continuous, discontinuous, concave, and convex properties of MOPs. MOPs are more than that. Ying Hu et al. [20] proposed a feature selection problem to solve fuzzy costs, in which fuzzy advantage relations were used instead of traditional Pareto dominance and fuzzy crowded distances were used to update files. A set of nondominant solutions can be calculated by using defined fuzzy concepts applied to an algorithm. This was a highly competitive approach to the problem of feature selection. The algorithm proposed by Zhang Yong et al. [21] was a powerful tool for optimizing building energy preferences. A perturbation strategy was proposed and control parameters such as inertia weight and acceleration were deleted compared with the traditional MOPSOs. The traditional MOPSOs were more sensitive to control parameters.

2.4. Opposite Direction. In 2005, Hamid R. Tizhoosh [22] proposed the concept of the opposite direction. Some researchers used the opposite direction strategy for initialization, two sets of positions were generated when the position was initialized. One set was randomly generated, and the other set was generated in the opposite direction based on the existing position. Literature [23] used the opposite direction for learning strategies. The reason for the opposite direction is because in a given environment, our search direction may be opposite to the direction of the optimal solution. If we continue to search in the wrong direction, the algorithm will not get better results. So we can observe all directions in the search process. Sometimes it may be advantageous to look in the opposite direction. If the opposite direction is advantageous, then the first step in finding the opposite direction is as follows:

Definition 5 (opposite direction [22]). x is a real number defined in a certain interval, $x \in [L, U]$, if $\overline{x}$ is the opposite value of x, the definition of $\overline{x}$ is as follows:

$$\overline{x} = L + U - x. \tag{8}$$

Similarly, the opposite number can also be defined in the case of multiple dimensions.

Corollary 1 (see [22]). *There is $x = (x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n \in R$ and $x_i \in [L_i, U_i]$, then the opposite of $\overline{x}$ is defined by $x_1, x_2, \ldots, x_n$, which is defined as follows:*

$$\overline{x}_i = L_i + U_i - x_i \, (i = 1, 2, \ldots, n). \tag{9}$$

3. HDMOPSO

Research has shown that the choice of "leader" particles in PSO is crucial in recent years. The "leader" particle can chose the best solution the particle has found so far in single-objective optimization, but MOPs is not the same as single-objective optimization because multiobjective optimization is restricted by multiobjective and generates more than one nondominant solution each time. Therefore, a good "leader" selection strategy can effectively improve the convergence and diversity of the algorithm. In addition, Because PSO has a fast convergence speed, it may make the algorithm fall into local optimal in MOPs. The nondominant solution particles after each iteration are used to nonlinear reduce the opposite mutation so that the particles can explore more area in this paper. Finally, this paper also uses an external archive control strategy to improve the convergence and diversity of the algorithm. The details are introduced as follows.

3.1. Control Strategy for External Archive. The external archive is used to store the nondominant solutions after each iteration, and it is the candidate solutions, also called the solutions. When the external archive is empty, the solutions will directly enter the external archive. Because a new set of solutions is generated after each update population, this new set of solutions is compared to the historical solution in the external archive. Such behavior is called "elitism". It has three situations. (1) If the new solutions dominate the historical solutions in the external archive, the historical solutions will be deleted in the external archive. (2) If the new solution is dominated by the historical solutions in the external archive, the dominated new solutions will also be automatically discarded. (3) If the historical solutions in the external archive and the new solutions do not dominate each other,

the solutions in the external archive and the newly generated solutions remain unchanged. After judging the above three situations, the final solutions will be placed in an external archive. However, as the number of iterations increases, the nondominant solutions in the external archive gradually increase, which will increase the subsequent calculation work and also easily cause the algorithm to fall into a local optimal situation. Therefore, an external archive control strategy is needed to control the nondominant solutions within a certain threshold. The convergence and diversity of the algorithm are improved at the same time.

This paper uses hypercube technology to control external archive within a certain threshold. The main problem of external archive control is that the particles in the external archive are all nondominant solutions and do not dominate each other. There may be very small nondominant solutions that prevent the algorithm from maintaining diversity. Therefore, this paper mainly builds hypercube based on the value of the objective function value of the nondominant solutions, from which the densest hypercube is selected to create a hypercube, and then one is randomly deleted. The cycle continues until all new solutions after elitism are put into the external archive. Figure 1 shows the control strategy of the external archive when the objective number is two. Begin to establish fixed hypercube according to the objective function value of the current nondominant solutions, and then select a hypercube with high density each time to establish an adaptive hypercube again, and randomly delete one from the hypercube with high density. The above method can effectively delete the denser nondominant solutions in the external archive and increase the diversity of the algorithm. In addition, each time fixed hypercube is established based on the objective function value of the current nondominant solutions, and then an adaptive hypercube is established in the densest hypercube, which greatly reduces the computational complexity of the algorithm.

3.2. Nonlinear Decreasing opposite Mutation Strategy. At present, mutation strategy is often used to improve the performance of the algorithm, but the mutation used in this paper is a combination of exploration and convergence. It differs from most of the mutations proposed by the researchers, because most of the current mutation is to increase the particle's exploration ability and the ability to jump out of the local optimal solution. For example, the mixed mutation and jump mutation proposed in the literature [24]. However, the opposite mutation of nonlinear decreasing in this paper is that the current nondominant solution is not guided by better particles in the social part, so opposite search is carried out to prevent the algorithm from falling into the local optimal solution. In addition, the global search tends to local search in nonlinear decreasing way, which is beneficial to the convergence of the algorithm. So the mutation also has the ability to converge.

In the multiobjective particle swarm optimization, as the iteration increases, the algorithm converges quickly.

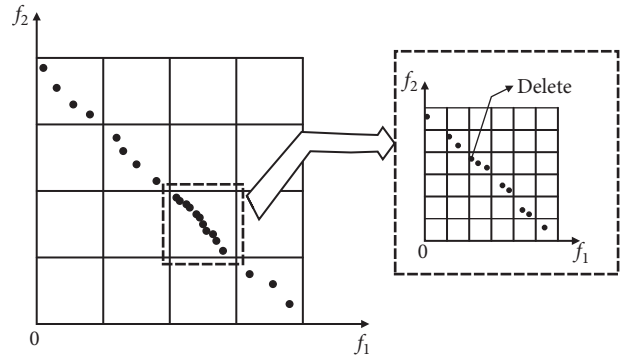

Figure 1: Control strategy for external archive.

According to the characteristics of the PSO, all particles are generally searched based on the particles of the nondominant solutions that have been generated. As a result, the particles currently in the nondominant solutions can only judge the next search direction based on the inheritance of the previous velocity and their own experience. If these particles have no particle leadership, the search may deviate from the right direction. If it continues to search in this way, it will not only affect the convergence velocity of the algorithm, but also affect the convergence of the algorithm. Therefore, this paper proposes a nonlinear decreasing opposite mutation strategy. If the particle loses the direction guided by the social part, it may be beneficial to search in the opposite direction. But it is definitely impossible for all particles to search in the opposite direction. Because all the particles search in the opposite direction, the population will gradually move away from the better solution and it is always part of the search process, which makes the algorithm unable to converge well. So, this paper adopts the opposite mutation of the particles that get new nondominant solutions after each iteration. Figure 2 shows mutation simulation that there are 15 particles in a certain area, and after a certain iteration, three nondominant particles are produced. As can be seen from the left image in Figure 2, there are some areas in this area that can be further explored. Therefore, we use the strategy of opposite mutation of nondominant solutions particles so that the particles can explore more areas, the same as 1, 2, and 3 in Figure 2. However, it is impossible to explore farther places all the time, which will hinder the convergence of the algorithm and may also deteriorate the performance of the algorithm. So we used nonlinear decreasing opposite mutation. In the early stage of the algorithm, the particles of the new nondominant solutions undergoing farther opposite mutation can better perform global exploration. With the increase of iterations, the algorithm also enters the final stage, because the algorithm is best to converge to the Pareto front, so it needs to be locally explored to the current nondominant solutions in the later stage. Therefore, a small range of opposite mutation is carried out at the end of the algorithm, and local exploration is carried out, so that the algorithm can gradually converge to the Pareto front. The pseudocode of the nonlinear decreasing opposite mutation search strategy is shown in Figure 3.

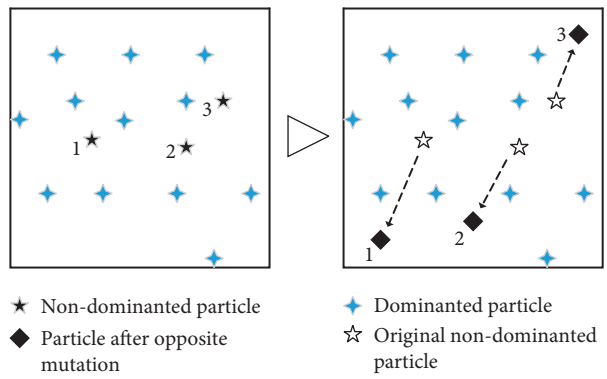

Figure 2: Mutation simulation.

```
% position=particle position
% gencount=current iteration
% maxgen=total number of iterations
% a=nonlinear decreasing factor
% rank(i)=1, i is the non-dominanted solution at this time
function Opposition-Mutation-Operator(position, gencount, U, L, rank, maxgen)
      a=(1-gencount/maxgen);
   for i=1 to n
       if rank(i)=1
           Position(i,:)=U(i,:)+L(i,:)-a*position(i,:);
       else
           position(i,:)=position(i,:);
       end if
   end for
end function
```

Figure 3: Pseudocode of nonlinear decreasing opposite mutation.

3.3. Setting the Social Part Based on the Method of Combining Hypercube and Distance. Figure 4 shows the flying direction of the particle in PSO. The symbols in the figure indicate the reference equation (6). It can be seen from Figure 4 that the particle i finally determines the next flight direction according to three parts. Since the social part of the guidance represents the optimal position that the group finds, the position has more resources, so $gbest_i$ is the key factor for the next flight direction of particle i. From the above analysis, we can know that actually refers to $gbest_i$ position. Since in the single-objective particle swarm optimization algorithm, there is only one optimal solution of the population after each iteration, so the effect of the particle learning of the population following the optimal solution in the population in the social part may be the best. However, there is more than one optimal solution in the multiobjective particle swarm optimization. If one optimal solution is used blindly to lead the particles to fly, it may cause the particles to converge in one direction and lose the diversity of the algorithm. At present, the most used strategy in this part is roulette-wheel, which uses a certain probability to make each particle choose a leader to lead itself, but this method is more random.

This paper avoids this randomness and proposes a method based on the combination of hypercube and distance to set the social part of the particle velocity update equation under external archive. According to the establishment of hypercube in the external archive, the candidate solutions in the external archive are divided into a certain number of hypercube according to the objective function value. Take the nondominant solutions set in each hypercube to find its average value. In this process, the nondominant solutions in the same hypercube can effectively exchange information. Figure 5 shows the hypercube created by the candidate solutions in the external archive when the problem is two objectives. For any hypercube, if the space has k nondominant solutions, the average of different dimensions is calculated. As shown in (10), where j represents the j-th hypercube, u represents the number of hypercube with nondominant solutions, k represents that the space has k nondominant solutions, and i represents decision variable dimension. It can be seen from Figure 5 that there is more than one such hypercube, so this paper also uses the shortest distance method to make each particle autonomously choose its own average value. As shown in (11), the particles are compared with the average value in each hypercube until the $\overline{X}_j$ corresponding to the minimum $d\_\min$ is selected. At this time, the velocity update equation of the particle swarm algorithm is shown in (12). In equation (12), $\overline{X}_j$ is the direction guide of the i particle in the social part in t iterations.$\overline{X}_j$ is determined by equation (11). The rest of the parameter analysis is consistent with equation (6).

$$\overline{X}_{j_i} = \frac{x_{j_{i1}} + x_{j_{i2}} + \ldots + x_{j_{ik}}}{k}, \ (i = 1, 2, 3, \ldots, n; \ j = 1, 2, \ldots, u), \tag{10}$$

$$d\_\min = \left(\sum_{j=1}^{u} \overline{X}_j - x \right)^{1/2}, \tag{11}$$

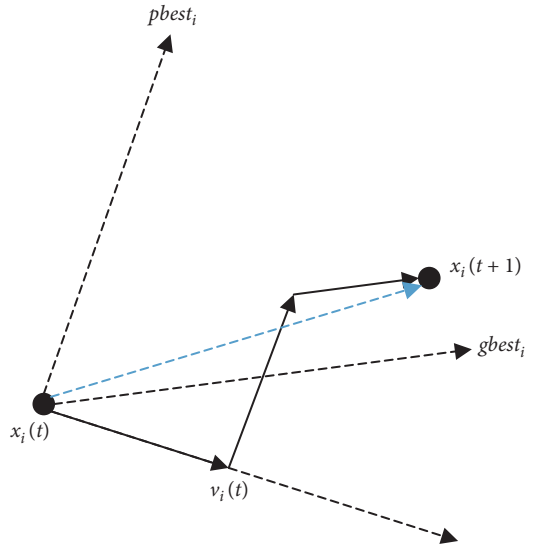

Figure 4: Display of the flying direction of particle *i*.

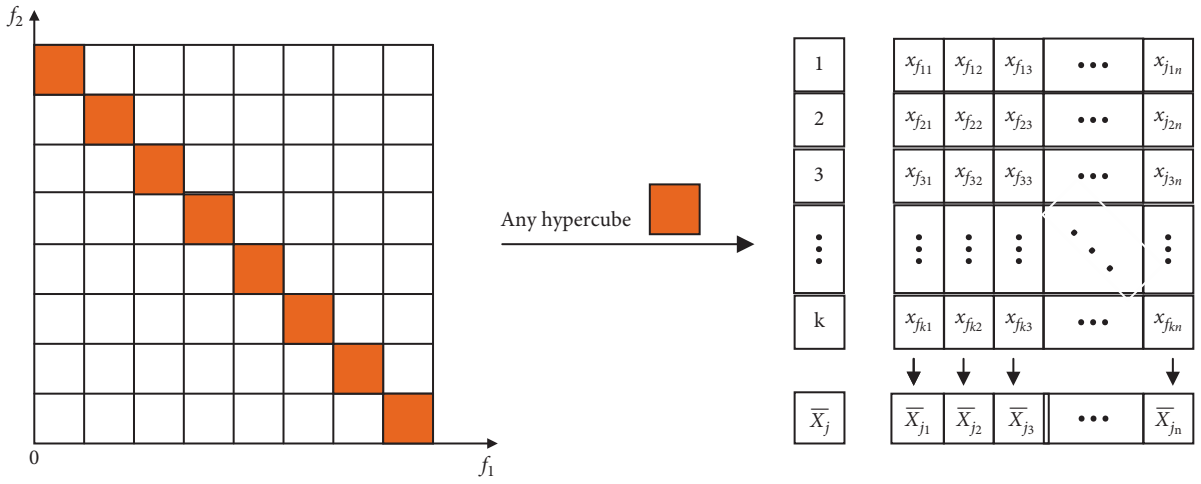

Figure 5: The setting of the social part is based on hypercube.

$$v_i(t+1) = wv_i(t) + c_1 r_1 \left(pbest_i(t) - x_i(t)\right) + c_2 r_2 \left(\overline{X}_j(t) - x_i(t)\right). \tag{12}$$

3.4. Steps of HDMOPSO. The steps of HDMOPSO are as follows:

Step 1 (initialize). First, set related parameters, such as population size N, inertia weight w, and learning factors c_1, c_2, etc. Secondly, randomly initialize the positions of N particles and set the velocity $v = 0$. Finally, the individual extreme value is determined.

Step 2. The dimension of particles is disturbed, similar to literature [12]. However, in this paper, random selection of reference dimensions, random selection of particles and dimensions requiring variation are used to increase the diversity of the algorithm.

Step 3. Calculate the target value for each particle.

Step 4. The Pareto dominant sorting method is used to sort each particle (reference [12]), and the nondominant sorting is selected to be released into the external archive.

Step 5. Update the external archive as described in Section 3.1.

Step 6. The opposite mutation of the method particles is nonlinear decreasing according to Section 3.2.

Step 7. The individual extreme values are updated based on the sorting values of Step 4. If the particle ranks higher, the individual extremum is updated. If there is no promotion, there is no update. If the rank is the same, the choice is made randomly with a probability of 0.5.

Step 8. Set up the social section as described in Section 3.3.

Step 9. The particle velocity and position are updated using equation (12) and (7).

Step 10. If the maximum number of iterations is reached, the algorithm ends; otherwise, return to Step 2.

4. Experimental Study

4.1. Test Problems. In order to better test the performance of HDMOPSO, this paper uses the test problems of ZDT1-4 and ZDT6 [25], which are concave and convex in geometrical problems, and the test problems have two objectives. In order to show the performance of this algorithm as much as possible, this paper also uses DTLZ1-7 [26], the test problems are geometrically linear, convex, and scalable. Finally, this paper also uses the more complex and difficult to converge to the Pareto front UF1-10 [27] test problems to test the performance of the algorithm in this paper.

4.2. Performance Indicators. Inverse generation distance (IGD) [28] and hypervolume (HV) [29], standard deviation (Std.), statistical boxplot, convergence trajectory comparison, and Pareto front simulation are used to compare the performance of each algorithm. The reason for choosing these test indicators is because they are used in most of the literature to compare the performance of various multiobjective optimization algorithms. Both IGD and HV are comprehensive indicators used to evaluate the convergence and diversity of algorithms. But the two are different. IGD is the average of the minimum distance between the set of points on Pareto front surface and the population. HV does not need Pareto front, only a reference point, and the comprehensive performance of the algorithm is evaluated according to the hypervolume obtained from the solution to the reference point. Among them, the smaller the IGD value, the better the convergence and diversity of the algorithm. The larger the value of HV, the better the quality of the solution set. Std. is used to evaluate the stability of the algorithm, and use statistical boxplots to further illustrate the stability of the algorithm. The trajectory compares the convergence velocity of each algorithm.

4.3. Experimental Settings. In this section, we will compare HDMOPSO with seven MOPSOs (MOPSO [15], MOPSOCD [30], dMOPSO [18], SMPSO [17], NMPSO [31], MPSOD [32], CMOPSO [10]) and seven MOEAs (AGEII [33], ANSGAIII [34], DGEA [35], NSGAIII [36], MOEAD [19], ARMOEA [37], and AGEMOEA [38]) to verify its performance (the full name of the acronyms here is shown in Table 1). The settings in the test problems are as follows: ZDT1-4 and ZDT6 are two objective problems, the population size is 200, the number of decision variables of ZDT1-3 is 30 dimensions, and the number of decision variables of ZDT4 and ZDT6 is 10 dimensions; The DTLZ settings are all three objective problems, the population size is 100, and the number of decision variables is 12 dimensions; UF1-7 is set to two objective problems, while UF8-10 is set to three objective problems, the population size is all 200, and the number of decision variables is all 30 dimensions. Each algorithm is evaluated 10,000 times and each algorithm is independently run 30 times. In order to ensure the fairness of algorithm comparison, other parameters of the comparison algorithm are set with reference to the original text. The parameters data is shown in Table 2. In Table 2, p_c and p_m represent crossover probability and mutation probability respectively. η_c and η_m are the distribution indices of simulating binary crossover (SBX) and polynomial-based mutation (PM) respectively. *div* is the number of divisions of the coordinate axes. w, c_1, c_2, c_3 is a parameter in the speed update in PSO. F and CR are differential evolution parameters set in MPSOD. r is the number of selected elite particles in CMOPSO. R is the number of direction vectors in DGEA. All experimental results are obtained in MATLAB R2020b version in a computer with a PC of 3.60 GHz and 16 GB of storage. The original codes of all comparison algorithms are provided by PlatEMO [39].

4.4. Comparison with MOPSOs. Table 3 shows the average and standard deviation of the IGD obtained from 30 independent runs of the HDMOPSO and MOPSOs on 22 test problems. The following is a data analysis of Table 3 (The best average for each test instance in Table 3 is highlighted in bold. The meaning of bold highlights in Tables 4–6 is the same as in Table 3). It can be seen from Table 3 that the test problems of the HDMOPSO in ZDT have a good IGD value. Especially the effect on ZDT4 is particularly obvious, which shows that the algorithm performs better on convex sets. However, it can be seen from the data of ZDT3 that the HDMOPSO has a significant deterioration compared to other seven algorithms, indicating that the algorithm has a poor effect on discontinuous test problems. UF test problems are obviously better, and 80% of the IGD values of the other seven algorithms show a clear advantage. Especially UF7 and UF10 are an order of magnitude better than the other seven algorithms. However, the experimental result cannot converge to the Pareto front obviously on the UF test problems, and it can be further improved in future research. It can be seen from the test problems of DTLZ that the algorithm is seriously degraded. SMPSO and MPSOD performed well in the test functions of DTLZ, but only two problems well. It is worth noting that the HDMOPSO has achieved better results on DTLZ6. From the overall effect analysis, it can be found that among the 22 test problems given above, the performance of the HDMOPSO is the best.

Table 4 shows the average and standard deviation of HV obtained from 30 independent runs of the HDMOPSO and MOPSOs on 22 test problems. It can be seen from Table 4 that ZDT test problems are obviously better, because 80% of ZDT test problems are better than the other seven algorithms. The test problems of UF have some changes, but there are also 80% test problems that are better. There is still a significant deterioration in the test problems of DTZL. It can be seen from Table 4 that the HDMOPSO and MOPSOs still have strong competitiveness.

Table 1: List of acronyms.

Acronyms	The full name of an acronym
MOPSO	MOPSO: a Proposal for multiple objective particle swarm optimization
MOPSOCD	An effective use of crowding distance in multiobjective particle swarm optimization
dMOPSO	A multiobjective particle swarm optimizer based on decomposition
SMPSO	SMPSO a New PSO-based metaheuristic for multiobjective optimization
NMPSO	particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems
MPSOD	A new multiobjective particle swarm optimization algorithm based on decomposition
CMOPSO	A competitive mechanism based multiobjective particle swarm optimizer with fast convergence
AGEII	A fast approximation-guided evolutionary multiobjective algorithm
NSGAIII	An evolutionary many-objective optimization algorithm using reference – point-based non-dominated sorting approach, part I: Solving problems with box constraints
MOEAD	MOEA/D:a multiobjective evolutionary algorithm based on decomposition
ARMOEA	An indicator based multiobjective evolutionary algorithm with reference point adaptation for better versatility
AGEMOEA	An adaptive evolutionary algorithm based on non-Euclidean geometry for many-objective optimization

Table 2: Parameter settings of all the compared algorithms.

	Algorithms	Parameters settings
1	MOPSO	$w \in [0.1, 0.5]$, c_1, $c_2 \in [1.5, 2.5]$, div = 10
2	MOPSOCD	$w \in [0.1, 0.5]$, c_1, $c_2 \in [1.5, 2.5]$
3	dMOPSO	$w \in [0.1, 0.5]$, c_1, $c_2 \in [1.5, 2.5]$
4	SMPSO	$w \in [0.1, 0.5]$, c_1, $c_2 \in [1.5, 2.5]$, $p_m = 1/n$
5	NMPSO	$w \in [0.1, 0.5]$,c_1, c_2, $c_3 \in [1.5, 2.5]$, $p_m = 1/n$, $\eta_m = 20$
6	MPSOD	$w \in [0.1, 0.5]$, c_1, c_2, $c_3 \in [1.5, 2.5]$, $p_c = 0.9$, $F = 0.5$, $CR = 0.5$, $p_m = 1/n$, $\eta_m = 20$, $\eta_c = 20$
7	CMOPSO	$r = 10$
8	AGEII	$p_m = 1/n$, $p_c = 0.9$, $\eta_m = 20$, $\eta_c = 20$
9	ANSGAIII	$p_m = 1/n$, $p_c = 1.0$, $\eta_m = 20$, $\eta_c = 20$
10	DGEA	$R = 10$
11	NSGAIII	$p_m = 1/n$, $p_c = 1.0$, $\eta_m = 20$, $\eta_c = 20$
12	MOEAD	$p_m = 1/n$, $p_c = 1.0$, $\eta_m = 20$, $\eta_c = 20$
13	ARMOEA	$p_m = 1/n$, $p_c = 1.0$, $\eta_m = 20$, $\eta_c = 20$
14	AGEMOEA	$p_m = 1/n$, $p_c = 1.0$, $\eta_m = 20$, $\eta_c = 20$
15	HDMOPSO	$w = 0.4$, $c_1 = c_2 = 2$, div = 50

In order to more intuitively see the stability of the algorithm in this paper when compare with MOPSOs, we use a statistical boxplot of the IGD values obtained from 30 independent runs of each algorithm. Figure 6 shows the 12 test problems in Table 3 that perform well. It can be seen from Figure 6 that it is consistent with the results obtained in Table 3. Not only is the optimization result better, but it also shows a better advantage in terms of stability. Although not significant compared with other algorithms in ZDT6, they all performed better than MOPSO, MOPSOCD and MPSOD. The comparison between HDMOPSO, NMPSO and dMOPSO on DTLZ6 is not obvious, but it has obvious advantages over the other five algorithms. For the other 10 test problems, they all showed good advantages, especially UF5, UF7, UF9, and UF10 showed obvious advantages.

Another important performance of HDMOPSO is the speed of convergence. Figure 7 shows the IGD convergence trajectory obtained by HDMOPSO and MOPSOs running 10000 times in ZDT6, UF4, and DTLZ6. Although HDMOPSO is not obvious compared with NMPSO on ZDT6 and DTLZ6, the convergence speed of HDMOPSO on UF4 is significantly better than that of optimal NMPSO. A closer look at the convergence trajectories of ZDT6 and DTLZ6 will reveal that HDMOPSO converges faster than NMPSO in the early stage of the evaluation. In the data in Table 3, the IGD value of NMPSO on the two test problems of ZDT6 and DTLZ6 is better than that of NMPSO. The above shows that HDMOPSO still has some advantages over NMPSO in terms of convergence speed.

Finally, Figure 8 shows the Pareto front simulation of each algorithm on ZDT4. It can be seen from the figure that HDMOPSO almost all converge to the Pareto front, and the distribution is good. Other algorithms have no convergent Pareto front.

Through the above analysis of the comparison of HDMOPSO and MOPSOs, it can be concluded. Among the 22 test functions of HDMOPSO, 12 test problems performed better on the IGD indicator, and it can be seen from Figure 6 that the stability is also better. On the HV indicator, 13 test problems performed well. Figures 7 and 8 also briefly temporarily show the convergence of HDMOPSO. In particular, it can be concluded from the data in Tables 3 and 4 that HDMOPSO is more competitive than the other seven algorithms in the above ZDT and UF test problems. It also shows that the operator proposed in this paper has good performance in convergence and diversity, which confirms the relevant statement in Section 3 of this paper.

Table 3: IGD values of HDMOPSO and MOPSOs on 22 test problems.

Problems	MOPSO	MOPSOCD	dMOPSO	SMPSO	NMPSO	MPSOD	CMOPSO	HDMOPSO
ZDT1	7.5088e − 1	3.9812e − 3	4.8573e − 2	9.5769e − 2	3.8449e − 2	1.0348e − 1	3.1875e − 3	**3.1535e − 3**
	(2.92e − 1)	(5.25e − 3)	(1.60e − 2)	(9.47e − 2)	(2.19e − 2)	(4.21e − 2)	(4.35e − 4)	**(1.27e − 4)**
ZDT2	1.3707e + 0	1.1354e − 1	3.8831e − 2	9.2544e − 2	7.5055e − 2	1.4178e − 1	**2.9214e − 3**	3.3724e − 3
	(3.04e − 1)	(2.14e − 1)	(1.16e − 2)	(1.65e − 1)	(1.16e − 1)	(7.53e − 2)	**(4.96e − 4)**	(1.66e − 4)
ZDT3	7.6697e − 1	6.0399e − 2	3.6479e − 2	2.1381e − 1	9.1792e − 2	2.2252e − 1	**3.8754e − 3**	1.9279e − 1
	(2.06e − 1)	(5.76e − 2)	(7.25e − 3)	(9.78e − 2)	(1.05e − 2)	(6.02e − 2)	**(8.05e − 4)**	(8.24e − 4)
ZDT4	1.4121e + 1	1.8729e + 1	3.0691e + 0	9.3197e + 0	1.6335e + 1	3.6865e + 1	2.0777e + 1	**3.2676e − 3**
	(4.52e + 0)	(5.99e + 0)	(4.33e + 0)	(4.44e + 0)	(6.71e + 0)	(4.99e + 0)	(5.99e + 0)	**(1.66e − 4)**
ZDT6	1.2262e − 1	3.8317e − 3	5.5255e − 3	1.9286e − 3	2.2633e − 3	1.7753e − 2	1.5817e − 3	**1.4406e − 3**
	(5.57e − 1)	(1.75e − 3)	(5.36e − 3)	(9.20e − 5)	(1.88e − 4)	(9.12e − 3)	(3.73e − 5)	**(1.40e − 4)**
UF1	5.4509e − 1	7.0292e − 1	6.4403e − 1	3.8697e − 1	1.2881e − 1	2.5801e − 1	**8.9465e − 2**	1.0285e − 1
	(9.68e − 2)	(1.37e − 1)	(9.29e − 2)	(9.73e − 2)	(2.06e − 2)	(3.95e − 2)	**(1.16e − 2)**	(3.90e − 3
UF2	1.0642e − 1	1.3831e − 1	9.5994e − 2	1.0196e − 1	8.3881e − 2	1.1263e − 1	6.2882e − 2	**4.2773e − 2**
	(1.52e − 2)	(1.41e − 2)	(7.11e − 3)	(9.90e − 3)	(7.73e − 3)	(7.82e − 3)	(7.46e − 3)	**(4.68e − 3)**
UF3	5.1345e − 1	3.7654e − 1	3.3278e − 1	4.5443e − 1	3.4509e − 1	5.0080e − 1	3.7938e − 1	**2.2609e − 1**
	(2.66e − 2)	(5.93e − 2)	(8.05e − 3)	(5.30e − 2)	(4.01e − 2)	(1.39e − 2)	(4.19e − 2)	**(3.25e − 2**
UF4	1.1332e − 1	7.6570e − 2	1.3842e − 1	1.1399e − 1	6.0533e − 2	9.9699e − 2	1.1406e − 1	**5.6199e − 2**
	(1.32e − 2)	(8.53e − 3)	(3.90e − 3)	(5.74e − 3)	(7.40e − 3)	(3.70e − 3)	(9.14e − 3)	**(3.62e − 3)**
UF5	3.1549e + 0	3.9278e + 0	3.2280e + 0	2.9766e + 0	1.6894e + 0	2.8148e + 0	8.6853e − 1	**4.7202e − 1**
	(4.02e − 1)	(4.15e − 1)	(2.91e − 1)	(5.34e − 1)	(4.72e − 1)	(2.29e − 1)	(2.37e − 1)	**(6.69e − 2)**
UF6	2.3228e + 0	2.6922e + 0	2.5325e + 0	1.2647e + 0	6.2828e − 1	1.3585e + 0	**3.9239e − 1**	5.7701e − 1
	(5.49e − 1)	(7.84e − 1)	(6.61e − 1)	(4.51e − 1)	(7.21e − 2)	(2.17e − 1)	**(5.48e − 2)**	(1.12e − 1)
UF7	6.1070e − 1	6.4461e − 1	3.9902e − 1	3.8701e − 1	1.8045e − 1	2.3004e − 1	1.5988e − 1	**5.4736e − 2**
	(1.01e − 1)	(1.08e − 1)	(8.50e − 2)	(1.41e − 1)	(1.40e − 1)	(5.59e − 2)	(1.35e − 1)	**(4.64e − 3)**
UF8	4.0988e − 1	7.9818e − 1	3.4771e − 1	3.9381e − 1	4.8985e − 1	5.5254e − 1	6.3473e − 1	**2.9892e − 1**
	(4.08e − 2)	(1.73e − 1)	(2.78e − 2)	(5.03e − 2)	(1.02e − 1)	(4.72e − 2)	(9.14e − 2)	**(8.93e − 2)**
UF9	5.5595e − 1	8.6183e − 1	5.8222e − 1	5.5428e − 1	4.5999e − 1	6.5550e − 1	9.1997e − 1	**1.3840e − 1**
	(3.53e − 2)	(1.14e − 1)	(3.31e − 2)	(3.84e − 2)	(7.11e − 2)	(4.06e − 2)	(1.16e − 1)	**(2.20e − 2)**
UF10	2.2568e + 0	4.9302e + 0	9.4535e − 1	2.8317e + 0	1.5045e + 0	4.1995e + 0	4.3317e + 0	**5.1341e − 1**
	(2.69e − 1)	(7.78e − 1)	(1.42e − 3)	(4.19e − 1)	(2.82e − 1)	(3.64e − 1)	(4.34e − 1)	**(9.72e − 2)**
DTLZ1	6.5415e + 1	4.3410e + 1	**1.1274e + 1**	1.3471e + 1	3.1643e + 1	4.0410e + 1	6.0047e + 1	4.4077e + 1
	(2.14e + 1)	(1.15e + 1)	**(9.86e + 0)**	(1.55e + 1)	(6.32e + 0)	(5.50e + 0)	(1.90e + 1)	(1.40e + 1)
DTLZ2	1.0105e − 1	1.0093e − 1	1.4135e − 1	8.9297e − 2	8.0083e − 2	**5.6798e − 2**	6.1246e − 2	1.0454e − 1
	(1.17e − 2)	(7.88e − 3)	(1.21e − 2)	(7.51e − 3)	(2.41e − 3)	**(8.22e − 4)**	(1.20e − 3)	(7.54e − 3)
DTLZ3	1.5740e + 2	1.1338e + 2	5.4112e + 1	**2.9074e + 1**	9.0109e + 1	1.1838e + 2	1.3237e + 2	1.2671e + 2
	(5.27e + 1)	(3.54e + 1)	(5.58e + 1)	**(3.08e + 1)**	(1.87e + 1)	(1.59e + 1)	(3.85e + 1)	(3.40e + 1)
DTLZ4	4.0345e − 1	2.9034e − 1	3.1557e − 1	4.4687e − 1	1.2632e − 1	**6.2524e − 2**	1.5281e − 1	1.1168e − 1
	(1.71e − 1)	(4.99e − 2)	(3.37e − 2)	(1.86e − 1)	(1.41e − 1)	**(5.04e − 3)**	(2.69e − 1)	(3.22e − 2)
DTLZ5	1.5467e − 2	2.1884e − 2	4.1222e − 2	**5.7212e − 3**	1.2661e − 2	5.4102e − 2	7.7990e − 3	2.2555e − 2
	(6.41e − 3)	(7.37e − 3)	(5.96e − 3)	**(3.92e − 4)**	(2.12e − 3)	(5.42e − 3)	(6.03e − 4)	(2.85e − 3)
DTLZ6	3.1831e + 0	1.0237e − 2	3.2892e − 2	1.1052e + 0	1.5140e − 2	2.5577e − 1	2.3427e − 1	**7.5019e − 3**
	(7.91e − 1)	(1.62e − 2)	(3.09e − 4)	(9.46e − 1)	(2.31e − 3)	(2.58e − 1)	(4.95e − 1)	**(1.07e − 3)**
DTLZ7	1.6046e + 0	8.9805e − 2	1.3944e − 1	2.0575e − 1	**7.5483e − 2**	1.6499e − 1	1.7726e − 1	9.8273e − 2
	(7.71e − 1)	(7.76e − 3)	(5.84e − 3)	(1.52e − 1)	**(3.57e − 3)**	(1.57e − 2)	(2.33e − 1)	(1.16e − 2)
Best/All	0/22	0/22	1/22	2/22	1/22	2/22	4/22	12/22

The best average on each test instance is highlighted in bold.

4.5. Comparison with MOEAs. Table 5 shows the average IGD of HDMOPSO and MOEAs for 30 independent runs on 22 test problems. It can be seen from Table 5 that HDMOPSO is significantly better than the other four algorithms in the ZDT test problems. Of course, the performance of the ZDT3 test problem here is not very obvious, which also shows that HDMOPSO is not outstanding in the non-continuous set. For other concave and convex sets, HDMOPSO has obvious advantages over the rest of MOEAs, because ZDT1, ZDT2, and ZDT6 are all an order of magnitude lower than the best value of the other seven algorithms, and it is in the ZDT4 test problem. It is two orders of magnitude lower than the best one of the other MOEAs algorithms. In addition, the UF test problems also show a clear advantage, because 90% of the 10 UF test problems are better than the other four algorithms. Although the single UF optimization effect is not very obvious, the overall effect is good. In the DTLZ test problem, you can see the algorithm obvious deterioration, and none of the algorithms given can show obvious advantages in the DTLZ test problems, and the better one is AGEMOEA. Because 3 of the 7 DTLZ test problems are better than other test problems. Only one or two of the remaining algorithms performed well on the DTLZ test problems. Such a result may be that each test problem of DTLZ is quite different and contains many local optimal values, which makes it more

Table 4: HV values of HDMOPSO and MOPSOs on 22 test problems.

Problems	MOPSO	MOPSOCD	dMOPSO	SMPSO	NMPSO	MPSOD	CMOPSO	HDMOPSO
ZDT1	8.7931e − 2	7.1927e − 1	6.6243e − 1	6.0408e − 1	6.7972e − 1	5.7261e − 1	7.1985e − 1	**7.2106e − 1**
	(9.03e − 2)	(7.63e − 3)	(1.85e − 2)	(1.14e − 1)	(2.38e − 2)	(5.43e − 2)	(6.73e − 4)	**(1.67e − 4)**
ZDT2	0.0000e + 0	3.6202e − 1	3.8608e − 1	3.6414e − 1	3.8523e − 1	2.7324e − 1	4.4470e − 1	**4.4479e − 1**
	(0.00e + 0)	(1.38e − 1)	(1.89e − 2)	(1.23e − 1)	(9.41e − 2)	(7.70e − 2)	(8.96e − 4)	**(3.14e − 4)**
ZDT3	1.1302e − 1	5.6641e − 1	6.0143e − 1	5.1021e − 1	5.7190e − 1	4.5070e − 1	5.9962e − 1	**6.5930e − 1**
	(9.47e − 2)	(4.00e − 2)	(1.46e − 2)	(7.81e − 2)	(5.59e − 3)	(4.78e − 2)	(1.34e − 3)	**(5.18e − 4)**
ZDT4	0.0000e + 0	0.0000e + 0	6.2941e − 2	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	**7.2109e − 1**
	(0.00e + 0)	(0.00e + 0)	(7.98e − 2)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	**(1.94e − 4)**
ZDT6	3.6205e − 1	3.8818e − 1	3.8626e − 1	3.9003e − 1	3.8979e − 1	3.7439e − 1	**3.9034e − 1**	3.8917e − 1
	(8.57e − 2)	(1.85e − 3)	(5.94e − 3)	(9.26e − 5)	(1.63e − 4)	(9.32e − 3)	**(4.35e − 5)**	(1.28e − 4)
UF1	1.3865e − 1	4.8415e − 2	6.5077e − 2	2.6626e − 1	5.2452e − 1	3.6635e − 1	5.8159e − 1	**5.8358e − 1**
	(6.26e − 2)	(4.52e − 2)	(5.42e − 2)	(8.05e − 2)	(3.33e − 2)	(4.49e − 2)	(1.25e − 2)	**(5.07e − 3)**
UF2	6.0344e − 1	5.4564e − 1	6.1588e − 1	6.0714e − 1	6.1881e − 1	5.7945e − 1	6.4476e − 1	**6.7344e − 1**
	(9.56e − 3)	(1.92e − 2)	(6.00e − 3)	(8.34e − 3)	(8.74e − 3)	(9.61e − 3)	(7.24e − 3)	**(5.90e − 3)**
UF3	1.5813e − 1	2.5471e − 1	3.0604e − 1	2.0705e − 1	2.9285e − 1	1.7439e − 1	2.7643e − 1	**4.4309e − 1**
	(1.85e − 2)	(4.65e − 2)	(1.01e − 2)	(4.80e − 2)	(3.54e − 2)	(1.29e − 2)	(3.07e − 2)	**(4.02e − 2)**
UF4	2.9015e − 1	3.3601e − 1	2.5150e − 1	2.9022e − 1	3.6468e − 1	3.0708e − 1	2.8639e − 1	**3.6976e − 1**
	(1.65e − 2)	(1.17e − 2)	(4.74e − 3)	(6.68e − 3)	(9.76e − 3)	(4.93e − 3)	(1.07e − 2)	**(4.36e − 3)**
UF5	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	1.4949e − 2	**1.9772e − 2**
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(2.77e − 2)	**(2.21e − 2)**
UF6	0.0000e + 0	0.0000e + 0	0.0000e + 0	1.8676e − 3	3.3322e − 2	0.0000e + 0	**1.4986e − 1**	1.3127e − 2
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(7.67e − 3)	(3.38e − 2)	(0.00e + 0)	**(7.13e − 2)**	(1.39e − 2)
UF7	5.1589e − 2	2.0540e − 2	1.6202e − 1	1.7451e − 1	3.8134e − 1	2.7511e − 1	4.1920e − 1	**5.0640e − 1**
	(4.38e − 2)	(2.93e − 2)	(6.16e − 2)	(1.02e − 1)	(1.10e − 1)	(6.12e − 2)	(9.58e − 2)	**(6.73e − 3)**
UF8	1.7316e − 1	8.5853e − 3	2.7047e − 1	1.6701e − 1	2.8249e − 1	5.2054e − 2	9.4227e − 3	**3.5290e − 1**
	(3.13e − 2)	(1.78e − 2)	(1.87e − 2)	(3.92e − 2)	(6.24e − 2)	(1.89e − 2)	(1.56e − 2)	**(1.67e − 2)**
UF9	2.1069e − 1	2.6834e − 2	2.2614e − 1	2.1584e − 1	3.1904e − 1	1.1375e − 1	1.2671e − 2	**6.3437e − 1**
	(3.67e − 2)	(3.09e − 2)	(1.75e − 2)	(3.84e − 2)	(5.62e − 2)	(2.83e − 2)	(1.52e − 2)	**(2.37e − 2)**
UF10	0.0000e + 0	0.0000e + 0	9.0885e − 2	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	**3.1914e − 1**
	(0.00e + 0)	(0.00e + 0)	(4.33e − 5)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	**(4.40e − 2)**
DTLZ1	3.0031e − 5	0.0000e + 0	6.0507e − 3	**7.7696e − 2**	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0
	(1.64e − 4)	(0.00e + 0)	(3.06e − 2)	**(1.78e − 1)**	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)
DTLZ2	4.4743e − 1	4.5905e − 1	3.6806e − 1	4.5953e − 1	**5.5708e − 1**	5.4695e − 1	5.3735e − 1	4.7654e − 1
	(2.05e − 2)	(1.68e − 2)	(2.25e − 2)	(1.65e − 2)	**(1.26e − 3)**	(2.35e − 3)	(3.01e − 3)	(1.15e − 2)
DTLZ3	0.0000e + 0	0.0000e + 0	0.0000e + 0	**4.3486e − 2**	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	**(7.73e − 2)**	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)
DTLZ4	2.5413e − 1	3.2113e − 1	3.0697e − 1	2.9025e − 1	5.3648e − 1	**5.3767e − 1**	4.8396e − 1	5.1071e − 1
	(9.14e − 2)	(6.45e − 2)	(4.79e − 2)	(1.12e − 1)	(6.52e − 2)	**(9.08e − 3)**	(1.33e − 1)	(1.03e − 2)
DTLZ5	1.8647e − 1	1.8447e − 1	1.5607e − 1	**1.9841e − 1**	1.9596e − 1	1.4596e − 1	1.9642e − 1	1.8259e − 1
	(5.26e − 3)	(6.82e − 3)	(1.02e − 2)	**(3.22e − 4)**	(7.26e − 4)	(6.89e − 3)	(5.33e − 4)	(3.45e − 3)
DTLZ6	0.0000e + 0	**1.9703e − 1**	1.8259e − 1	6.4336e − 2	1.9555e − 1	1.0561e − 1	1.5968e − 1	1.9487e − 1
	(0.00e + 0)	**(9.32e − 3)**	(2.20e − 4)	(8.99e − 2)	(1.02e − 3)	(5.88e − 2)	(8.12e − 2)	(4.19e − 3)
DTLZ7	5.2658e − 2	2.6585e − 1	2.4595e − 1	2.3873e − 1	**2.7377e − 1**	2.1759e − 1	2.6121e − 1	2.6668e − 1
	(5.50e − 2)	(2.37e − 3)	(3.67e − 3)	(2.32e − 2)	**(1.57e − 3)**	(1.05e − 2)	(2.34e − 2)	(3.33e − 3)
Best/All	0/22	1/22	0/22	3/22	2/22	1/22	2/22	13/22

difficult to optimize the algorithm. HDMOPSO and MOEAs in a total of 22 test problems, HDMOPSO performs better than the other seven algorithms with 15 and showed obvious advantages in the test problems of ZDT and UF. HDMOPSO is more competitive compared with the other seven algorithms.

In order to further prove the results obtained above, the HV average and standard deviation of each algorithm independently run 30 times are also used. The results are shown in Table 6. It can be seen from the HV indicator that HDMOPSO has a clear advantage over the other algorithms in the ZDT test problems, reaching 100% excellence. UF also show a better advantage than other algorithms, because 80% of the UF test problems performed better than the other seven MOEAs algorithms. Although the data optimization is not very obvious, the overall data can still see obvious advantages. HDMOPSO deteriorated significantly in the DTLZ test problems. The best performer in DTLZ is AGEMOEA, because 3 out of 7 DTLZ problems are better. Compared to the results in Tables 5 and 6, although there are some differences in the conclusions drawn from the data, it can also be clearly concluded that HDMOPSO is more competitive than other MOEAs in ZDT and UF test problems.

In order to show more clearly that HDMOPSO not only performs excellent numerically, but also performs well in terms of stability. Figure 9 shows the statistical boxplot of the

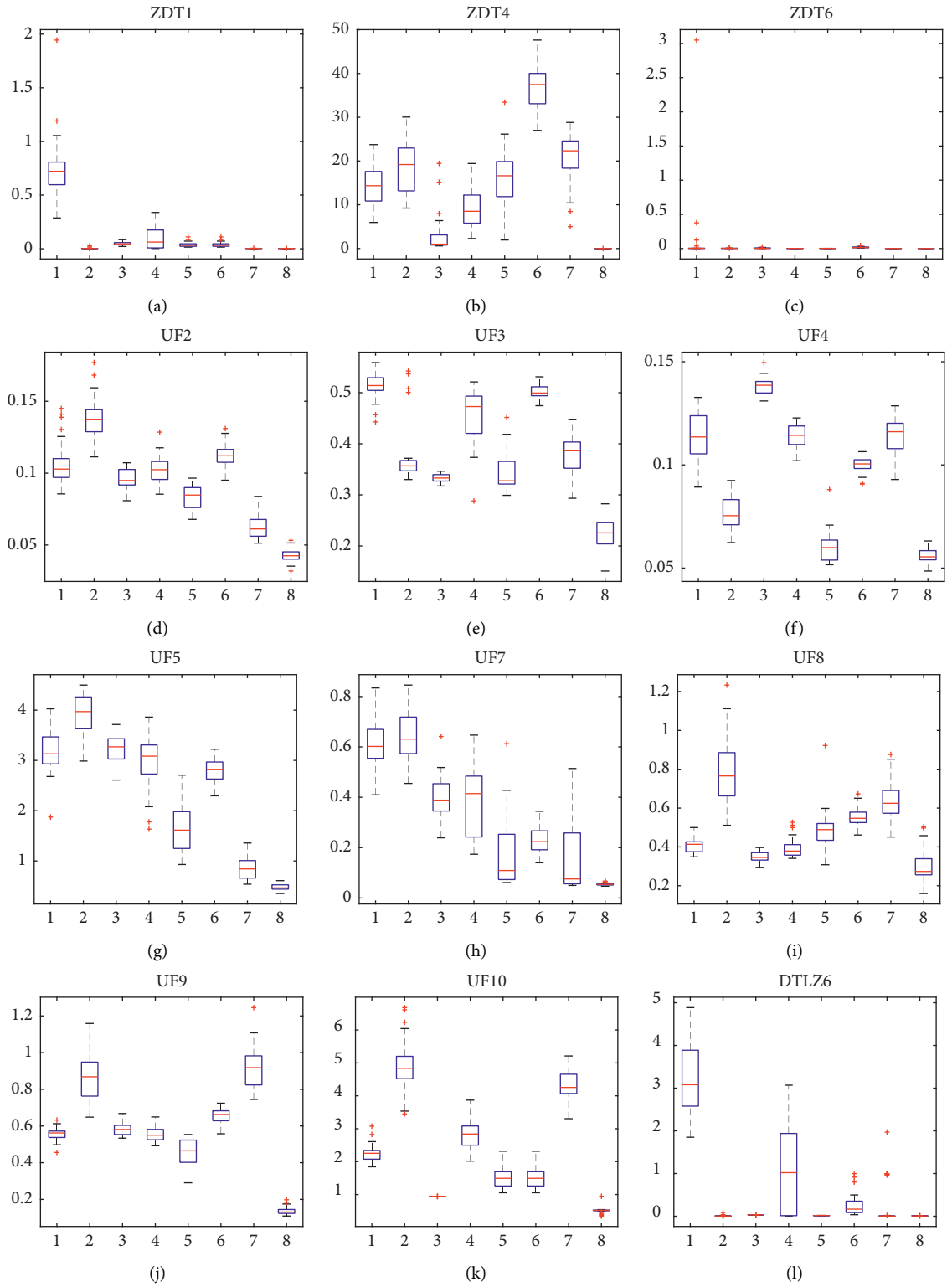

Figure 6: IGD values statistical boxplot of HDMOPSO and MOPSOs on ZDT1, ZDT4, ZDT6, UF2-5, UF7-10, and DTLZ6 problems. (1, 2, 3, 4, 5, 6, 7, and 8 in each statistical boxplot represent MOPSO, MOPSOCD, dMOPSO, SMPSO, NMPSO, MPSOD, CMOPSO, and HDMOPSO, respectively).

15 test problems in Table 5 that perform well. It can be seen from Figure 9 that HDMOPSO performs better on ZDT1-2, ZDT4 and ZDT6, and its stability is better than other MOEAs. HDMOPSO has shown obvious advantages in UF test problems except for UF6, and the stability is better than other algorithms. In the DTLZ test problem, although it is

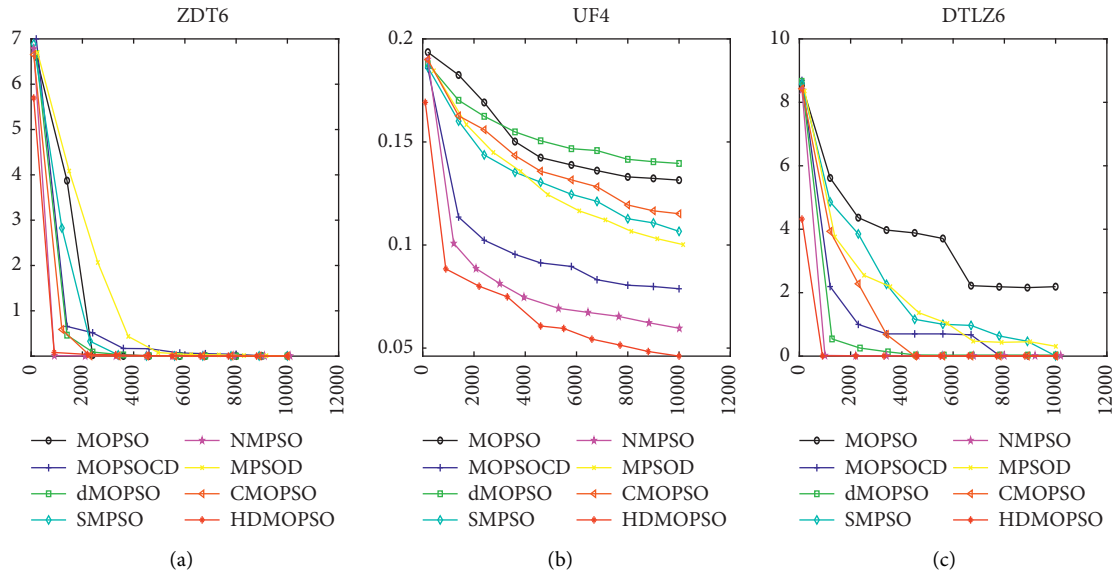

Figure 7: IGD values convergence trajectory of HDMOPSO and MOPSOs comparison on ZDT6, UF4, and DTLZ6 problems.

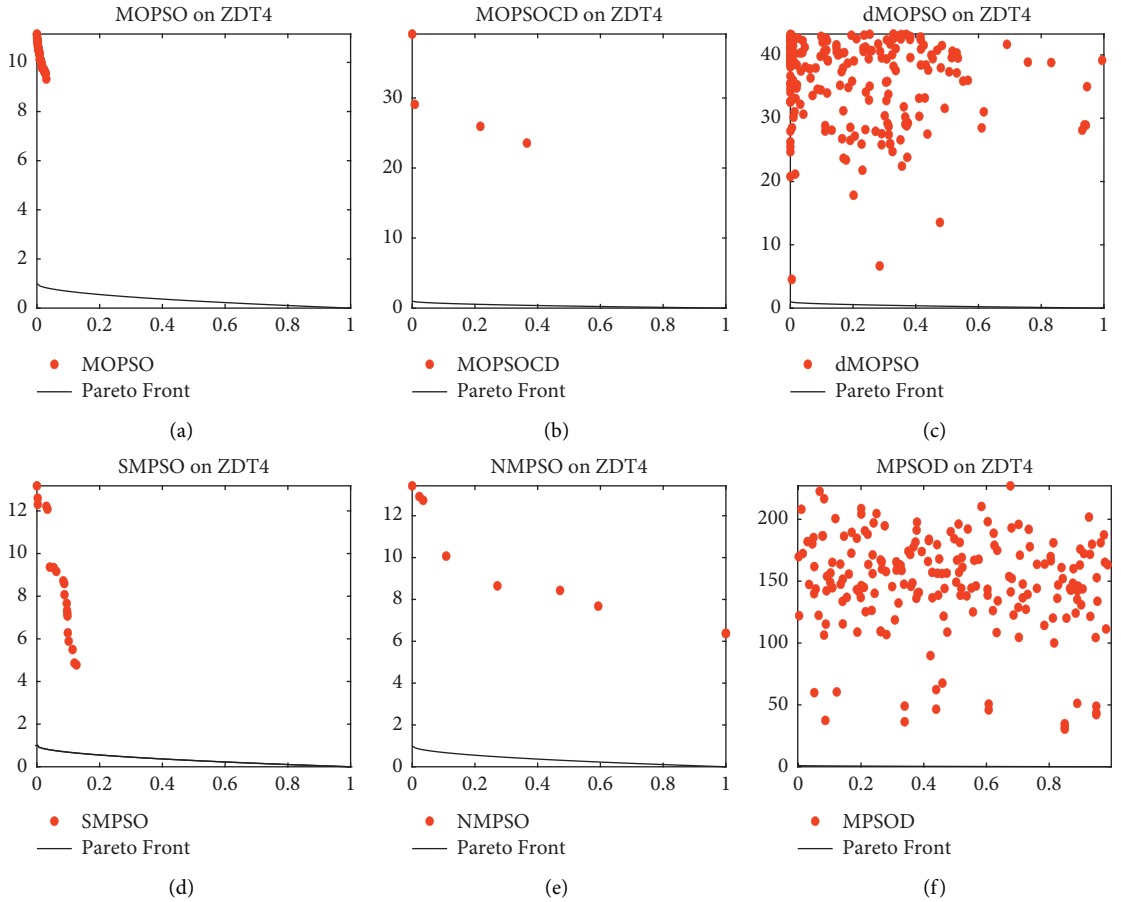

Figure 8: Continued.

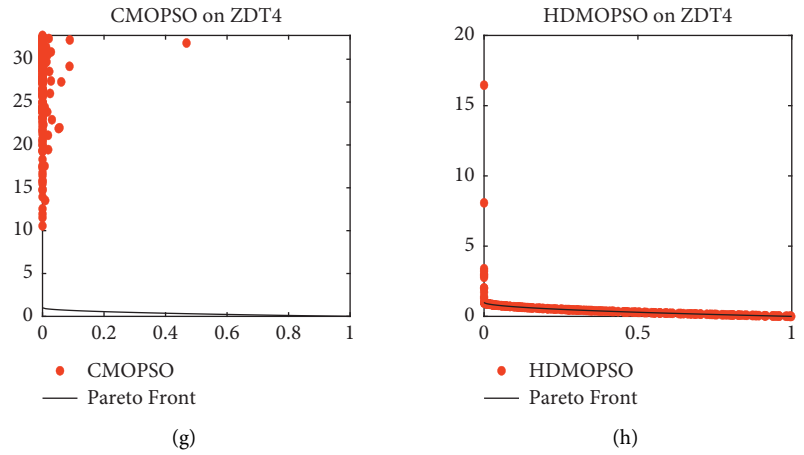

Figure 8: The Pareto front simulation of HDMOPSO and MOPSOs on ZDT4 problem.

Table 5: IGD values of HDMOPSO and MOEAs on 22 test Problems.

Problems	AGEII	ANSGAIII	DGEA	NSGAIII	MOEAD	ARMOEA	AGEMOEA	HDMOPSO
ZDT1	8.5782$e-2$	1.0467$e-1$	1.2053$e+0$	1.0429$e-1$	1.9889$e-1$	7.6736$e-2$	3.6880$e-2$	**3.1535$e-3$**
	(2.08$e-2$)	(1.70$e-2$)	(2.29$e-1$)	(1.49$e-2$)	(9.18$e-2$)	(3.69$e-2$)	(6.45$e-3$)	**(1.27$e-4$)**
ZDT2	1.8054$e-1$	2.0715$e-1$	9.0498$e-1$	1.9997$e-1$	5.8069$e-1$	6.7290$e-1$	7.7285$e-2$	**3.3724$e-3$**
	(8.01$e-2$)	(5.94$e-2$)	(3.82$e-1$)	(3.51$e-2$)	(6.00$e-2$)	(5.31$e-2$)	(1.92$e-2$)	**(1.66$e-4$)**
ZDT3	8.9131$e-2$	9.4678$e-2$	1.0018$e+0$	9.3148$e-2$	1.8357$e-1$	5.5582$e-2$	**3.2164$e-2$**	1.9279$e-1$
	(2.48$e-2$)	(1.55$e-2$)	(2.14$e-1$)	(1.44$e-2$)	(6.61$e-2$)	(3.07$e-2$)	**(1.03$e-2$)**	(8.24$e-4$)
ZDT4	5.5026$e-1$	2.5973$e+0$	8.8328$e+0$	2.7783$e+0$	5.5262$e-1$	1.6582$e+0$	9.0188$e-1$	**3.2676$e-3$**
	(3.48$e-1$)	(7.71$e-1$)	(6.15$e+0$)	(7.38$e-1$)	(1.82$e-1$)	(7.07$e-1$)	(3.21$e-1$)	**(1.66$e-4$)**
ZDT6	2.4504$e-1$	1.5490$e+0$	1.2914$e-1$	1.4790$e+0$	8.3351$e-2$	9.1101$e-1$	3.9277$e-1$	**1.4406$e-3$**
	(1.14$e-1$)	(1.97$e-1$)	(6.77$e-1$)	(2.24$e-1$)	(3.09$e-2$)	(2.46$e-1$)	(1.22$e-1$)	**(1.40$e-4$)**
UF1	1.5388$e-1$	1.3636$e-1$	6.2298$e-1$	1.5270$e-1$	3.0044$e-1$	1.1692$e-1$	1.1202$e-1$	**1.0285$e-1$**
	(5.14$e-2$)	(2.85$e-2$)	(1.34$e-1$)	(4.12$e-2$)	(8.04$e-2$)	(2.29$e-2$)	(2.27$e-2$)	**(3.90$e-3$)**
UF2	9.2897$e-2$	8.1301$e-2$	1.7032$e-1$	8.2564$e-2$	2.3152$e-1$	7.8838$e-2$	7.0535$e-2$	**4.2773$e-2$**
	(1.19$e-2$)	(7.06$e-3$)	(2.16$e-2$)	(5.03$e-3$)	(5.81$e-2$)	(1.20$e-2$)	(7.08$e-3$)	**(4.68$e-3$)**
UF3	4.9379$e-1$	4.7702$e-1$	5.7403$e-1$	4.8138$e-1$	3.3739$e-1$	4.3339$e-1$	4.2878$e-1$	**2.2609$e-1$**
	(3.06$e-2$)	(8.89$e-3$)	(4.99$e-2$)	(7.78$e-3$)	(2.31$e-2$)	(2.91$e-2$)	(2.67$e-2$)	**(3.25$e-2$)**
UF4	1.0075$e-1$	9.4497$e-2$	1.2237$e-1$	9.5983$e-2$	1.1437$e-1$	8.1805$e-2$	8.1923$e-2$	**5.6199$e-2$**
	(4.48$e-3$)	(3.05$e-3$)	(8.65$e-3$)	(3.42$e-3$)	(5.47$e-3$)	(2.63$e-3$)	(3.13$e-3$)	**(3.62$e-3$)**
UF5	9.9788$e-1$	1.4897$e+0$	2.9718$e+0$	1.4989$e+0$	1.4515$e+0$	6.7696$e-1$	7.2508$e-1$	**4.7202$e-1$**
	(2.90$e-1$)	(3.01$e-1$)	(6.70$e-1$)	(3.52$e-1$)	(2.29$e-1$)	(1.71$e-1$)	(1.92$e-1$)	**(6.69$e-2$)**
UF6	6.7744$e-1$	7.3813$e-1$	2.5695$e+0$	7.5191$e-1$	5.3913$e-1$	**5.0980$e-1$**	5.2598$e-1$	5.7701$e-1$
	(2.48$e-1$)	(1.39$e-1$)	(7.97$e-1$)	(1.41$e-1$)	(1.32$e-1$)	**(6.57$e-2$)**	(7.62$e-2$)	(1.12$e-1$)
UF7	2.7369$e-1$	2.0424$e-1$	7.3512$e-1$	1.9615$e-1$	4.5705$e-1$	2.1657$e-1$	1.6563$e-1$	**5.4736$e-2$**
	(1.10$e-1$)	(7.04$e-2$)	(1.32$e-1$)	(7.53$e-2$)	(1.11$e-1$)	(1.05$e-1$)	(9.49$e-2$)	**(4.64$e-3$)**
UF8	3.6710$e-1$	3.5163$e-1$	7.4397$e-1$	3.4069$e-1$	5.7647$e-1$	3.4331$e-1$	3.5650$e-1$	**2.9892$e-1$**
	(3.96$e-2$)	(3.51$e-2$)	(1.22$e-1$)	(3.59$e-2$)	(2.54$e-1$)	(5.95$e-2$)	(4.59$e-2$)	**(8.93$e-2$)**
UF9	5.0642$e-1$	4.9393$e-1$	7.6655$e-1$	4.8937$e-1$	5.1151$e-1$	4.5574$e-1$	4.8648$e-1$	**1.3840$e-1$**
	(9.51$e-2$)	(5.80$e-2$)	(1.05$e-1$)	(4.30$e-2$)	(9.53$e-2$)	(5.93$e-2$)	(7.66$e-2$)	**(2.20$e-2$)**
UF10	2.2698$e+0$	2.2711$e+0$	4.6292$e+0$	2.3699$e+0$	7.2339$e-1$	1.0200$e+0$	1.1947$e+0$	**5.1341$e-1$**
	(8.82$e-1$)	(3.91$e-1$)	(8.83$e-1$)	(4.58$e-1$)	(9.63$e-2$)	(2.54$e-1$)	(4.26$e-1$)	**(9.72$e-2$)**
DTLZ1	5.1405$e+0$	4.1193$e+0$	3.4648$e+1$	3.9783$e+0$	3.9079$e+0$	2.8103$e+0$	**2.3748$e+0$**	4.4077$e+1$
	(2.27$e+0$)	(1.44$e+0$)	(2.14$e+1$)	(1.45$e+0$)	(2.46$e+0$)	(1.69$e+0$)	**(1.02$e+0$)**	(1.40$e+1$)
DTLZ2	9.8293$e-2$	5.8990$e-2$	1.0238$e-1$	5.4934$e-2$	**5.4947$e-2$**	5.5105$e-2$	5.6639$e-2$	1.0454$e-1$
	(3.67$e-3$)	(1.55$e-3$)	(1.29$e-2$)	(1.70$e-4$)	**(2.57$e-4$)**	(2.49$e-4$)	(6.03$e-4$)	(7.54$e-3$)
DTLZ3	2.0041$e+1$	1.0668$e+1$	9.7820$e+1$	1.1436$e+1$	1.7152$e+1$	**7.5904$e+0$**	7.7873$e+0$	1.2671$e+2$
	(8.37$e+0$)	(5.22$e+0$)	(5.91$e+1$)	(5.42$e+0$)	(8.56$e+0$)	**(4.27$e+0$)**	(4.32$e+0$)	(3.40$e+1$)
DTLZ4	1.7246$e-1$	1.5477$e-1$	2.2674$e-1$	1.9844$e-1$	5.3593$e-1$	2.6570$e-1$	1.2354$e-1$	**1.1168$e-1$**
	(2.01$e-1$)	(1.97$e-1$)	(1.22$e-1$)	(2.52$e-1$)	(3.20$e-1$)	(2.45$e-1$)	(1.74$e-1$)	**(3.22$e-2$)**

Table 5: Continued.

Problems	AGEII	ANSGAIII	DGEA	NSGAIII	MOEAD	ARMOEA	AGEMOEA	HDMOPSO
DTLZ5	4.3520e − 2	1.1211e − 2	5.7895e − 2	1.2431e − 2	3.2469e − 2	5.7577e − 3	**5.7118e − 3**	2.2555e − 2
	(4.17e − 4)	(1.32e − 3)	(1.07e − 2)	(1.53e − 3)	(7.07e − 4)	(1.51e − 4)	**(1.72e − 4)**	(2.85e − 3)
DTLZ6	9.0139e − 2	2.3606e − 2	1.3535e + 0	2.0972e − 2	1.0585e − 1	5.1735e − 3	**5.1700e − 3**	7.5019e − 3
	(1.57e − 1)	(4.81e − 2)	(1.06e + 0)	(5.03e − 3)	(2.37e − 1)	(5.18e − 4)	**(1.03e − 4)**	(1.07e − 3)
DTLZ7	1.3022e − 1	1.1682e − 1	4.6400e − 1	9.8316e − 2	2.3783e − 1	3.0406e − 1	2.4587e − 1	**9.8273e − 2**
	(5.00e − 2)	(1.06e − 1)	(3.09e − 1)	(7.55e − 2)	(2.25e − 1)	(2.62e − 1)	(2.55e − 1)	**(1.16e − 2)**
Best/All	0/22	0/22	0/22	1/22	0/22	2/22	4/22	15/22

Table 6: HV values of HDMOPSO and MOEAs on 22 test problems.

Problems	AGEII	ANSGAIII	DGEA	NSGAIII	MOEAD	ARMOEA	AGEMOEA	HDMOPSO
ZDT1	6.0606e − 1	5.8450e − 1	4.4826e − 3	5.8438e − 1	5.1233e − 1	6.3314e − 1	6.7352e − 1	**7.2106e − 1**
	(2.62e − 2)	(2.08e − 2)	(2.09e − 2)	(1.89e − 2)	(7.39e − 2)	(2.92e − 2)	(8.65e − 3)	**(1.67e − 4)**
ZDT2	2.2885e − 1	2.0517e − 1	3.0196e − 2	2.1195e − 1	8.7068e − 2	3.8852e − 3	3.4325e − 1	**4.4479e − 1**
	(6.09e − 2)	(4.47e − 2)	(8.64e − 2)	(2.79e − 2)	(2.21e − 2)	(9.36e − 3)	(2.39e − 2)	**(3.14e − 4)**
ZDT3	5.4307e − 1	5.4222e − 1	3.1248e − 2	5.3874e − 1	5.8582e − 1	5.8511e − 1	5.8154e − 1	**6.5930e − 1**
	(2.30e − 2)	(1.73e − 2)	(4.54e − 2)	(1.11e − 2)	(5.62e − 2)	(5.30e − 2)	(1.81e − 2)	**(5.18e − 4)**
ZDT4	2.1891e − 1	0.0000e + 0	0.0000e + 0	0.0000e + 0	1.6314e − 1	8.3158e − 3	5.6042e − 2	**7.2109e − 1**
	(1.74e − 1)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(1.11e − 1)	(2.36e − 2)	(9.66e − 2)	**(1.94e − 4)**
ZDT6	1.4324e − 1	0.0000e + 0	3.7415e − 1	0.0000e + 0	2.7816e − 1	2.6587e − 3	6.8723e − 2	**3.8917e − 1**
	(7.39e − 2)	(0.00e + 0)	(7.16e − 2)	(0.00e + 0)	(3.84e − 2)	(7.76e − 3)	(4.98e − 2)	**(1.28e − 4)**
UF1	4.9545e − 1	5.1256e − 1	9.6182e − 2	4.9318e − 1	4.2186e − 1	5.5831e − 1	5.5828e − 1	**5.8358e − 1**
	(7.03e − 2)	(3.37e − 2)	(7.52e − 2)	(5.06e − 2)	(4.74e − 2)	(2.44e − 2)	(3.20e − 2)	**(5.07e − 3)**
UF2	6.0395e − 1	6.1498e − 1	5.1072e − 1	6.1363e − 1	5.5211e − 1	6.3112e − 1	6.3627e − 1	**6.7344e − 1**
	(1.02e − 2)	(7.79e − 3)	(2.83e − 2)	(6.45e − 3)	(2.50e − 2)	(6.46e − 3)	(6.09e − 3)	**(5.90e − 3)**
UF3	1.7440e − 1	1.8893e − 1	1.1804e − 1	1.8488e − 1	3.0498e − 1	2.0647e − 1	2.1327e − 1	**4.4309e − 1**
	(2.58e − 2)	(8.50e − 3)	(2.74e − 2)	(7.68e − 3)	(3.55e − 2)	(2.47e − 2)	(1.86e − 2)	**(4.02e − 2)**
UF4	2.9950e − 1	3.1424e − 1	2.7710e − 1	3.1270e − 1	2.8594e − 1	3.3215e − 1	3.3372e − 1	**3.6976e − 1**
	(6.87e − 3)	(3.80e − 3)	(1.02e − 2)	(4.40e − 3)	(6.95e − 3)	(3.56e − 3)	(3.93e − 3)	**(4.36e − 3)**
UF5	6.5633e − 3	0.0000e + 0	0.0000e + 0	3.4451e − 4	0.0000e + 0	**3.0366e − 2**	2.5807e − 2	1.9772e − 2
	(1.76e − 2)	(0.00e + 0)	(0.00e + 0)	(1.53e − 3)	(0.00e + 0)	**(4.49e − 2)**	(4.27e − 2)	(2.21e − 2)
UF6	3.0354e − 2	1.0728e − 2	0.0000e + 0	9.3560e − 3	**9.3876e − 2**	4.8097e − 2	3.7781e − 2	1.3127e − 2
	(4.03e − 2)	(1.60e − 2)	(0.00e + 0)	(1.44e − 2)	**(6.38e − 2)**	(3.11e − 2)	(3.16e − 2)	(1.40e − 2)
UF7	2.8984e − 1	3.2586e − 1	1.8432e − 2	3.2919e − 1	2.2900e − 1	3.6263e − 1	3.8745e − 1	**5.0640e − 1**
	(8.79e − 2)	(6.92e − 2)	(2.96e − 2)	(6.85e − 2)	(6.15e − 2)	(7.07e − 2)	(8.00e − 2)	**(6.73e − 3)**
UF8	1.7267e − 1	2.2906e − 1	2.1604e − 2	2.2585e − 1	1.5686e − 1	3.0044e − 1	2.9239e − 1	**3.5290e − 1**
	(5.81e − 2)	(3.97e − 2)	(2.25e − 2)	(3.84e − 2)	(6.22e − 2)	(2.17e − 2)	(3.78e − 2)	**(1.67e − 2)**
UF9	2.2393e − 1	2.5375e − 1	6.5632e − 2	2.5881e − 1	2.9108e − 1	3.1461e − 1	2.9501e − 1	**6.3437e − 1**
	(7.64e − 2)	(5.14e − 2)	(5.11e − 2)	(4.10e − 2)	(5.77e − 2)	(4.40e − 2)	(5.19e − 2)	**(2.37e − 2)**
UF10	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	3.4782e − 2	3.0422e − 5	0.0000e + 0	**3.1914e − 1**
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(2.76e − 2)	(1.16e − 4)	(0.00e + 0)	**(4.40e − 2)**
DTLZ1	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)
DTLZ2	5.4387e − 1	5.4919e − 1	4.4879e − 1	5.5562e − 1	5.5533e − 1	**5.5668e − 1**	5.5633e − 1	4.7654e − 1
	(2.34e − 3)	(2.81e − 3)	(2.60e − 2)	(6.43e − 4)	(6.50e − 4)	**(7.25e − 4)**	(7.20e − 4)	(1.15e − 2)
DTLZ3	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0	0.0000e + 0
	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)	(0.00e + 0)
DTLZ4	5.0701e − 1	5.0866e − 1	3.9182e − 1	4.8909e − 1	3.2464e − 1	4.6540e − 1	**5.2232e − 1**	5.1071e − 1
	(1.04e − 1)	(8.65e − 2)	(8.39e − 2)	(1.19e − 1)	(1.68e − 1)	(1.06e − 1)	**(8.82e − 2)**	(1.03e − 2)
DTLZ5	1.7846e − 1	1.9462e − 1	1.6149e − 1	1.9336e − 1	1.8245e − 1	1.9839e − 1	**1.9854e − 1**	1.8259e − 1
	(1.15e − 3)	(8.57e − 4)	(8.23e − 3)	(1.04e − 3)	(5.75e − 4)	(2.37e − 4)	**(1.87e − 4)**	(3.45e − 3)
DTLZ6	1.6313e − 1	1.8686e − 1	4.4790e − 2	1.8986e − 1	1.6070e − 1	1.9938e − 1	**1.9957e − 1**	1.9487e − 1
	(3.65e − 2)	(3.31e − 2)	(7.18e − 2)	(2.64e − 3)	(5.38e − 2)	(2.48e − 4)	**(7.46e − 5)**	(4.19e − 3)
DTLZ7	2.3918e − 1	2.6170e − 1	1.5277e − 1	2.6464e − 1	2.4079e − 1	2.4795e − 1	2.5682e − 1	**2.6668e − 1**
	(6.10e − 3)	(1.22e − 2)	(6.87e − 2)	(9.25e − 3)	(1.66e − 2)	(2.58e − 2)	(2.65e − 2)	**(3.33e − 3)**
Best/All	0/22	0/22	0/22	0/22	1/22	2/22	3/22	14/22

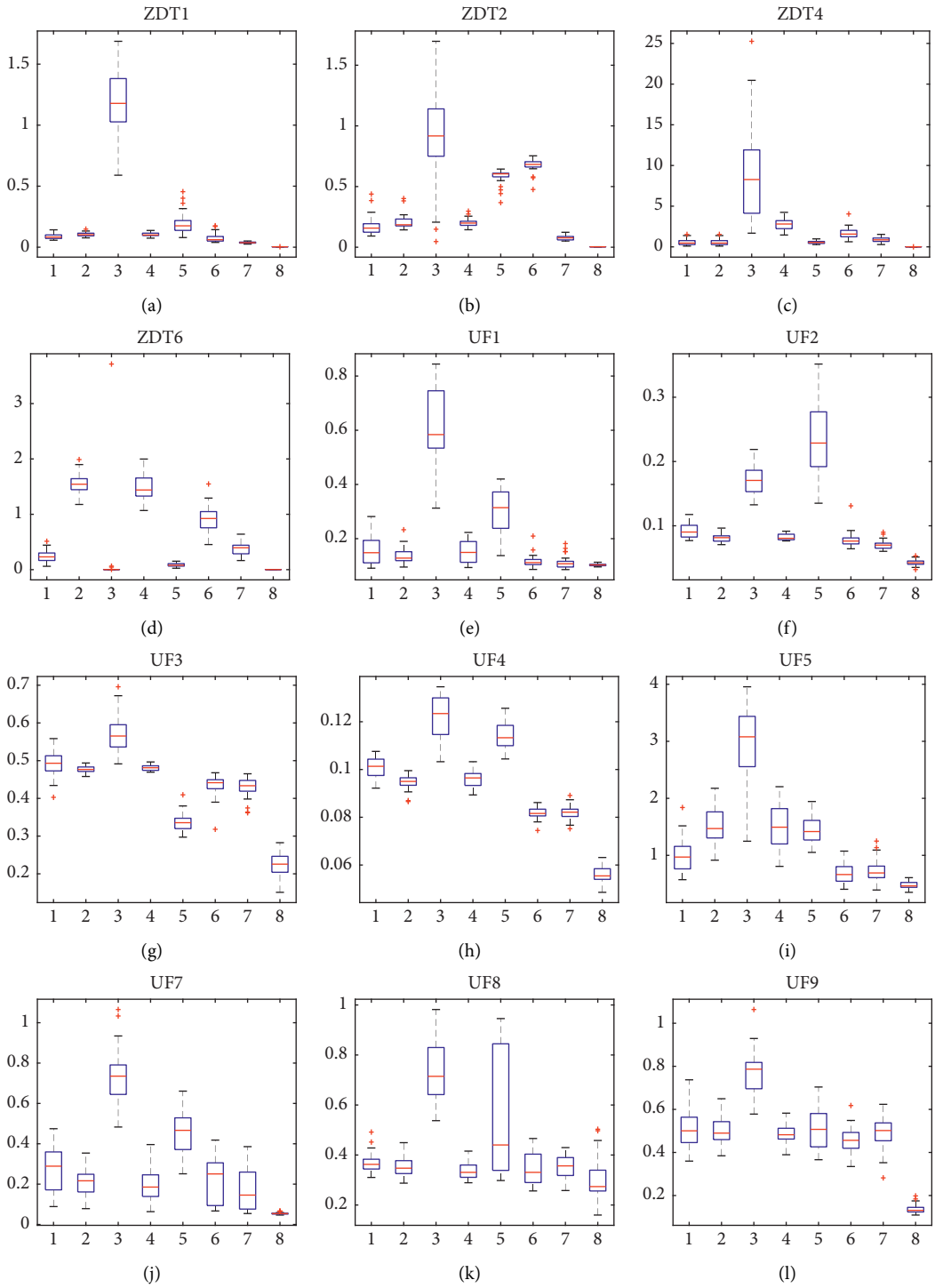

Figure 9: Continued.

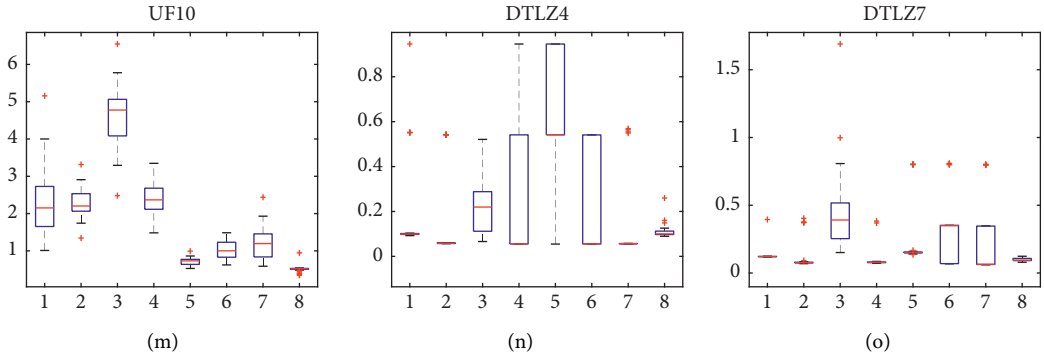

Figure 9: IGD value Statistical boxplot of HDMOPSO and MOEAs on ZDT1, ZDT2, ZDT4, ZDT6, UF1-5, UF7-10, DTLZ4, and DTLZ7 problems. (1, 2, 3, 4, 5, 6, 7, and 8 in each Statistical boxplot represent AGEII, ANSGAIII, DGEA, NSGAIII, MOEAD, ARMOEA, AGEMOEA, and HDMOPSO, respectively).

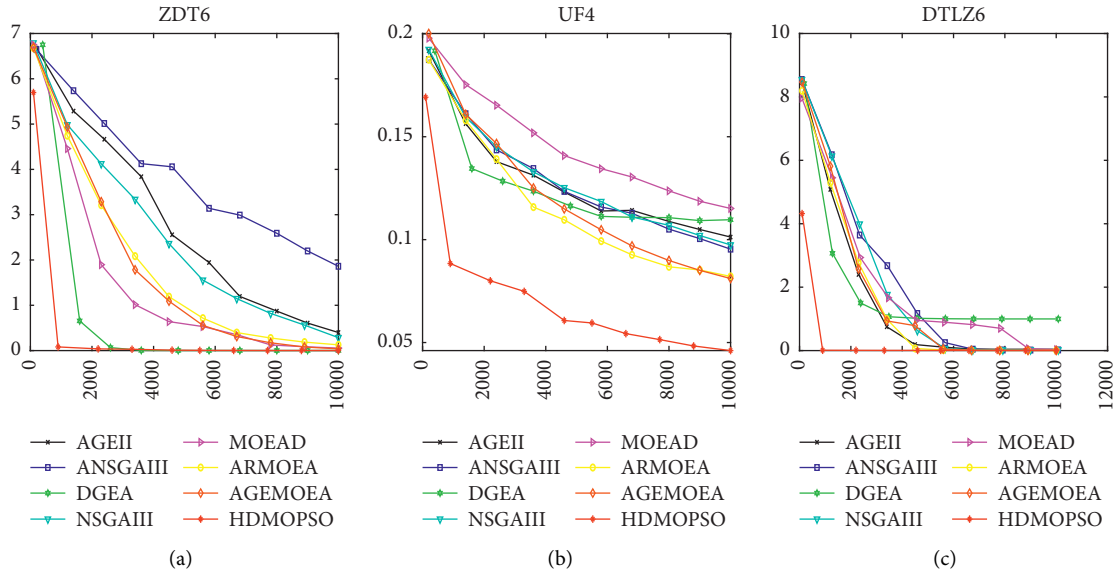

Figure 10: IGD values convergence trajectory of HDMOPSO and MOEAs comparison on ZDT6, UF4, and DTLZ6 problems.

obtained from Table 5 that DTLZ4 has obvious advantages over other algorithms compared to DTLZ7, it can be seen from Figure 7 that it is not obvious compared to the ANSGAIII and AGEMOEA on DTLZ4. Nothing is more stable than the ANSGAIII and AGEMOEA and the average value is no better than it except for outliers. Although there are no outliers in DTLZ7, there is still a little difference compared with the ANSGAIII and NSGAIII.

Figure 10 shows the IGD convergence trajectory obtained by HDMOPSO and MOEAs running 10000 times in ZDT6, UF4, and DTLZ6. HDMOPSO is compared with the MOEAs, it not only has an absolute advantage in the convergence speed, but also shows an obvious advantage in the convergence value, which also reflects the fast convergence speed of the particle algorithm.

Finally, Figure 11 shows the Pareto front simulation of HDMOPSO and MOEAs on ZDT4. It can be seen from the figure that the algorithms in this paper almost all converge to the Pareto front, and the distribution is good. No other algorithm has a convergent Pareto front.

Through the above analysis of the comparison of HDMOPSO and MOEAs, HDMOPSO performs better in 22 test problems and 15 test problems with IGD values. Figure 9 also shows good stability. 14 of HV performed better. From Figures 10 and11, it can be seen briefly that HDMOPSO has obvious advantages in terms of convergence speed and convergence. Especially, it shows obvious advantages compared with MOEAs in the ZDT and UF test problems. Therefore, HDMOPSO is more competitive than MOEAs.

4.6. Computational Complexity of HDMOPSO. The complexity of HDMOPSO algorithm mainly depends on the complexity of the three parts in Section 3 of this article. The following will be analyzed one by one according to the three parts. First, in the maintenance of the external archive, the

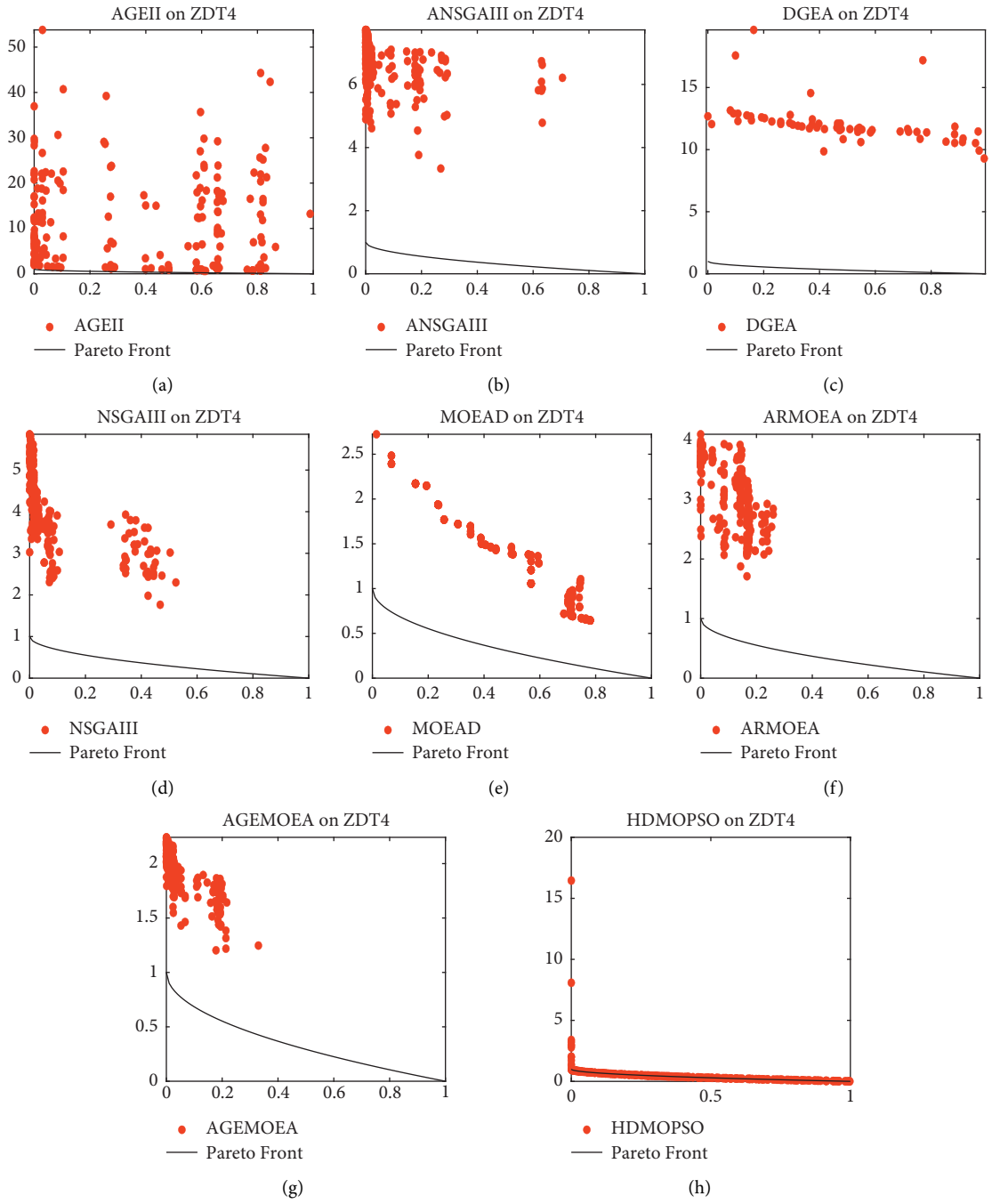

Figure 11: The Pareto front simulation of HDMOPSO and MOEAs on ZDT4 problem.

strategy of rebuilding the hypercube is used and there is only one operator. So complexity is $O(m \times div)$ (m is the target number. div is the number of divisions of the coordinate axes). Second, the complexity of the opposite mutation with nonlinear decreasing is O (N) (N is the number of particles) according to the pseudocode in Figure 3. Thirdly, the setting of the social part based on the method of combining hyperspace and distance mainly consists of three parts. First, the hypercube are selected one by one and the optimal positions of each hypercube are averaged. Second, each particle finds its nearest average position. Finally, the particle determines the flight of the social part according to the shortest average position it finds. So the complexity of the third part is $O(A \times a) + O(N \times A \times m) + O(N)$ (A is the number of optimal solutions to store in the external archive. a is the number of optimal solutions in the hypercube). According to the operation rules of the symbol O, the complexity above can be simplified as O $(A(N \times M + a) + m \times div)$.

In addition, tic and toc codes in MATLAB software are also used to calculate the running time of each algorithm when evaluating (FEs) 10,000 times (unit: *second*) It can be

Table 7: Running times of HDMOPSO and MOPSOs on 22 test problems.

Problems	FEs	MOPSO	MOPSOCD	dMOPSO	SMPSO	NMPSO	MPSOD	CMOPSO	HDMOPSO
ZDT1	10000	1.8954e − 01	2.1995e − 01	3.0756e − 01	1.5614e − 01	5.9784e + 00	9.8553e − 01	9.6836e − 01	6.2169e + 00
ZDT2	10000	9.6836e − 01	1.8677e − 01	3.0587e − 01	1.5278e − 01	5.7637e + 00	8.5288e − 01	9.3910e − 01	6.3731e + 00
ZDT3	10000	1.6025e − 01	1.3933e − 01	3.1736e − 01	1.5136e − 01	3.7805e + 00	8.8155e − 01	3.6910e − 01	5.2505e + 00
ZDT4	10000	1.4136e − 01	1.1535e − 01	3.3401e − 01	1.2446e − 01	1.2178e + 00	4.0052e − 01	4.4006e − 01	4.1853e + 00
ZDT6	10000	1.6083e − 01	3.0051e − 01	3.0535e − 01	1.4522e − 01	6.9666e + 00	6.0061e − 01	1.9409e + 00	1.1422e + 01
UF1	10000	1.5675e − 01	1.3745e − 01	3.4933e − 01	1.5089e − 01	5.5508e − 01	1.1290e + 00	2.4897e − 01	3.5535e + 00
UF2	10000	1.8101e − 01	1.4694e − 01	4.0179e − 01	1.6999e − 01	9.5793e − 01	1.1746e + 00	2.8913e − 01	9.6000e + 00
UF3	10000	1.8831e − 01	2.4739e − 01	3.6602e − 01	1.7367e − 01	1.0686e + 00	1.1528e + 00	2.8029e − 01	4.0495e + 00
UF4	10000	1.6966e − 01	1.5075e − 01	3.8537e − 01	1.6703e − 01	2.6208e + 00	1.2591e + 00	2.9030e − 01	9.8464e + 00
UF5	10000	1.6004e − 01	1.5048e − 01	3.8623e − 01	1.5731e − 01	5.4259e − 01	1.0700e + 00	2.9916e − 01	1.3402e + 01
UF6	10000	1.7337e − 01	1.5025e − 01	3.9267e − 01	1.6589e − 01	5.3052e − 01	1.1377e + 00	2.9791e − 01	2.6452e + 00
UF7	10000	1.6423e − 01	1.4502e − 01	3.4968e − 01	1.6433e − 01	7.6412e − 01	1.1829e + 00	3.1794e − 01	8.6314e + 00
UF8	10000	2.2966e − 01	1.7421e − 01	3.9023e − 01	2.1134e − 01	1.5602e + 00	9.6623e − 01	5.5715e − 01	4.8477e + 00
UF9	10000	2.2266e − 01	1.7022e − 01	3.8962e − 01	2.0454e − 01	1.3716e + 00	1.0086e + 00	5.3158e − 01	8.8649e + 00
UF10	10000	1.9929e − 01	1.7375e − 01	3.7649e − 01	1.9299e − 01	6.3617e − 01	9.8017e − 01	4.5164e − 01	1.4856e + 00
DTLZ1	10000	1.9451e − 01	1.7336e − 01	4.2895e − 01	1.7963e − 01	1.9483e + 00	1.4728e + 00	3.4286e − 01	4.4390e + 00
DTLZ2	10000	2.5668e − 01	2.1649e − 01	4.0326e − 01	1.9691e − 01	2.8527e + 00	1.4684e + 00	1.4363e + 00	3.8444e + 00
DTLZ3	10000	3.6626e − 01	1.8473e − 01	4.4459e − 01	1.9127e − 01	1.0501e + 00	1.0501e + 00	3.6626e − 01	1.9394e + 00
DTLZ4	10000	2.4343e − 01	1.7660e − 01	4.4017e − 01	1.9800e − 01	2.8646e + 00	1.2343e + 00	1.2343e + 00	9.8830e − 01
DTLZ5	10000	2.5944e − 01	1.9055e − 01	4.4258e − 01	2.0864e − 01	2.2953e + 00	9.3072e − 01	9.1520e − 01	2.3477e + 00
DTLZ6	10000	2.4352e − 01	3.2588e − 01	4.4845e − 01	2.1578e − 01	3.6248e + 00	1.3673e + 00	8.9641e − 01	5.3365e + 00
DTLZ7	10000	2.3549e − 01	3.5500e − 01	3.5500e − 01	2.0576e − 01	2.9453e + 00	6.8225e − 01	9.0226e − 01	1.0615e + 01

Table 8: Running times of HDMOPSO and MOEAs on 22 test problems.

Problems	FEs	AGEII	ANSGAIII	DGEA	NSGAIII	MOEAD	ARMOEA	AGEMOEA	HDMOPSO
ZDT1	10000	1.6906e + 01	4.7437e − 01	4.2843e − 01	2.9046e − 01	1.7671e + 00	6.4435e − 01	2.5825e − 01	6.2169e + 00
ZDT2	10000	1.3348e + 01	4.4615e − 01	4.2499e − 01	2.9139e − 01	1.7384e + 00	3.3364e − 01	2.4050e − 01	6.3731e + 00
ZDT3	10000	1.4356e + 01	4.5518e − 01	4.0627e − 01	2.8519e − 01	1.7178e + 00	8.8620e − 01	2.6861e − 01	5.2505e + 00
ZDT4	10000	8.1142e + 00	3.9137e − 01	3.3979e − 01	2.7679e − 01	1.6371e + 00	2.7897e − 01	2.4642e − 01	4.1853e + 00
ZDT6	10000	8.1726e + 00	4.0665e − 01	3.8004e − 01	2.7077e − 01	1.6758e + 00	2.7440e − 01	2.1910e − 01	1.1422e + 01
UF1	10000	2.1910e − 01	4.8534e − 01	4.2519e − 01	2.8050e − 01	1.8027e + 00	3.9434e − 01	2.3187e − 01	3.5535e + 00
UF2	10000	3.0681e + 01	4.8148e − 01	4.5153e − 01	2.9503e − 01	2.0535e + 00	9.2059e − 01	3.0219e − 01	9.6000e + 00
UF3	10000	1.5124e + 01	4.8845e − 01	4.4217e − 01	3.0029e − 01	2.0223e + 00	4.1813e − 01	2.5723e − 01	4.0495e + 00
UF4	10000	6.6710e + 01	4.8374e − 01	4.5074e − 01	3.1567e − 01	1.9588e + 00	1.0578e + 00	2.9759e − 01	9.8464e + 00
UF5	10000	5.5514e + 00	4.6839e − 01	4.1768e − 01	2.9088e − 01	1.9183e + 00	3.2421e − 01	2.3231e − 01	1.3402e + 01
UF6	10000	7.9447e + 00	4.6724e − 01	4.2344e − 01	2.9147e − 01	1.9433e + 00	3.3038e − 01	2.3917e − 01	2.6452e + 00
UF7	10000	2.2273e + 01	4.5917e − 01	4.2218e − 01	2.8460e − 01	2.0121e + 00	4.4279e − 01	2.5829e − 01	8.6314e + 00
UF8	10000	1.0657e + 01	5.7163e − 01	4.1730e − 01	3.2666e − 01	2.1912e + 00	2.0018e + 00	4.8602e − 01	4.8477e + 00
UF9	10000	1.0713e + 01	5.5393e − 01	4.1814e − 01	3.1820e − 01	2.5255e + 00	1.4048e + 00	3.9560e − 01	8.8649e + 00
UF10	10000	8.5054e + 00	5.5813e − 01	4.1871e − 01	3.2024e − 01	2.5230e + 00	1.0070e + 00	2.7937e − 01	1.4856e + 00
DTLZ1	10000	5.0505e + 00	5.7524e − 01	4.2878e − 01	3.2626e − 01	2.4902e + 00	9.6535e − 01	3.6443e − 01	4.4390e + 00
DTLZ2	10000	8.7972e + 01	5.2632e − 01	4.6034e − 01	3.6273e − 01	2.4898e + 00	4.6962e + 00	7.8888e − 01	3.8444e + 00
DTLZ3	10000	1.0630e + 02	5.2184e − 01	3.9689e − 01	3.8554e − 01	2.4625e + 00	4.2094e + 00	7.3357e − 01	1.9394e + 00
DTLZ4	10000	1.0630e + 02	5.2184e − 01	3.9689e − 01	3.8554e − 01	2.4625e + 00	4.2094e + 00	7.3357e − 01	9.8830e − 01
DTLZ5	10000	1.8164e + 02	6.6040e − 01	3.8343e − 01	4.0999e − 01	2.5314e + 00	4.1013e + 00	6.1905e − 01	2.3477e + 00
DTLZ6	10000	1.8625e + 01	6.8678e − 01	4.2519e − 01	4.1715e − 01	2.5983e + 00	3.8306e + 00	7.1626e − 01	5.3365e + 00
DTLZ7	10000	2.0237e + 01	5.7245e − 01	3.7660e − 01	3.7607e − 01	2.4422e + 00	4.0381e + 00	6.4493e − 01	1.0615e + 01

seen from Tables 7 and 8 that HDMOPSO is almost in the same order of magnitude as MOPSOs and MOEAs in most of the test problems, but HDMOPSO algorithm is still a little slower than other algorithms. However, statistics after operation depend on computer hardware, software and other environmental factors. Therefore, the data at this time is for reference only.

5. Conclusions and Future Work

In this paper, a multiobjective particle swarm optimization combining hypercube and distance is proposed. Establish a combination of hypercube and distance in the external archive to set the social part of the particle speed update, and improve the optimization ability of the algorithm. In

addition, a nonlinear decreasing opposite mutation strategy is also used. It not only increases the exploration ability, but also prevents the algorithm from premature convergence. Finally, the hypercube technology is used to control the external archive, which improves the computing power of the algorithm while also increasing the diversity of the algorithm. HDMOPSO is compared with MOPSOs and MOEAs. The results show that HDMOPSO has good performance, especially the ZDT and UF test problems have strong competitiveness compared with MOPSOs and MOEAs. Therefore, the strategy of this paper is an effective improvement of particle swarm optimization in solving multiobjective optimization problems.

In the future research work, on the one hand, the optimization of the algorithm is studied. It can be seen from the data in this paper that the optimization effect of the algorithm is poor in complex problems, so more effective strategies are needed to solve most of MOPs. On the other hand, the main reason proposed by the algorithm is that it can effectively solve practical problems in reality, so it is one of the key points of further research to solve practical problems in reality.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant nos.71461027) and Innovative talent team in Guizhou Province (Qian Ke HE Pingtai Rencai[2016]5619).

References

[1] Y. Tian, H. Wang, X. Zhang, and Y. Jin, "Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization," *Complex & Intelligent Systems*, vol. 3, no. 4, pp. 247–263, 2017.

[2] B. Qu, C. Li, J. Liang, L. Yan, K. Yu, and Y. Zhu, "A self-organized speciation based multi-objective particle swarm optimizer for multimodal multi-objective problems," *Applied Soft Computing*, vol. 86, Article ID 105886, 2020.

[3] Y. Zhang, S. Cheng, Y. Shi, D.-w. Gong, and X. Zhao, "Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm," *Expert Systems with Applications*, vol. 137, pp. 46–58, 2019.

[4] N. Shahabi Sani, M. Manthouri, and F. Farivar, "A multi-objective ant colony optimization algorithm for community detection in complex networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 1, pp. 5–21, 2020.

[5] Q. Lin, Y. Ma, J. Chen et al., "An adaptive immune – inspired multi-objective algorithm with multiple differential evolution strategies," *Information Sciences*, vol. 430-431, pp. 46–64, 2018.

[6] Q. Zhu, Q. Lin, W. Chen et al., "An external archive – guided multiobjective particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2794–2808, 2017.

[7] Y. Yang, T. Zhang, W. Yi et al., "Deployment of multistatic radar system using multi-objective particle swarm optimisation," *IET Radar, Sonar & Navigation*, vol. 12, no. 5, pp. 485–493, 2018.

[8] R. Leng, A. Ouyang, Y. Liu, L. Yuan, and Z. Wu, "A multi-objective particle swarm optimization based on grid distance," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 3, Article ID 2059008, 2020.

[9] Q. Feng, Q. Li, P. Chen et al., "Multi-objective particle swarm optimization algorithm based on adaptive angle division," *IEEE Access*, vol. 7, pp. 916–930, 2019.

[10] X. Zhang, X. Zheng, R. Cheng, J. Qiu, and Y. Jin, "A competitive mechanism based multi-objective particle swarm optimizer with fast convergence," *Information Sciences*, vol. 427, pp. 63–76, 2018.

[11] G. Peng, Y.-W. Fang, W.-S. Peng, D. Chai, and Y. Xu, "Multi-objective particle optimization algorithm based on sharing-learning and dynamic crowding distance," *Optik*, vol. 127, no. 12, pp. 5013–5020, 2016.

[12] 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.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, Perth, Australia, November 1995.

[14] F. Van Den Bergh, *An Analysis of Particle Swarm Optimizers (PSO)*, pp. 78–85, University of Pretoria, Pretoria, 2001.

[15] C. C. Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2, pp. 1051–1056, 2002.

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

[17] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, and E. Alba, "SMPSO: a new PSO-based metaheuristic for multi-objective optimization," *IEEE Symposium on computational intelligence in multi criteria decision-making (MCDM)*, vol. 1, pp. 66–73, 2009.

[18] S. Zapotecas Martinez and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pp. 69–76, Dublin, Ireland, July 2011.

[19] Q. Hui Li and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[20] Y. Hu, Y. Zhang, and D. Gong, "Multi-objective particle swarm optimization for feature selection with fuzzy cost," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 874–888, 2020.

[21] Z. Yong, Y. Li-Juan, Z. Qian, and S. Xiao-Yan, "Multi-objective optimization of building energy performance using a particle swarm optimizer with less control parameters,"

Journal of Building Engineering, vol. 32, Article ID 101505, 2020.

[22] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 1, pp. 695–701, Vienna, Austria, November 2005.

[23] L. Kang, R.-S. Chen, W. Cao, and Y.-C. Chen, "Non-inertial opposition-based particle swarm optimization and its theoretical analysis for deep learning applications," *Applied Soft Computing*, vol. 88, Article ID 106038, 2020.

[24] Y. Zhang, D.-W. Gong, and J. Cheng, "Multi-objective particle swarm optimization approach for cost-based feature selection in classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 64–75, 2015.

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

[26] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," *Evolutionary multi-objective optimization*, vol. 7, no. 2, pp. 105–145, 2005.

[27] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multi-objective optimization test instances for the CEC 2009 special session and competition," vol. 264Technical Report CES 487, pp. 1–30, University of Essex, Colchester, UK and Nanyang technological University, Singapore, 2008.

[28] C. A. C. Coello and N. C. Cortes, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[29] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[30] C. R. Raquel and P. C. Naval, "An effective use of crowding distance in multi-objective particle swarm optimization," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pp. 257–264, Washington DC USA, June 2005.

[31] Q. Lin, S. Liu, Q. Zhu et al., "Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 32–46, 2016.

[32] C. Dai, Y. Wang, and M. Ye, "A new multi-objective particle swarm optimization algorithm based on decomposition," *Information Sciences*, vol. 325, pp. 541–557, 2015.

[33] M. Wagner and F. Neumann, "A fast approximation-guided evolutionary multi-objective algorithm," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 687–694, Amsterdam, The Netherlands, July 2013.

[34] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference – point based non-dominated sorting approach, part II: handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2013.

[35] C. He, R. Cheng, and D. Yazdani, "Adaptive offspring generation for evolutionary large – scale multi-objective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, 2020.

[36] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference – point-based non-dominated sorting approach, part *i*: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2013.

[37] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator-based multiobjective evolutionary algorithm with reference point Adaptation for better versatility," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609–622, 2018.

[38] A. Panichella, "An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 595–603, Prague, Czech Repulic, July 2019.

[39] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: a MATLAB platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.