mathematics

MDPI

Article

Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy for Many-Objective Optimization

Gui Li [1], Gai-Ge Wang [1,2,3,*] and Shan Wang [4,5]

1 Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China; liguiatqingdao@gmail.com
2 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
3 Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006, China
4 Faculty of Arts and Humanities, University of Macau, Macau 999078, China; shanwang@um.edu.mo
5 Institute of Collaborative Innovation, University of Macau, Macau 999078, China
* Correspondence: wgg@ouc.edu.cn

Abstract: Due to the complexity of many-objective optimization problems, the existing many-objective optimization algorithms cannot solve all the problems well, especially those with complex Pareto front. In order to solve the shortcomings of existing algorithms, this paper proposes a co-evolutionary algorithm based on dynamic learning strategy. Evolution is realized mainly through the use of Pareto criterion and non-Pareto criterion, respectively, for two populations, and information exchange between two populations is used to better explore the whole objective space. The dynamic learning strategy acts on the non-Pareto evolutionary to improve the convergence and diversity. Besides, a dynamic convergence factor is proposed, which can be changed according to the evolutionary state of the two populations. Through these effective heuristic strategies, the proposed algorithm can maintain the convergence and diversity of the final solution set. The proposed algorithm is compared with five state-of-the-art algorithms and two weight-sum based algorithms on a many-objective test suite, and the results are measured by inverted generational distance and hypervolume performance indicators. The experimental results show that, compared with the other five state-of-the-art algorithms, the proposed algorithm achieved the optimal performance in 47 of the 90 cases evaluated by the two indicators. When the proposed algorithm is compared with the weight-sum based algorithms, 83 out of 90 examples achieve the optimal performance.

Keywords: evolutionary algorithms (EAs); many-objective optimization; coevolution; dynamic learning; performance indicators

Citation: Li, G.; Wang, G.-G.; Wang, S. Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy for Many-Objective Optimization. *Mathematics* **2021**, *9*, 420. https://doi.org/10.3390/math9040420

Academic Editor: Alicia Cordero

Received: 1 February 2021
Accepted: 19 February 2021
Published: 20 February 2021

1. Introduction

In recent years, with the development of technology, more and more new problems appear in the field of industry or control and so on. This problem is usually characterized by containing more than one objective function to be optimized, and these objective functions contradict each other. Such problems are generally called multi-objective optimization problems (MOPs) or many-objective optimization problems (MaOPs). Generally, MOPs are problems having two or three objectives, and MaOPs contain more than four objectives [1]. A MaOP is defined as follows:

$$\begin{aligned} &minimize \quad F(x) = \{f_1(x), f_2(x), \ldots, f_M(x)\} \\ &subject \ \ to \quad x \in X \end{aligned} \tag{1}$$

where M is the objective number. $x = (x_1, x_2, \ldots, x_n)$ is decision variable, and $X \subseteq R^n$ is the decision space of the n-dimensional real number field. F is a mapping from decision space to objective space, and F(x) contains M different objective functions $f_i(x)$ ($i = 1, 2, \ldots, M$).

Generally, the optimal solutions of MaOPs are distributed on Pareto front (PF), and the solutions on PF generally show the trade-off on all M objectives. Therefore, it is impossible to obtain an optimal solution by an optimization method to make it optimal on all objectives. What is needed to solve MaOPs is to obtain a set that has a finite number of solutions, so that the solutions in this solution set can well represent the whole PF, no matter in terms of convergence or diversity. The EAs had its unique advantages in solving MaOPs because of its population-based characteristics. After years of development, scholars from all over the world have proposed various many-objective optimization algorithms (MaOEAs) to solve MaOPs. Because different MaOPs also have different characteristics, currently no general algorithm can perfectly solve all MaOPs.

MaOPs usually have more than three objective numbers, so the objective space cannot be visualized, and the high-dimension calculation equation can only be obtained through derivation in solutions with fewer objectives. For example, grid-based evolutionary algorithm (GrEA) [2] deduces the high-dimensional grid calculation equation through a two-dimensional equation. Moreover, with the increasing of the objective number, the number of non-dominated individuals in the population will also increase exponentially. Studies have shown that almost all the solutions in the obtained population will be non-dominated when $M > 12$ [3]. As a result, the selection pressure of the algorithm based on non-dominated sorting is reduced, which makes the algorithm unable to solve the MaOPs well. In addition, the shape and density of PF vary greatly for different problems, which brings great challenges to obtaining a solution set with good convergence and diversity.

To overcome these difficulties with MaOPs, a number of MaOEAs have been proposed. For example, on the basis of fast and elitist multi-objective genetic algorithm (NAGA-II) [4], reference points are introduced to guide individual convergence and help evolution through the concept of domination, called NSGA-III [5,6]. Knee point-driven evolutionary algorithm (KnEA) [7] used knee points to guide individual convergence, and GrEA introduced the concept of grid to choose the better of two non-dominated individuals. In addition, some indicator-based algorithms are proposed, such as indicator-based selection in multi-objective search (IBEA) [8] and fast hypervolume-based algorithm (HypE) [9], which adopt $I_{\varepsilon+}$ [10] and hypervolume (HV) [11,12] indicators as the criteria for selecting individuals, respectively. The selection process is the evolution process of the whole population or individuals towards the direction with better indicator values. Finally, there is the evolutionary algorithm based on decomposition (MOEA/D) [13], which adopts the idea of mathematical decomposition to decompose a MaOP into M subproblems for simultaneous optimization. Common decomposition approaches include weighted sum approach, Tchebycheff approach, and penalty-based boundary intersection approach. There is a new way called MOEA/D-PaS [14] to combine the decomposition with Pareto adaptive scalarizing methods to balance the selection pressure toward the Pareto optimal and the algorithm robustness to PF.

At the same time, these MaOEAs also contain some disadvantages. The convergence speed of the Pareto-based algorithm is slow, or even unable to converge to PF. Indicator-based algorithms often tend to favor one or some of special regions of PF. Although the convergence speed of an indicator-based algorithm is generally relatively fast, the diversity of the solution set is usually poor. Besides, the decomposition-based MaOEAs are very dependent on the selection of decomposition approaches, such as weighted sum method in dealing with the non-convex problem (in the case of minimization) of PF shape, and not all Pareto optimal vectors can surely be obtained [13]. In addition, when the shape and density of PF are very complex and changeable, the traditional method to generate a weight vector is not suitable for this environment.

Existing multi-objective optimization algorithms (MOEAs) will still be affected by the increased number of objectives when dealing with MOPs, especially MAOPs. Moreover, the complexity of the PF also brings great challenges to the MOEAs. However, the dynamic learning strategy can pay more attention to the improvement of convergence when the population has poor convergence in the early evolutionary stage, and pay more attention

to the improvement of diversity in the late evolutionary stage. In addition, coevolution is a promising idea to improve the quality of individuals in a population through the mode of cooperation (or competition) between multiple populations (or subpopulations). Through coevolution, some key information of the population (such as the evolutionary state of the population) can be obtained in the process of algorithm iteration. The information of the population can be fed back flexibly to change the dynamic level (this will be discussed in Section 4) of dynamic learning strategies. In conclusion, the combination of a dynamic learning strategy and a coevolution model is a promising way to improve the convergence and diversity of the population.

The rest of this paper will be arranged as follows. Section 2 introduces the related works of the coevolutionary algorithm, other background, and the motivation for a two-population coevolution algorithm with dynamic learning strategy (DL-TPCEA). Section 3 will give the background of the MaOPs. The algorithm framework, process details, and parameter settings will be introduced in Section 4. Sections 5 and 6 carry out the analysis of experiments and the conclusion, respectively.

2. Related Works

In biology, the concept of coevolution is defined as follows: an adaptive coevolution in which two interacting species develop in the course of evolution. An evolutionary type of genetic evolution in which one species is influenced by another. At the biological level, it has several major significances:

1. Promote the increase of biological diversity;
2. Promote the co-adaptation of species;
3. Maintain the stability of the biological community.

This idea was successfully introduced into computer algorithms. More and more researchers begin to pay attention to the performance improvement brought by coevolution strategy to the EAs. In order to adapt to increasingly complex problems, Potter et al. incorporated the idea of coevolution into EAs. It extended the evolutionary paradigm of the time and described an architecture that evolved subcomponents into collections of collaborative species [15]. Then they analyzed the robustness of cooperative coevolutionary algorithms (CCEAs), which provided a theoretical basis for the effectiveness of coevolutionary strategies [16]. Wiegand et al. also used evolutionary game theoretic (EGT) models to help understand CCEAs and analyze whether CCEAs are really suitable for optimization tasks [17]. One of the EGT models was the multi-population symmetric game, which can be used to analyze and model the coevolutionary algorithm. In this context, coevolution tended to decompose an evolving population into several small subpopulations and ensured that each subpopulation did not interfere with each other. Then the individual in the population was optimized continuously through the cooperation of each subpopulation. The effectiveness of CCEAs is verified using CCEAs to solve complex problems (or structure) [18,19].

The coevolution strategy of CCEAs can group the population, which is suitable for large-scale optimization problems (LSOPs). The dimension of decision variables in LSOP is too high, so grouping is a good solution at present. This is also the initial application scenario of CCEAs. Yang et al. considered that traditional CCEAs can only deal with and decompose separable LSOPs, but often cannot solve the inseparable LSOPs. Therefore, a stochastic grouping scheme and adaptive weighting were introduced into problem decomposition and coevolution, and a new differential evolutionary algorithm was used to replace the traditional evolutionary algorithm. Through this improvement, the algorithm can effectively optimize the 1000-dimensional indivisible problem [20]. In addition, a multilevel coevolution (MLCC) [21] framework was proposed to solve LSOPs. MLCC was a framework that determined the size of a group when the problem was decomposed. MLCC constructed a set of problem decomposers based on random grouping strategies with different group sizes, and used an adaptive mechanism to select decomposers based on historical performance to self-adapt between different levels.

CCEAs is also applied to optimization problems in other scenarios. Liu et al. used cooperative coevolution (CC) to improve the speed of evolutionary programming (EP) [22]. However, this study showed that the time cost increased linearly as the dimension of the problems was increased. CC was also used to deal with global optimization and find global optimal solutions [23]. Chen et al. proposed a cooperative coevolution with variable interaction learning (CCVIL) framework [24], which treated all variables as independent and put them into separate groups, and then continuously merged groups when found the relationship between them at the iteration.

In addition to the above optimization problems, many researchers in recent years have begun to apply CC to MaOPs. Tan et al. combined SPEA2 and CC effectively and proposed SPEA2-CC [25]. After experimental comparison, the performance of SPEA2-CC was significantly better than that of the original SPEA2 as the number of objectives increases. SPEA2-CC provided theoretical support for the scalability of performance of CC in MaOPs.

A lot of researchers combined CC with the preference of the decision maker to deal with MaOPs, which led to the preference-inspired coevolutionary algorithm (PICEA) [26]. Researchers have shown that PICEA can handle not only MOPs, but also MaOPs [27]. The experiments showed that the preference-driven coevolution algorithm was superior to some other methods under the measurement of a hypervolume indicator. One defect of PICEA was the uneven distribution of the obtained solutions on PF, which means poor diversity. In order to solve this problem, an improved fitness allocation method (PICEA-g) [28] was proposed, which can consider the density information of solutions. In addition, a new preference-inspired coevolutionary algorithm using weight vectors (PICEA-w) [29] was proposed. The algorithm coevolved with the candidate solution during the search process. Coevolution adaptively constructed the appropriate weights in the optimization process, thus, effectively led the candidate solutions to the PF.

Liang et al. proposed a multi-objective coevolutionary algorithm based on a decomposition method [30], which used subpopulations to enhance objectives. Running on multiple subpopulations and external archive via the differential evolution (DE) operator to improve each objective and diversify the trade-offs of external archiving solutions. In addition, when an objective was not optimized, computing resources on that objective were allocated to other objectives and external archive strengthens the tradeoffs on all objectives. In addition, PF was approximated by parallel subpopulations [31]. Firstly, the MaOPs were decomposed by using a uniformly distributed weight vector, and then each subpopulation was associated with a weight vector. Using subpopulations to optimize each subproblem, and elite individuals in subpopulations were used to produce offspring. This can not only enhanced the diversity of the population, but also accelerated the convergence rate.

There were also studies that used new approaches to further improve CC performance on MaOPs. Shu et al. proposed a preference-inspired coevolution algorithm (PICEA-g/LPCA) with local principal component analysis (PCA) oriented goal vectors [32]. PICEA-g/LPCA was a further improvement on the basis of PICEA-g, and it used local PCA to extend the ability of PICEA-g and improved the convergence. In addition, a co-evolutionary particle swarm optimization algorithm with a bottleneck objective learning (BOL) strategy [33] was proposed to meet the convergence and diversity challenges in finite population size. In this algorithm, multiple subpopulations coevolved to maintain diversity. The BOL strategy was also used to improve convergence across all objectives. Elitist learning strategy (ELS) was also used to jump out of local PFs, and juncture learning strategy (JLS) was used to develop areas that are missing in PF.

Coevolution strategies have now been applied to many problems. In addition to general MaOPs, there are dynamic interval many-objective optimization problems (IMaOPs) [34,35], large-scale multi-objective optimization problems (LSMOPs) [33,36], and feature selection [37]. Finally, some recent work also uses coevolution or learning techniques [38,39] to deal with MaOPs [40–42].

As described in Section 1, the solution set obtained by Pareto-based MOEAs has a good distribution on PF, but there is a general problem of slow convergence and the performance will decline with the increase of the objective number. Non-Pareto MOEAs shows good convergence performance, but not good diversity performance. The solution set of non-Pareto MOEAs tends to converge to one or some special regions of PF, especially in the case of extremely irregular PF. Li et al. proposed a bi-criterion evolution (BCE) framework in 2015 [43], which performed well in many-objective optimization. In the BCE framework, two populations evolved simultaneously. One used the Pareto criterion (PC) and the other used the non-Pareto criterion (NPC). The aim was to take advantage of both approaches and compensate for their shortcomings. These two parts work together to promote evolution through the exchange of information between populations. Among them, NPC population led PC population to converge, and PC population can make up for the loss of NPC population in diversity. The two operations included in the framework, population maintenance and individual exploration, were used to preserve good non-dominated individuals and explore unexplored areas of NPC population respectively. Although the framework of BCE did not use the method of subpopulation coevolution in CC, the idea of cooperation between the two populations should also belong to CC.

Dynamic learning strategy (DLS) can consider the evolutionary state of solution set during algorithm iteration. It is well known that the initial population of MOEAs is randomly generated without specific requirements. The randomly generated solution is to take the value of the solution in the domain $[x_{\min}, x_{\max}]$ in the case of a normal (Gaussian) distribution. The equation is as follows:

$$x = x_{\min} + rand * (x_{\max} - x_{\min}) \tag{2}$$

where *rand* is a random number generated by a standard normal distribution. So, the convergence of initial population is very poor, just random individuals in the solution space. DLS can pay attention to this point, so that in the initial stage of population evolution, it can ensure rapid convergence of solutions by using more computing resources to the selection of convergence-related solutions. As the iteration goes on, the solutions converge towards PF. At this time, it is necessary to keep the solution set more diversified. Therefore, with the iteration of MaOEAs, computational resources will gradually incline to diversity-related solutions to maintain a better distribution of the population on PF.

Therefore, an effective combination of BCE and DLS may yield relatively good results, as confirmed by the experimental results in Section 5. This paper takes advantage of the coevolution of information interaction between the two populations, and introduces DLS into the environmental selection of NPC to better enable the evolution of NPC population. The cost value (CV) [44] will be selected as indicator. This algorithm will be called DL-TPCEA. The detailed algorithm will be described in Section 4.

3. The Background of MaOPs

At present, many single objective optimization problems in the optimization field have become the focus of research, such as workshop scheduling problems [45–49] and numerical optimization problems [50,51]. Most of these can be solved by classical algorithms and their improved versions, such as artificial bee colony algorithm (ABC) [47,48,52,53], particle swarm optimization (PSO) [51,54], monarch butterfly optimization (MBO) [55–58], ant colony optimization (ACO) [59,60], krill herd algorithm (KH) [52,61–64], elephant herding optimization (EHO) [65–67], and other metaheuristic algorithms [68–77]. However, there are some problems in many-objective optimizations, which cannot be solved by single objective techniques. Because of the conflicts between objectives, all objectives cannot be optimized simultaneously using single objective techniques. MaOPs also have different characteristics, which are described in more detail below. The current MaOPs in the field of many-objective optimizations are mainly divided into the following categories:

(1) General MaOPs: As mentioned in Equation (1), general MaOPs are problems with M conflicting objectives. The overall goal of solving MaOPs is to obtain a solution set that

can characterize PF, but at the same time there are a variety of problems. For objective numbers, MaOPs are more difficult to resolve than MOPs. Low dimensional optimization is mainly solved by non-dominated sorting, such as NSGA-II [4] and improving the strength of the Pareto evolutionary algorithm (SPEA2) [78]. The non-dominated sorting is described as follows: for the minimization problem, taking two vectors x_1 and x_2 in Ω, if and only if $f_i(x_1) \leq f_i(x_2)$ for each i in $\{1, 2, \ldots, M\}$ and $f_j(x_1) < f_j(x_2)$ for at least one j in $\{1, 2, \ldots, M\}$. Let us call it $F(x_1)$ Pareto dominates $F(x_2)$, and the notation is $F(x_1) > F(x_2)$, and if, and only if, no point x in Ω to satisfy $F(x) > F(x^*)$, called $F(x^*)$ Pareto optimal solution and x^* is Pareto optimal point, and the set of all Pareto optimal solutions is PF mentioned above, the set of all Pareto optimal points is called Pareto Set (PS).

(2) Large-scale MaOPs: these problems often involve high dimensional decision variables. In general, MOPs are called large-scale MOPs (LSMOPs) [79] when its decision variable dimension $N > 100$. The performance of the MaOEAs will decrease as the number of decision variables increases. For example, when using a mutation operator to mutate individuals, the probability of producing good individuals after mutation will also decrease due to the large dimension of decision variables. There are some researches on LSMOPs. At present, most of this work is based on classifying decision variables and dealing with them separately.

Ma et al. proposed a many-objective evolutionary algorithm based on decision variable analysis (MOEA/DVA) [80], which divided the whole population into convergence-related variables and diversity-related variables through decision variable analysis strategy. Moreover, MOEA/DVA optimized the two parts respectively, so that the convergence and diversity of the population were maintained well. Zhang et al. [81] proposed an evolutionary algorithm based on decision variable clustering for large-scale many-objective optimization problems (LMEA). LMEA used k-means clustering method and takes the angle between solutions and the direction of convergence as the feature to carry out the clustering, and divided the decision variables into convergence-related variables and diversity-related variables. LMEA further classified the unclassified individuals in MOEA/DVA to promote the convergence and diversity of the population. In addition, Chen et al. [1] proposed an evolutionary algorithm based on covariance matrix adaptation evolution strategy and scalable small subpopulation to solve large-scale many-objective optimization problems (S^3-CMA-ES).

The above work is based on the premise of grouping decision variables to deal with large-scale many-objective optimization problems, which makes a great contribution to the large-scale many-objective optimization.

(3) Dynamic MaOPs (DMaOPs): DMaOPs add time (environment) variation to the general MaOPs. It is described as follows:

$$\begin{aligned} minimize \quad & F(x) = \{f_1(x, \mathrm{t}), f_2(x, t), \ldots, f_M(x, t)\} \\ & subject \ \ to \quad x \in X \end{aligned} \tag{3}$$

where t is time (environment) variation. When time (environment) changes, PF of the MaOPs also changes, that is, the optimal solution set in the previous state is not necessarily the optimal solution set in the current state. This means that the algorithm is not only required to adapt to the many-objective environment to optimize multiple objectives, but also needs the changes brought by the response time (environment). When the time (environment) changes, the algorithm can respond quickly and get the optimal solution set in the latest environment.

In the environment of DMaOPs, many excellent algorithms have been proposed. Liu et al. proposed a dynamic multi-population particle swarm optimization algorithm (DP-DMPPSO) based on decomposition and prediction [82]. Using the archive update mechanism based on the objective space decomposition and the population prediction mechanism to accelerate the convergence, the results show that the algorithm has a good effect in DMaOPs processing. Finally, there are also many dynamic multi-objective evo-

lutionary algorithms (DMOEAs) that use various optimization strategies [83–87] to deal with DMaOPs.

The main purpose of this paper is to solve the general MaOPs with high dimensional objective space, using the Pareto-based and non-Pareto-based methods for coevolution of the two populations, respectively. The two populations make use of the advantages of each other and make up for the disadvantages, which is very promising to solve the difficulty of optimization in the high-dimensional objective space. The details will be introduced in Section 4.

4. The Framework of DL-TPCEA

In this part, the specific process of the dynamic learning strategy will be introduced first, and then the DL-TPCEA will be introduced. All algorithmic details such as parameter control and algorithmic flow are given.

4.1. Dynamic Learning Strategy

4.1.1. The Description of DLS

Previous MOEAs generally used the immutable evolution strategy during iteration. For example, NSGA-II used non-dominated sorting to select the non-dominated solutions in population to control the convergence, and then used crowding distance to select among the non-dominated solutions to improve the diversity of the population. This method is very time-consuming because of Pareto sorting, and tends to have poor convergence effect when the number of objectives is relatively large. However, DLS will make full use of the advantages of fast running speed and good convergence effect of indicator-based algorithm. Moreover, the enhancement of diversity is further strengthened to balance the convergence and diversity of the solution set. This paper will take a two-objective problem as an example to illustrate the advantages of DLS over traditional immutable evolutionary strategies.

As shown in Figure 1a, after the population initialization, the distribution of these individuals in the objective space is very chaotic. In other words, the convergence and diversity of the population are poor. According to the current population, the priority is to get these individuals to converge to PF as soon as possible. This will be guided by indicator-based method. For example, in a practical engineering problem, the individuals on PF are those who can minimize the cost. In this case, more computing resources should be allocated to the process of convergence-related operations to achieve rapid convergence of the population to PF. A small part of the computational resources are then allocated to operations that increase the diversity of the population to ensure that the diversity of the population is not particularly poor.

After the above operation, the distribution of individuals in the population in the objective space will gradually move towards PF, as shown in Figure 1b. However, the convergence level of the whole population is not enough at this time, so high selection pressure is still needed to promote convergence. As the iteration goes on, the distribution of individuals in population in the objective space will gradually become close to PF, as shown in Figure 1c. As introduced in Section 1, the indicator-based algorithm converges quickly but loses diversity easily. The example in Figure 1c shows that these individuals are close to PF under the guidance of the indicator, but the convergence position is more inclined to the central region of PF. At this point, more computing resources need to be tilted to increase the diversity of the population, such as the preference to keep the individual A and the individual B in Figure 1c into the next generation. By changing the computational resource allocation according to the evolutionary state of the population, individuals in the population can maintain good convergence and diversity. As in Figure 1d, the individuals in the resulting solution set are uniformly distributed on PF.

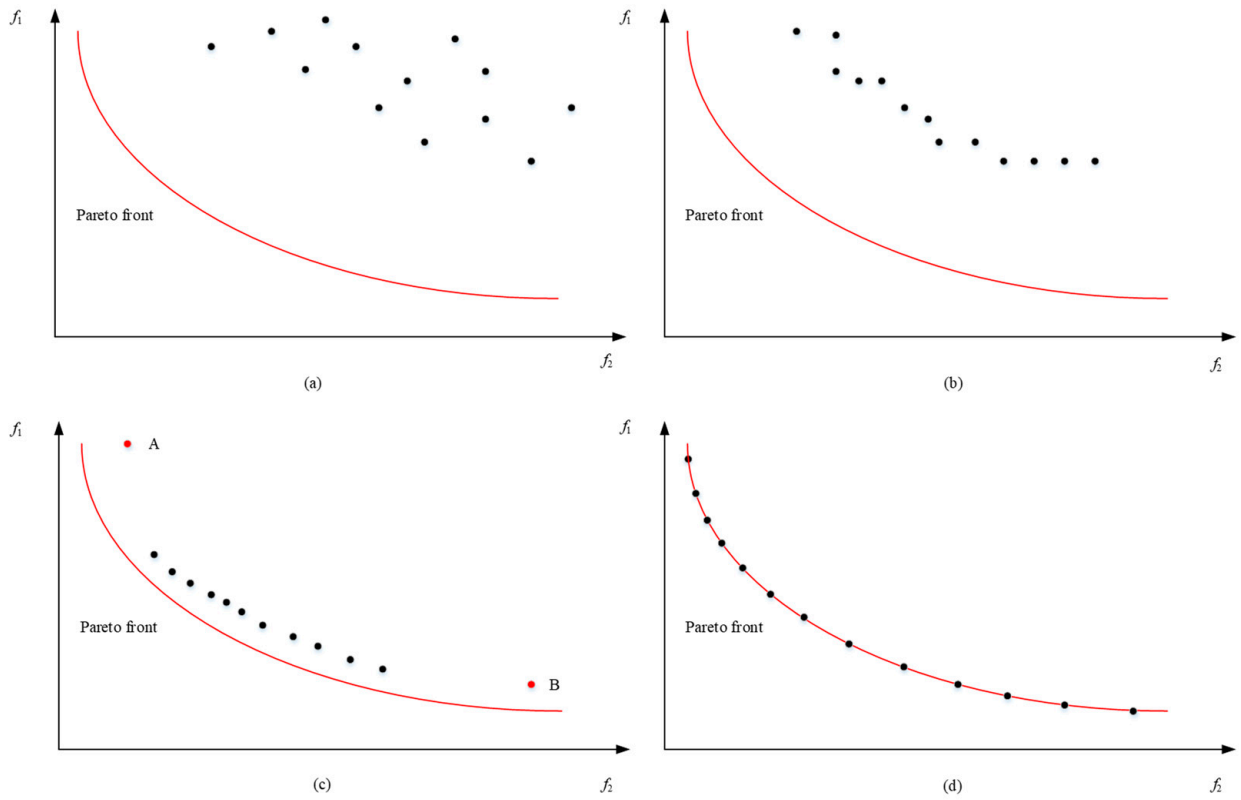

Figure 1. The process of dynamic learning strategy. (**a**) The state after initialization; (**b**) the state at the beginning of evolution; (**c**) the state of late evolution; (**d**) the state at the end of evolution.

4.1.2. The Details of DLS

The above is only a brief description of the steps of DLS; the following is a detailed explanation of the specific process of DLS. First, supposing the population size of MaOEAs is N. In an iteration, N new individuals are generated by crossover and mutation operators, at which time the original individuals and newly generated individuals form a new population, which is denoted as P_{2N} here. What needs to be done next is to select N individuals that are most conducive to maintain convergence and diversity through environmental selection as the initial population P_{new} of the next iteration. These operations are accomplished through DLS.

As shown in Algorithm 1, the $2N$ individuals are first layered by non-dominated sorting (Line 1, Algorithm 1). Here, *FrontNo* is the number of layers that each individual resides in, and *MaxFNo* is the largest number of layers that are non-dominated. Where *MaxFNo* satisfies:

$$\sum_{i=1}^{MaxFNo-1} L_i \leq N \&\& \sum_{i=1}^{MaxFNo} L_i > N \tag{4}$$

where L_i represents the number of individuals in the ith non-dominated layer ($i = 1, 2, \ldots,$ *MaxFNo*). Here, the non-dominated individuals in Layer 1 to layer *MaxFNo*-1 will preferentially select into P_{new} (Line 2, Algorithm 1), and then continue to select the remaining individuals in Layer *MaxFNo*.

Although in the case of 2- or 3-objective problems (MOPs), it may be more clearly layered. This makes the number of individuals in Layer *MaxFNo* smaller, which means fewer individuals are selected through DLS. However, with the increase of the objective number, the proportion of non-dominated individuals in the whole population also increased, almost all individuals are non-dominated when the objective number is more than 12 which is described in Section 1. This leads to an increase in the number of individuals in

Layer *MaxFNo*, even if all individuals in the population are in Layer *MaxFNo*. This also makes the role of DLS greatly increased, and become more useful in solving MaOPs.

Algorithm 1 Dynamic Learning Strategy

Input: Population: P_{2N}; Convergence factor: α; Parameter of p-norm: p
Output: Next generation population : P_{new}
1: $[FrontNo, MaxFNo] = NDSort()$
2: $P_{new} = P_{2N}(FrontNo < MaxFNo)$
3: Calculate C_n by Eq. (5)
4: Calculate D_n by Eq. (6)
5: Calculate the CV indicator of individuals by Eqs. (7) and (8)
6: Calculate the $L_p - norm$ distance between individuals
7: **for** $i = 1 : C_n$ **do**
8: Choose the individual by CV and put into P_{new}
9: **end for**
10: **for** $i = 1 : D_n$ **do**
11: Choose the individual by $PNormDis$ and put into P_{new}
12: **end for**

Next, the values of C_n and D_n will be calculated according to needs (Lines 3–4, Algorithm 1), representing the number of convergence-related individuals and diversity-related individuals that need to be preserved, respectively. This is also the key for DLS to ensure dynamic computational resource allocation within algorithm iteration. The calculation equation of the C_n is as follows:

$$C_n = \left\lceil R_{gen} \times \alpha \times (1 - \frac{gen}{max\ gen}) \right\rceil \tag{5}$$

where *gen* represents the current number of iterations and *maxgen* represents the maximum number of iterations. R_{gen} represents the total number of individuals that need to be selected at Layer *MaxFNo* at generation *gen*. Moreover, $\alpha \in [0, 1]$ is a convergence factor that controls the rate of convergence of the population. Through experimental research, it is found that when α is about 0.9, the performance can reach the best. In this way, the convergence speed can be achieved quickly at the same time; it will not fall into the local optimal. The symbol $\lceil \cdot \rceil$ rounds the element to the nearest integer greater than or equal to that element. Then the number of diversity-related individuals that needs to be preserved will continue to be calculated. The calculation of D_n is as follows:

$$D_n = R_{gen} - C_n \tag{6}$$

After C_n and D_n are calculated, two indicators of convergence and diversity will be calculated for the individuals in population, and R_{gen} individuals in Layer *MaxFNo* will be retained according to the rules of DLS. In this paper, cost value (*CV*) [44] will be selected as the convergence-related indicator. Let $F(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_M(x_i))$ be the objective vector for individual x_i ($i = 1, 2, \ldots, N$). Then the mutual evaluation of individual x_i by individual x_j is as follows:

$$cv_{ij} = \max_m \{ f_m(x_j) / f_m(x_i) \},\ m = 1, 2, \ldots, M \tag{7}$$

Then the mutual evaluation of each individual in the population is as follows:

$$CV_i = \min_{j \neq i} cv_{ij}, j = 1, 2, \ldots, N \tag{8}$$

This indicator will not be affected by the change of objective number, and the characteristics of this indicator can be clearly understood according to Equations (7) and (8). The first point is that x_i is a non-dominated individual when $CV_i > 1$, and the second is that x_i is a dominated individual when $CV_i \leq 1$. Therefore, we can use this indicator as convergence-related indicator to select the individuals in Layer *MaxFNo*. The individuals that have larger CV will be retained, in other words, retaining the individuals who have better convergence.

As for the diversity-related indicators, the distance between individuals in the population is generally used as the evaluation criterion. For example, Euclidean distance is used to calculate the crowding distance in NSGA-II. In this paper, the L_p-norm-based distance is selected to calculate the distance between individuals in the population. It has been experimentally demonstrated that the L_p-norm-based distance is more efficient than the Euclidean distance, Manhattan distance, etc., especially when dealing with MaOPs [88]. Parameter p of L_p-norm-based distance is recommended as $1/M$. Therefore, L_p-norm-based distance is selected as the diversity-related indicator in this paper.

After the calculation of two indicators for individuals in the population, the individuals in Layer *MaxFNo* were selected and saved to P_{new} according to C_n and D_n, until the size of P_{new} reached N.

4.2. The Framework of BCE

There are two main populations in BCE, namely NPC population and PC population. These two parts use the non-Pareto method and Pareto method to evolve the population, respectively. For the NPC population, any non-Pareto evolutionary criterion can be used directly. However, when the next generation is produced through competition, the environmental selection needs to select individuals from both NPC population and PC population (NPC selection). For PC population, non-dominated individuals from NPC and PC population are reserved (PC selection). Since the number of non-dominated solutions is unknown, population maintenance operation is carried out to eliminate some individuals with poor diversity when the number of non-dominated individuals is greater than the predefined threshold N.

Because of NPC population convergence speed is relatively fast, the individuals in NPC population can accelerate the convergence of PC population in PC selection. Because of the diversity of PC population is better, NPC population can explore the unexplored areas on PF through individual exploration operation and use the individuals in NPC population to enhance the diversity of PC population. In this way, the two populations interact with each other to promote the evolution of each population so that the convergence and diversity of the final solution set are good. The final output here is the PC population.

4.2.1. PC Selection and NPC Selection

The process of PC selection is to select non-dominated individuals from the mixed set of PC population and the new individuals produced by PC and NPC evolutions. NPC selection is based on the criteria of NPC evolution, which conducts environmental selection on a mixture of NPC populations and new individuals generated by PC populations. Assuming that the evolution of NPC populations uses indicator-based algorithms, then NPC selection is the selection of individuals having better indicator values in the mixed population for the next generation.

For the evolution of NPC populations, some algorithms rely on the information of the parent generation to update the individual, which is not feasible here. So individuals in the PC population are compared with individuals in the NPC population. If an individual in the PC population is better than one or more individuals in the NPC population according to the evolutionary criteria of the NPC population, then that individual (or a random one of those individuals) in the NPC population will be replaced by that individual in the PC population.

4.2.2. Population Maintenance

In PC selection, all non-dominated individuals are selected from a hybrid population of new individuals resulting from PC and NPC evolutions. Therefore, it is likely to make the number of non-dominated individuals larger than the preset threshold N (population size), especially when the objective number is large. Therefore, an effective means of population maintenance should be added to ensure that the PC population maintains a representative (with better convergence and diversity) group of individuals.

Population maintenance is to ensure the quality of individuals in a population through niche techniques. This is also a popular technique in EAs to assess the crowding degree of each individual in the population by the location and number of individuals in the niche (objective space). The crowding degree of individual p in population P is defined as follows:

$$D(p) = 1 - \prod_{q \in P, q \neq p} R(p, q) \tag{9}$$

$$R(p, q) = \begin{cases} d(p, q)/r, & if \quad d(p, q) \leq r \\ 1, & otherwise \end{cases} \tag{10}$$

where $d(p, q)$ is the Euclidian distance between individuals p and q, and r is the radius of the niche. Due to the size of each objective is different, in order to prevent the influence of problem size, the objective value of individuals in population will be normalized by maximum and minimum normalized first when calculating the distance.

It means that each is in the other's niche when the Euclidean distance between individuals p and q is less than r. This point can be seen in Equations (9) and (10), and the range of this crowding degree $D(p)$ is [0, 1]. Otherwise, there would be no effect on the crowding of these two individuals since $R(p, q) = 1$. When $d(p, q) \leq r$, the larger the Euclidean distance between the two individuals is, the smaller the calculated crowding degree will be, which means that the two individuals have a good crowding degree. So, this is a good way to eliminate the more crowded (poor diversity) individuals in the population.

Since the population is constantly evolving, it is not appropriate to set a fixed niche radius r. The setting of r must be related to the evolutionary state of the population. The radius r of the niche in BCE was set as the average Euclidian distance from each individual to k closest individuals in the population. The aim is to include one or more individuals in the niche of as many individuals as possible. Here, k is recommended to be set to 3 for better performance. Based on this crowding degree, the most congested individual in the population (the population of non-dominated individuals selected by PC selection) is eliminated each time and the crowding degree is recalculated. This process is repeated until the number of remaining non-dominated individuals is N.

4.2.3. Individual Exploration

The evolution part of the NPC population in BCE usually has high selection pressure; it converges quickly. However, the general NPC evolution tends to converge to one or more regions of PF, rather than the entire PF. This leads to a lack of diversity, as there are areas of PF that have never even been explored. It is through individual exploration that NPC evolution explores unknown areas on PF to achieve the purpose of increasing the diversity of NPC population. Individual exploration will explore some promising individuals in the PC population rather than all individuals in the PC population, because some individuals in the PC population have been well explored by NPC populations. These promising individuals generally have been eliminated (by NPC evolution), are less developed, or are not even visited in NPC evolution. From this point of view, the discussion is mainly focused on two types of individuals in the PC population:

1. Individuals whose niche has no NPC individual;
2. Individuals whose niche has only one NPC individual.

First of all, for the first group of individuals, these individuals are not in the niche of individuals in the NPC population. Such individuals are far away from the individuals in

the NPC population in the objective space, obviously not the individuals favored by the NPC criterion. However, it is such individuals that are in areas that NPC evolution has not explored. While the second kind of individuals have an NPC individual in niche, which is not a lot of individuals when considering that the k is set to 3. However, such individuals are likely to be located in areas where NPC evolution is incomplete, and it is necessary to explore such promising individuals.

During individual exploration, the above two kind of individuals contained in the PC population are first marked and stored in set S (individual sets to be explored). Then, the variation operation is carried out on the individuals in set S, and all the new individuals generated by the variation operator are stored in set T (the new individuals set generated by individual exploration) for the next PC selection. The variation operator here can be selected arbitrarily, but it should be noted that the number of parent individuals required by the selected variation operator should be changed accordingly.

The influence of the radius of the niche should also be considered here. A relatively small radius may allow all individuals in the PC population to be explored, as there may not be many NPC individuals in each individual's niche. The reverse is also true, larger radius may cause all individuals to remain unexplored. Therefore, a dynamically varying radius is used here, which can vary with the size of the PC population.

With the continuous evolutions of PC and NPC, more and more non-dominated individuals are produced, and the selection pressure of PC gradually decreases. This slows down PC evolution when the number of newly created non-dominated individuals exceeds the size of the remaining PC population that can be stored. This allows for less individual exploration, allowing the high selection pressure of NPC to play a greater role. The dynamic radius of the niche is set as follows:

$$r = (N\prime/N) * r_0 \tag{11}$$

where N represents the PC population size, and N' represents the size of the PC population before population maintenance, and r_0 represents the base niche radius calculated by means of population maintenance.

In the case of fixed computational resources (functional evaluations), this process of adaptive exploration is necessary according to the evolutionary state of the population. On the one hand, individual exploration can make up for the lack of diversity in NPC population. On the other hand, when there is a lack of convergence, more computing resources can be given to NPC evolution to accelerate convergence under higher selection pressure.

As shown in Figure 2, individual exploration on a 3-objective optimization problem is given. The triangle of coordinates in the figure represents the Pareto front of the problem, and the points in the figure represent the distribution of individuals in the population in the objective space. Suppose the NPC population is shown in Figure 2a, and the PC population is shown in Figure 2b. Due to the characteristics of NPC population, the obtained solution set may be distributed in some part of the Pareto front. For example, the population in Figure 2a is concentrated to the left and to the top of the Pareto front, while there is no individual distribution on the right. While PC population is relatively evenly distributed around the Pareto front, but the convergence is not good (some points do not converge to the Pareto front). Moreover, the role of individual exploration is to explore the promising individuals in the PC population to promote the diversity of the NPC population. It can be seen here that several individuals marked in red in Figure 2b are still promising individuals although they have not converged to the Pareto front. By exploring these solutions, it is possible to get some solutions that have never been explored in the PC population but have a good diversity. After continuous individual exploration, the diversity of PC population will also be improved and finally reach the state, as shown in Figure 2c. The population in Figure 2c well balances convergence and diversity, thus, achieving the purpose of individual exploration.

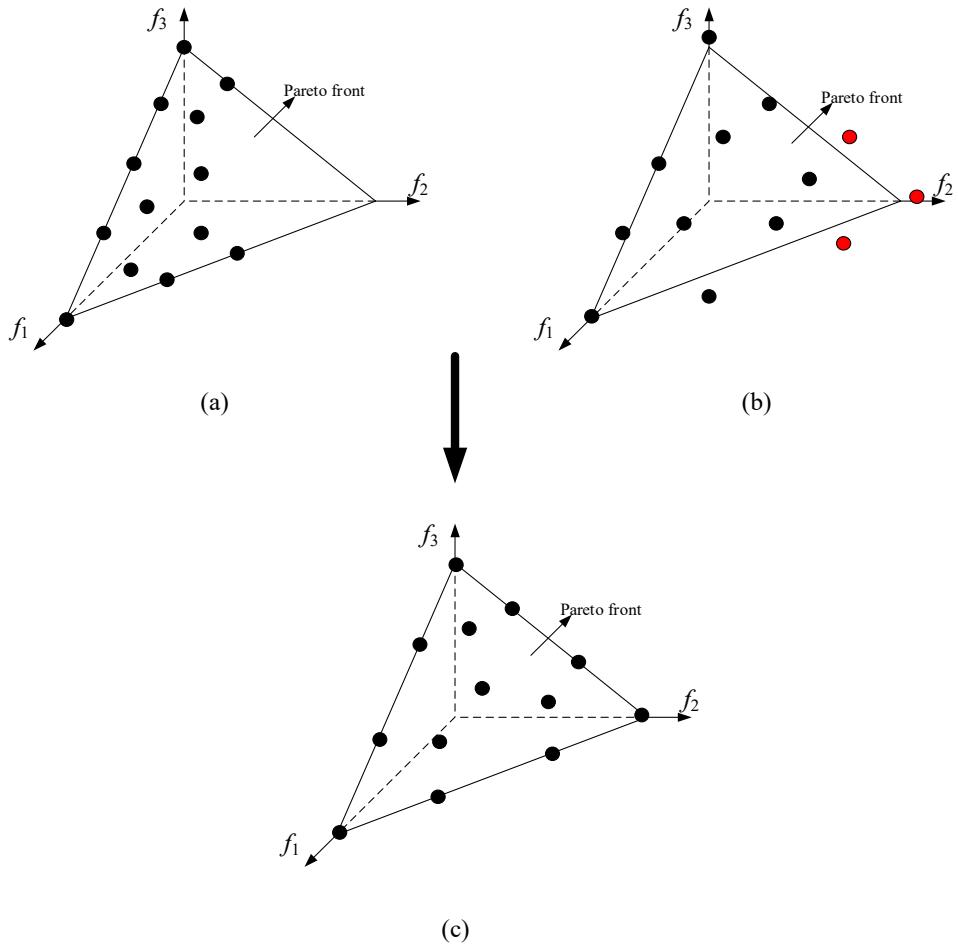

Figure 2. The individual exploration process in bi-criterion evolution (BCE). (**a**) The optimal solution set obtained by non-Pareto criterion; (**b**) the optimal solution set obtained by Pareto criterion; (**c**) the optimal solution set obtained by the individual exploration.

4.3. Two-Population Coevolutionary Algorithm with Dynamic Learning Strategy

From the above description, it can be seen that these strategies have great advantages and far-reaching significance in solving many-objective optimization problems. Next, we will introduce DL-TPCEA in the above context.

4.3.1. The Process of DL-TPCEA

Algorithm 2 gives the whole process of DL-TPCEA, from which it can be seen that the input parameters of DL-TPCEA include population size N, objective number M, and function evaluations (FEs). The final output is the population in which PC evolution. First, a parameter setting (Line 1, Algorithm 2) will be performed, which is mainly set for the current iteration number gen and the maximum iteration number $maxgen$ in Equation (5). In addition, L_p-norm-based distance is also used in DLS for diversity maintenance, where the value p is also initialized. The inverse of the objective number ($1/M$) will be used here as the value of p. The parameter settings (Line 5, Algorithm 2) in the later steps do the same thing.

Before the proceeding of BCE, the populations (PC population and NPC population) in both evolutionary approaches should first be initialized (Lines 2–3, Algorithm 2). The NPC population randomly generates N decision vectors with dimension D in the domain by satisfying the normal distribution, where D represents the dimension of the decision variable. The PC population is generated by PC selection on the NPC population. This ensures that the individuals stored in the PC population will always be non-dominated.

Algorithm 2 Framework of DL-TPCEA

Input: Population size: N; The number of objective: M; Function evaluations: FEs
Output: The final population : PC
1: $ParameterSet()$
2: $NPC = Initialization()$
3: $PC = PCSelection(NPC)$
4: **while** $NotTermination(PC)$ **do**
5: $ParameterSet()$
6: $[NewPC, ExRatio] = Exploration(PC, NPC)$
7: $NPC = EnvironmentalSelection([NPC, NewPC], ExRatio, p)$
8: $NewNPC = Variation(NPC)$
9: $NPC = EnvironmentalSelection([NPC, NewNPC], ExRatio, p)$
10: $PC = PCSelection([PC, NPC, NewNPC])$
11: **end while**

When the algorithm begins to iterate, the individual exploration (Line 6, Algorithm 2) described in Section 4.2.3 is first performed. Exploring whether there are individuals in the PC population that the NPC population has not been (fully) explored. If these individuals existed, it will be stored in set *S* as described above. Then, the new individuals generated using variation operator to *S* was store in the set *T*. Finally, the returned NewPC population is all individuals in the set *T*. The *ExRatio* is a ratio coefficient, which represents the proportion of individuals to be explored. The *ExRatio* is calculated as follows:

$$ExRatio = \frac{Length(S)}{Length(PC)} \tag{12}$$

where $Length(\cdot)$ represents the size of the set or population. When *ExRatio* is greater than 0, it indicates that there are individuals in the PC population that need to be explored. The larger *ExRatio* means the more individuals in PC population need to be explored, and the value range of *ExRatio* is in [0, 1).

The *ExRatio* is set to dynamically change the convergence factor (dynamic convergence factor) of DLS later when using DLS for environment selection. As new individuals are generated by individual exploration, most of these individuals are located in areas that have not been explored or are not fully explored in NPC evolution. Therefore, the exploration at this iteration should pay more attention to these individuals, which means more diversity-related individuals should be appropriately selected to better explore these regions in the evolution of NPC. In this case, the convergence factor is appropriately scaled down according to the size of *ExRatio* at this iteration to achieve this purpose. The detailed process is described in Section 4.3.2.

After individual exploration, the following is the evolution of NPC population (Lines 7–9, Algorithm 2) and PC population (Line 10, Algorithm 2), respectively. First of all, an environment selection is carried out, and the individuals in mixed population of NewPC population and NPC population is selected by using the non-Pareto criterion and stored in NPC population. The variation operator is then applied to the NPC population to generate a new NewNPC population. Then the individuals with better performance in non-Pareto criterion are selected from the mixed population of NPC population and NewNPC population. The evolution of PC population uses PC selection to select non-dominated individuals in mixed population of original PC population, NPC population, and NewNPC population. This will select all non-dominated individuals from the three populations to archive in the PC population. Population maintenance operation is performed on the PC population if necessary (when $Length(\text{PC}) > N$).

4.3.2. Environmental Selection in NPC Evolution

The process of environmental selection in NPC evolution is shown in Algorithm 3. The environmental selection mainly uses DLS to select NPC population. However, dynamic convergence factors α' should be set according to the evolutionary state of the current population before selection. As the number of individuals explored by individual exploration is different at each iteration, the value of *ExRatio* is also different. However, when individuals need to be explored, the convergence factor α should be scaled down. In order to respond to the information of the number of individuals to be explored, the dynamic convergence factor α' is calculated as follows:

$$\alpha\prime = \alpha - \omega * \sin(\frac{ExRatio * \pi}{2}) \tag{13}$$

where ω is a dynamic scaling factor and is set to 0.1. The main purpose of this setting is to prevent the convergence factor from scaling too much, because a good convergence performance can be maintained when the convergence factor is set at 0.9 or so. Since the value interval of *ExRatio* is [0, 1), the value interval of dynamic convergence factor α' is [0.8, 0.9). This allows DLS to play a better role even in individual exploration.

After the predefined parameters are set, the next step is to select the individuals in the candidate population using DLS as shown in Algorithm 1. The population, here, is generated by the BCE process rather than a hybrid population with a parent–child relationship. In addition, the convergence factors used are dynamic convergence factors α' that are scaled according to the state of individual exploration.

Algorithm 3 Environmental Selection in NPC Evolution

Input: The population to be selected: P; Dynamic convergence factor ratio: $ExRatio$; Parameter of p-norm: p
Output: The final population : $NewNPC$

```
//The convergence factor α is set dynamically according to ExRatio
α = 0.9; ω = 0.1
if ExRatio > 0 then
    Calculte α' by Eq. (13)
else
    α' = α
end if
//The population was selected using DLS
NewNPC = DLS(P, α', p)
```

4.3.3. The Time Complexity Analysis of DL-TPCEA

The time complexity of DL-TPCEA is mainly determined by the party that consumes more time during the evolution of PC and NPC. In PC selection, the time complexity of selecting non- dominated individuals from the three-part population (Line 10 of Algorithm 2) is $O(MN^2)$. The time complexity of population maintenance and individual exploration is also $O(MN^2)$. So, the time complexity of PC evolution is $O(MN^2)$. In the NPC evolution, the time complexity of first non-dominated sort is $O(N\log^{M-2}N)$. The time complexity of calculating the number of convergence-related and diversity-related individuals is C (C is a constant), while the time complexity of calculating the two indicators of candidate solutions is $O(MN^2)$ and $O(N^2)$, respectively. The time complexity of using the indicators to select the candidate solution is $O(N)$. In conclusion, the time complexity of DL-TPCEA is $\max\{O(N\log^{M-2}N), O(MN^2)\}$.

5. Experiments

This section will verify the performance of the DL-TPCEA through experiments. First of all, the proposed dynamic convergence factor will be through a number of experiments to get an optimal equation. In addition, this paper will conduct an experimental analysis of the role of individual exploration in the whole evolutionary process. Finally, the performance of the DL-TPCEA is validated against several state-of-the-art algorithms.

5.1. Parameter Setting

In order to give full play to the performance of MaOEAs on MaOPs with different objective number, different *FEs* and population size *N* should be set for different objective number *M*. Taking the WFG [89,90] test suite as an example, the number of dimensions *D* needs to be dynamically changed. Here is set as recommended $D = M + 9$. In addition, the number of objectives in the experiments conducted in this paper is divided into five groups, and the number of objectives is 3, 5, 8, 10, and 15, respectively. In terms of population size setting, since reference points are used in both MOEA/D-PaS and NSGA-III, the original reference points need to be generated in a certain way. In this case, Das and Dennis's approach [91] is used to generate the original reference points on the hyperplane, while the other algorithms should have the same initial population size to ensure fairness. In addition, the number of generated reference points is the same with set in NSGA-III [5,6]. So, the corresponding population size *N* is set to 91, 210, 156, 275, and 135, respectively. The corresponding number of *FEs* is 10^4–$10^4 \times 5$. The detailed parameter settings are shown in Table 1.

Table 1. The parameter settings of experiments.

M	*N*	*D*	*FEs*
3	91	12	10000
5	210	14	20000
8	156	17	30000
10	275	19	40000
15	135	24	50000

In the experiments of dynamic convergence factor, α in the base DLS are set to the recommended 0.9. In the setting of dynamic convergence factors, various functions monotonically increasing in the interval [0, 1] are adopted for dynamic adjustment, which will be described in detail in Section 5.2. For all comparative algorithms in experiments, the parameter settings on each objective were also consistent with those in Table 1.

In addition, the running device is PC, the system version is Windows 10 enterprise version, the processor is Intel(R) Core (TM) i3-8100 CPU 3.6 GHz, and the RAM is 8 GB.

5.2. Experiments on Dynamic Convergence Factors

This paper proposes the concept of dynamic convergence factor in Section 4.3.2. The main approach is to determine the size of convergence factor dynamically based on the basic DLS and the state of individual exploration. The purpose of dynamic convergence factor is to make DLS adapt better to the evolutionary state of the population, so as to achieve the optimal convergence factor setting. Since the optimal value range of the convergence factor α is the interval [0.8, 0.9], we take the value of a monotone increasing function in the interval [0, 1] as shown in Equation (13), multiply it by a dynamic scaling factor ω by the function mapping of proportional coefficient *ExRatio*, and then subtract the corrected value from the original. In this way, it is possible to dynamically change the value of α' according to the proportionality coefficient *ExRatio*. In addition, when the *ExRatio* is relatively large (more diversity-related solutions need to be explored), the convergence factor can be appropriately reduced to better satisfy the convergence and diversity balance of the population.

In order to give full play to the optimal performance of the DL-TPCEA, certain work require to select the monotone increasing function in the interval [0, 1] of Equation (13). Columns 3 through 11 of the first row in Table 2 show some common monotone increasing functions in the interval [0, 1] as a comparative experiment. For example, the corresponding formula of Tan in the fourth column is as follows:

$$\alpha\prime = \alpha - \omega * \tan(\frac{ExRatio * \pi}{4}) \tag{14}$$

Table 2. The results of various monotone increasing functions on the WFG test suite, and the inverted generational distance (IGD) values of the results are tested by Friedman test.

M/Func	Ori	Sin	Tan	x^1	$x^{1/2}$	$x^{1/3}$	$x^{1/M}$	x^2	x^3	x^M	Fix
3	5.56	4.44	6.22	6.22	5.67	6.78	6.78	5.22	7.33	6.67	5.11
5	5.44	5.56	6.44	6.89	7.33	4.22	5.67	6.67	7.22	4.11	6.44
8	7.22	5.22	6.44	5.11	4.72	4.89	5.78	6.50	6.44	8.22	5.44
10	4.78	4.89	5.33	5.89	6.56	6.17	6.28	7.44	6.00	6.67	6.00
15	6.00	5.00	6.33	6.00	6.44	6.33	5.89	5.67	6.00	6.11	6.22
Avg	5.80	5.02	6.16	6.02	6.14	5.68	6.08	6.30	6.60	6.36	5.84

In addition, column 2 corresponds to the original DLS that the value of α is set to 0.9. The last column is a control group, and the method used here is that when $ExRatio > 0$, the value of α is multiplied by a value less than 1 (set as 0.9 here). This is equivalent to setting up a fixed set of transformations instead of making dynamic changes through functional mapping and the proportional coefficient *ExRatio*. This group is designed to analyze and compare the advantages and disadvantages of fixed transformations over functional mappings.

According to the parameter settings in Table 1, the different algorithms were run independently 30 times on each WFG test suite. The average inverted generational distance (IGD) values of the 30 runs were performed using the Friedman test (the smaller the better) and presented in Table 2. The last row is the average of the five sets of Friedman tests. Dark gray represents the best result and light gray represents the second-best result. From the results, the best performance is obtained when the monotone increasing function is taken as the sine function, and the optimal value (minimum value) is obtained on the 3- and 15-objective WFG, respectively. The second-best result is obtained on the 10-objective WFG. Although the results on 5- and 8-objective WFG are not so good, they are also relatively small in terms of numerical values. At the same time, the sinusoidal results were also the best among the average Friedman results of the five experiments. In addition, when the monotone increasing function is $x^{1/3}$, it performs second-best, and obtains second best results on the 5- and 8-objective WFG, respectively, and also obtains second best results in the average Friedman test.

When the monotone increasing function is selected as x^M and $x^{1/2}$, the optimal value is obtained on 5- and 8-objective WFG, respectively. However, the average Friedman test results for these two functions are not very good. It is worth noting that the original DLS obtained the optimal result on 10-objective WFG, followed by the mapping of sine function. In addition, the set of fixed transformations also showed the second-best result on 3-objective WFG. The average Friedman test results of these two strategies are not much different, but they are not as good as the average Friedman test results of the sine function.

Therefore, the sine function mapping of *ExRatio* is finally selected in dynamic convergence factor. In the experiments in Section 5.3 below, the dynamic convergence factor in DL-TPCEA is calculated in the form shown in Equation (13).

5.3. Experiments on Comparative Algorithms

To verify the performance of the proposed DL-TPCEA, we compare it with five state-of-the-art algorithms: MOEA/D-PaS [14], NSGA-III [5], CMOPSO [92], Two_Arch2 [88], and DLEA. The brief introduction to these comparative algorithms is given below.

In MOEA/D-PaS, a Pareto adaptive scalarizing (PaS) approximation method was proposed, which approximated the optimal p value of the commonly used scalarizing method. This is the key to balancing Pareto optimal selection pressure and algorithm robustness to PF geometries. It guarantees that any solution can be found along PF for given some weight. PaS is combined with the decomposition-based algorithm (MOEA/D) to increase the ability of balanced convergence and diversity.

NSGA-III is an improved version based on the framework of NSGA-II [4]. The crowding distance operator that was used to balance diversity in NSGA-II is modified into a diversity keeping strategy based on weight vector guidance. NSGA-III used a set of pre-generated uniformly distributed weight vectors to simulate the distribution of PF. When selecting solutions, the candidate solutions with the shortest vertical distance to these weight vectors will be selected.

CMOPSO is an improved version of the multi-objective particle swarm optimization (MOPSO [93]) by adding a competition mechanism. CMOPSO makes particles pairwise competitions to select particles in each generation of population. This makes the performance of CMOPSO less dependent on global and local optimal particles stored in an external archive.

Two_Arch2 uses two external archives, where each archive promotes convergence (CA) and diversity (DA). The two archives use different selection principles, where CA is indicator-based and DA is Pareto-based. At the same time, L_p-norm-based diversity maintenance scheme was also proposed in Two_Arch2 to improve the diversity of the population.

DLEA mainly uses the DLS mentioned in Section 4.1. The algorithm mainly used DLS to enhance the balance of convergence and diversity in environmental selection. Two different indicators are used to improve the performance by maintaining the convergence and diversity, respectively. Meanwhile, the convergence factor α in DLEA was fixed at 0.9. Compared with DLEA, DL-TPCEA proposes the concept of dynamic convergence factor. The comparison of these two algorithms is mainly to highlight the performance improvement brought by the dynamic convergence factors and the coevolution of the two populations.

The five comparative algorithms selected have their own characteristics, including operators frequently used in the many-objective optimization field: decomposition-based operator, Pareto-based operator, indicator-based operator, external-archive-based operator, and weight-vector-based operator. Comparing these algorithms can show the performance advantage of an algorithm more significantly.

For DL-TPCEA and other five comparative algorithms, Tables 3 and 4 give the mean and standard deviation (in parentheses) of HV values run on five sets of WFG test suites, and the results in Tables 5 and 6 are corresponding IGD values. Wilkerson Rank-Sum test ($\alpha = 0.05$) was used to test the significant difference between HV values and IGD values of these six algorithms. The symbols $-$, $+$, and $\approx$ stand for that the indicator values of the comparative algorithms were significantly worse than, better than, and similar to that of DL-TPCEA, respectively. In addition, for each test instance, the best (maximum) HV value and the best (minimum) IGD value were highlighted in gray.

Table 3. HV results of six algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.7315e+1 (6.80e−1) −	2.0767e+1 (1.14e+0) −	1.1761e+1 (1.64e+0) −	1.9660e+1 (9.47e−1) −	2.4915e+1 (1.85e+0) −	2.7839e+1 (1.96e+0)
	5	7.8150e+2 (4.47e+2) −	2.8379e+3 (2.59e+2) ≈	1.6437e+3 (4.05e+1) −	2.2898e+3 (1.67e+2) −	2.8062e+3 (3.01e+2) ≈	2.6535e+3 (2.70e+2)
	8	5.4564e+6 (3.92e+5) −	6.5768e+6 (5.24e+5) ≈	5.2121e+6 (1.31e+5) −	6.4175e+6 (4.79e+5) ≈	8.2665e+6 (6.57e+5) +	6.9852e+6 (1.19e+6)
WFG2	3	5.7446e+1 (3.91e−1) −	5.7559e+1 (3.20e−1) −	5.8959e+1 (1.26e−1) ≈	5.7029e+1 (6.97e−1) −	5.8160e+1 (2.90e−1) −	5.8855e+1 (2.83e−1)
	5	5.2927e+3 (3.37e+2) −	6.0271e+3 (3.39e+1) −	6.0413e+3 (3.69e+1) −	5.9674e+3 (4.52e+1) −	5.9873e+3 (1.95e+1) −	6.1122e+3 (1.83e+1)
	8	1.0172e+7 (8.73e+6) ≈	2.1759e+7 (2.38e+5) ≈	2.1181e+7 (1.19e+5) −	2.1522e+7 (1.66e+5) −	2.1645e+7 (6.42e+4) −	2.1889e+7 (8.02e+4)
WFG3	3	6.0818e+0 (1.00e−1) +	5.5451e+0 (1.47e−1) −	5.7894e+0 (8.46e−2) −	5.5665e+0 (1.38e−1) −	6.1565e+0 (5.93e−2) +	5.9102e+0 (9.30e−2)
	5	1.1540e+0 (6.16e−2) ≈	1.2770e+0 (3.47e−1) ≈	4.1522e−1 (2.29e−1) −	1.2222e+0 (2.80e−1) ≈	1.8798e+0 (1.53e−1) +	1.1142e+0 (3.37e−1)
	8	1.4804e−2 (1.05e−4) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
WFG4	3	3.1857e+1 (3.32e−1) −	3.3687e+1 (2.16e−1) −	3.2170e+1 (1.79e−1) −	3.3801e+1 (2.25e−1) −	3.4028e+1 (2.89e−1) −	3.5106e+1 (1.47e−1)
	5	2.7725e+3 (2.50e+2) −	4.6140e+3 (2.49e+1) −	4.1442e+3 (4.18e+1) −	4.3112e+3 (3.61e+1) −	4.2991e+3 (5.25e+1) −	4.8905e+3 (2.30e+1)
	8	1.7243e+7 (2.76e+5) −	1.8176e+7 (2.42e+5) −	1.6073e+7 (5.95e+5) −	1.5929e+7 (5.49e+4) −	1.7267e+7 (1.02e+5) −	2.0259e+7 (1.20e+5)
WFG5	3	3.2283e+1 (1.20e−1) −	3.2008e+1 (1.51e−1) −	3.2193e+1 (3.26e−1) −	3.1499e+1 (2.29e−1) −	3.2310e+1 (1.84e−1) −	3.3072e+1 (2.45e−1)
	5	2.5150e+3 (2.93e+2) −	4.4522e+3 (2.15e+1) −	4.0531e+3 (1.17e+2) −	4.0890e+3 (3.42e+1) −	4.0815e+3 (5.67e+1) −	4.5798e+3 (1.63e+1)
	8	1.6273e+7 (2.15e+5) −	1.7208e+7 (1.66e+5) −	1.4956e+7 (9.07e+5) −	1.4988e+7 (1.39e+5) −	1.6070e+7 (3.91e+5) −	1.8993e+7 (7.80e+4)
WFG6	3	2.8635e+1 (2.28e+0) −	3.0041e+1 (7.45e−1) −	3.3213e+1 (3.90e−1) +	2.9430e+1 (7.41e−1) −	3.0771e+1 (7.68e−1) −	3.1280e+1 (8.81e−1)
	5	3.4025e+3 (3.89e+2) −	4.2268e+3 (1.19e+2) −	3.9875e+3 (1.69e+2) −	3.8312e+3 (9.55e+1) −	3.8672e+3 (1.54e+2) −	4.3856e+3 (8.73e+1)
	8	1.6134e+7 (5.09e+5) −	1.6452e+7 (7.46e+5) −	1.3440e+7 (5.23e+5) −	1.3529e+7 (2.04e+5) −	1.5082e+7 (3.53e+5) −	1.8534e+7 (4.34e+5)
WFG7	3	3.3709e+1 (3.69e−1) −	3.4017e+1 (2.20e−1) −	3.4256e+1 (1.77e−1) −	3.4185e+1 (2.61e−1) −	3.4551e+1 (2.13e−1) −	3.5763e+1 (9.24e−2)
	5	2.4688e+3 (3.46e+2) −	4.6163e+3 (3.97e+1) −	4.0731e+3 (8.89e+1) −	4.4358e+3 (4.30e+1) −	4.3903e+3 (4.88e+1) −	4.9534e+3 (8.29e+0)
	8	1.4711e+7 (3.57e+6) −	1.7534e+7 (3.70e+5) −	1.5266e+7 (4.98e+5) −	1.5848e+7 (1.99e+5) −	1.7622e+7 (2.71e+4) −	2.0675e+7 (3.51e+4)
WFG8	3	2.7295e+1 (4.85e−1) −	2.8098e+1 (3.27e−1) −	2.7821e+1 (3.24e−1) −	2.7825e+1 (3.80e−1) −	2.8807e+1 (2.59e−1) −	2.9372e+1 (2.48e−1)
	5	1.7367e+3 (1.64e+2) −	3.8951e+3 (3.91e+1) +	3.1389e+3 (7.11e+1) −	3.5199e+3 (5.54e+1) −	3.2468e+3 (7.93e+1) −	3.8241e+3 (4.35e+1)
	8	5.7959e+6 (4.70e+6) −	1.4732e+7 (2.88e+5) −	1.1287e+7 (2.78e+5) −	1.1568e+7 (1.40e+5) −	1.1853e+7 (3.02e+5) −	1.6523e+7 (2.10e+5)
WFG9	3	3.1191e+1 (2.40e+0) −	3.2676e+1 (4.33e−1) −	3.3517e+1 (2.19e−1) −	3.2831e+1 (5.47e−1) −	3.3327e+1 (3.12e−1) −	3.4084e+1 (2.54e−1)
	5	2.3419e+3 (4.40e+2) −	4.1754e+3 (1.73e+2) −	4.1393e+3 (2.07e+2) −	4.2066e+3 (6.28e+1) −	4.1163e+3 (6.78e+1) −	4.6044e+3 (4.08e+1)
	8	1.0250e+7 (4.79e+6) −	1.5390e+7 (7.88e+5) −	1.5401e+7 (4.42e+5) −	1.5036e+7 (3.68e+5) −	1.5800e+7 (3.83e+5) −	1.8621e+7 (1.34e+5)
	-	23	21	24	24	22	
	+	2	1	1	0	3	
	≈	2	5	2	3	2	

Table 4. HV results of six algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	10	2.4932e+9 (2.20e+8) −	2.9818e+9 (2.95e+8) ≈	2.0618e+9 (3.09e+7) −	2.8738e+9 (2.57e+8) ≈	4.3759e+9 (3.33e+8) +	3.0894e+9 (9.09e+7)
	15	2.8183e+16 (1.31e+15) −	6.2299e+16 (1.10e+16) +	2.8367e+16 (4.81e+14) −	3.6933e+16 (4.35e+15) ≈	7.6448e+16 (6.95e+15) +	3.9888e+16 (6.02e+15)
WFG2	10	3.1504e+9 (1.82e+9) −	9.5411e+9 (6.93e+7) ≈	9.1557e+9 (2.83e+7) −	9.4642e+9 (5.10e+7) −	9.4695e+9 (2.51e+7) −	9.5531e+9 (2.77e+7)
	15	3.5854e+16 (4.24e+16) −	1.7728e+17 (2.08e+14) −	1.5854e+17 (5.06e+15) −	1.7519e+17 (8.15e+14) −	1.7686e+17 (3.04e+14) −	1.7777e+17 (1.70e+14)
WFG3	10	4.9521e−5 (1.99e−7) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
	15	6.1526e−16 (1.36e−16) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
WFG4	10	3.6386e+9 (3.33e+9) −	8.5986e+9 (9.17e+7) −	6.6569e+10 (8.20e+10) ≈	6.8836e+9 (1.41e+8) −	7.5439e+9 (3.98e+7) −	9.1189e+9 (4.33e+7)
	15	2.9135e+16 (8.76e+15) −	1.5641e+17 (3.18e+15) −	1.1607e+17 (6.92e+15) −	1.2252e+17 (3.15e+15) −	1.4496e+17 (3.45e+15) −	1.7439e+17 (6.93e+14)
WFG5	10	8.0298e+8 (3.32e+7) −	8.2007e+9 (5.81e+7) −	6.4771e+9 (3.09e+8) −	6.3213e+9 (1.22e+8) −	7.1212e+9 (1.27e+8) −	8.5253e+9 (2.97e+7)
	15	1.4143e+16 (4.65e+7) −	1.4816e+17 (2.20e+15) −	1.1418e+17 (3.98e+15) −	1.0054e+17 (2.06e+15) −	1.2722e+17 (4.56e+15) −	1.6240e+17 (3.51e+14)
WFG6	10	2.2633e+9 (1.39e+9) −	7.7778e+9 (1.07e+8) −	5.8760e+9 (2.79e+8) −	6.0002e+9 (1.79e+8) −	6.6823e+9 (1.15e+8) −	8.1978e+9 (2.35e+8)
	15	2.0445e+16 (6.61e+15) −	1.4394e+17 (6.05e+15) −	1.0594e+17 (5.08e+15) −	9.3182e+16 (4.88e+15) −	1.2253e+17 (4.59e+15) −	1.5706e+17 (2.76e+15)
WFG7	10	1.8488e+9 (1.04e+9) −	8.7612e+9 (1.81e+8) −	6.6536e+9 (1.85e+8) −	6.7900e+9 (1.34e+8) −	7.7372e+9 (8.78e+7) −	9.2994e+9 (7.01e+6)
	15	1.6211e+16 (1.17e+14) −	1.5763e+17 (5.67e+15) −	1.2395e+17 (2.42e+15) −	1.0505e+17 (4.37e+15) −	1.4760e+17 (1.91e+15) −	1.7724e+17 (1.34e+14)
WFG8	10	8.6874e+8 (7.58e+6) −	7.5655e+9 (9.99e+7) −	4.4228e+9 (1.09e+8) −	4.8222e+9 (2.53e+8) −	5.3348e+9 (1.81e+8) −	7.8199e+9 (1.26e+8)
	15	1.7958e+16 (2.63e+15) −	1.1199e+17 (1.40e+16) −	6.6492e+16 (1.16e+16) −	7.0410e+16 (1.53e+15) −	1.0321e+17 (4.76e+15) −	1.5825e+17 (3.19e+15)
WFG9	10	1.0134e+9 (7.09e+8) −	7.7585e+9 (2.95e+8) −	6.3588e+9 (2.16e+8) −	6.6188e+9 (1.90e+8) −	6.7972e+9 (1.06e+8) −	8.4373e+9 (7.94e+7)
	15	1.2144e+16 (2.13e+15) −	1.4748e+17 (6.79e+15) ≈	1.1051e+17 (5.64e+15) −	1.0634e+17 (3.16e+15) −	1.2065e+17 (1.51e+15) −	1.5617e+17 (1.04e+16)
	−	16	12	15	14	14	
	+	2	1	0	0	2	
	≈	0	5	3	2	2	

Table 5. IGD results of six algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.5489e+0 (2.29e−2) −	1.3950e+0 (5.27e−2) −	1.7560e+0 (7.19e−2) −	1.4184e+0 (4.49e−2) −	1.2451e+0 (8.23e−2) −	1.1443e+0 (7.65e−2)
	5	2.7526e+0 (2.30e−1) −	1.4514e+0 (1.49e−1) ≈	2.2035e+0 (5.35e−2) −	1.7185e+0 (8.59e−2) −	1.4791e+0 (1.27e−1) ≈	1.5296e+0 (1.11e−1)
	8	3.0437e+0 (1.83e−1) −	2.3897e+0 (1.09e−1) ≈	2.8601e+0 (4.19e−2) −	2.3930e+0 (8.43e−2) ≈	2.0988e+0 (1.13e−1) ≈	2.2369e+0 (2.02e−1)
WFG2	3	2.3806e−1 (9.89e−3) −	1.7972e−1 (3.04e−2) −	1.1896e−1 (6.05e−3) −	1.7195e−1 (1.48e−2) −	2.5026e−1 (3.96e−2) −	1.1400e−1 (8.60e−3)
	5	1.7359e+0 (1.66e−1) −	7.1980e−1 (5.74e−2) −	6.0869e−1 (7.43e−2) −	6.2619e−1 (7.94e−2) −	1.3129e+0 (4.11e−1) −	5.1156e−1 (6.59e−2)
	8	6.3124e+0 (3.79e+0) −	3.1921e+0 (1.86e+0) −	1.1504e+0 (4.51e−2) −	1.1816e+0 (1.89e−1) ≈	3.3268e+0 (2.23e−1) −	1.0066e+0 (2.18e−1)
WFG3	3	1.0249e−1 (1.37e−2) ≈	1.6696e−1 (1.62e−2) −	1.2856e−1 (1.03e−2) −	1.5342e−1 (1.84e−2) −	8.6193e−2 (7.43e−3) +	1.1065e−1 (1.13e−2)
	5	1.3530e+0 (1.41e−1) −	5.4731e−1 (1.05e−1) ≈	7.7094e−1 (8.68e−2) −	4.1412e−1 (3.98e−2) +	3.6988e−1 (3.44e−2) +	5.6591e−1 (6.60e−2)
	8	8.8861e+0 (8.66e−5) −	9.8482e−1 (3.15e−1) +	1.5545e+0 (8.06e−2) ≈	9.4384e−1 (3.33e−2) +	7.3548e−1 (8.64e−2) +	1.6935e+0 (3.31e−1)
WFG4	3	2.6154e−1 (9.71e−3) −	1.8267e−1 (3.22e−3) −	2.1399e−1 (4.38e−3) −	1.9309e−1 (5.21e−3) −	2.2205e−1 (7.93e−3) −	1.7016e−1 (3.78e−3)
	5	2.6627e+0 (1.28e−1) −	1.1634e+0 (2.87e−3) −	9.9623e−1 (1.00e−2) −	1.0020e+0 (9.97e−3) −	1.0731e+0 (2.04e−2) −	9.7239e−1 (9.44e−3)
	8	3.4398e+0 (4.68e−2) −	2.7417e+0 (1.38e−2) −	2.6563e+0 (5.37e−2) −	2.8326e+0 (1.04e−2) −	2.7079e+0 (1.13e−2) −	2.5307e+0 (1.51e−2)
WFG5	3	2.1185e−1 (2.89e−3) −	1.9548e−1 (2.95e−3) −	1.8860e−1 (6.74e−3) ≈	2.1768e−1 (4.96e−3) −	2.3407e−1 (8.23e−3) −	1.8664e−1 (3.07e−3)
	5	2.7787e+0 (1.90e−1) −	1.1383e+0 (5.93e−3) −	9.8193e−1 (2.86e−2) +	9.8959e−1 (1.00e−2) ≈	1.0962e+0 (1.72e−2) −	9.9293e−1 (9.04e−3)
	8	3.3155e+0 (3.27e−2) −	2.7859e+0 (1.21e−2) −	2.8877e+0 (8.40e−2) −	2.8167e+0 (2.57e−2) −	2.8605e+0 (4.45e−2) −	2.6346e+0 (3.32e−2)
WFG6	3	3.0029e−1 (3.64e−2) −	2.5494e−1 (1.76e−2) −	1.9560e−1 (7.25e−3) +	2.8551e−1 (1.74e−2) −	2.7869e−1 (1.58e−2) −	2.4262e−1 (1.89e−2)
	5	1.8877e+0 (3.93e−1) −	1.1613e+0 (3.45e−3) −	1.0374e+0 (3.02e−2) ≈	1.0509e+0 (1.60e−2) ≈	1.1386e+0 (2.54e−2) −	1.0488e+0 (1.30e−2)
	8	3.2864e+0 (6.98e−2) −	2.8195e+0 (3.23e−2) ≈	2.9362e+0 (2.23e−2) −	2.9558e+0 (4.11e−2) −	2.9154e+0 (3.73e−2) −	2.8073e+0 (4.48e−2)
WFG7	3	2.2119e−1 (8.24e−3) −	1.8009e−1 (3.35e−3) ≈	1.7571e−1 (3.71e−3) ≈	1.9406e−1 (9.73e−3) −	2.2983e−1 (1.11e−2) −	1.7794e−1 (5.79e−3)
	5	2.9302e+0 (2.47e−1) −	1.1637e+0 (4.06e−3) −	1.0208e+0 (1.51e−2) +	9.8399e−1 (8.08e−3) +	1.1237e+0 (3.00e−2) −	1.0808e+0 (2.16e−2)
	8	4.7379e+0 (3.07e+0) −	2.7927e+0 (1.71e−2) −	2.6952e+0 (2.84e−2) +	2.8153e+0 (3.58e−2) −	2.8150e+0 (3.02e−2) −	2.7405e+0 (2.75e−2)
WFG8	3	3.3340e−1 (1.38e−2) −	3.0618e−1 (6.91e−3) −	3.1422e−1 (8.61e−3) −	3.2821e−1 (1.06e−2) −	3.2300e−1 (9.59e−3) −	2.9003e−1 (5.52e−3)
	5	2.9656e+0 (1.37e−1) −	1.1706e+0 (1.15e−2) −	1.2719e+0 (3.13e−2) −	1.1570e+0 (1.48e−2) −	1.2426e+0 (2.15e−2) −	1.1148e+0 (1.25e−2)
	8	1.1416e+1 (3.81e+0) −	3.1538e+0 (1.14e−2) −	3.3418e+0 (5.20e−2) −	3.4145e+0 (2.80e−2) −	3.2563e+0 (5.28e−2) −	2.9559e+0 (4.04e−2)
WFG9	3	2.5049e−1 (4.26e−2) −	1.8430e−1 (9.41e−3) −	1.6713e−1 (3.72e−3) −	1.8655e−1 (1.19e−2) −	2.0949e−1 (7.53e−3) −	1.6080e−1 (3.23e−3)
	5	2.5243e+0 (1.81e−1) −	1.1054e+0 (1.22e−2) −	9.8803e−1 (3.72e−2) −	9.7439e−1 (1.19e−2) −	1.0773e+0 (2.72e−2) −	9.5847e−1 (1.16e−2)
	8	6.4301e+0 (4.24e+0) −	2.8681e+0 (5.07e−2) −	2.9972e+0 (2.68e−2) −	2.9064e+0 (2.25e−2) −	2.9369e+0 (5.78e−2) −	2.6905e+0 (4.47e−2)
	-	26	21	19	20	22	
	+	0	1	4	3	3	
	≈	1	5	4	4	2	

Table 6. IGD results of six algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	10	3.4752e+0 (2.18e−1) −	2.5247e+0 (1.47e−1) ≈	3.1837e+0 (3.94e−2) −	2.5480e+0 (8.68e−2) ≈	2.0295e+0 (1.02e−1) +	2.3727e+0 (1.48e−1)
	15	4.4269e+0 (1.80e−1) −	2.8567e+0 (2.36e−1) +	3.9681e+0 (6.38e−2) −	3.5058e+0 (1.05e−1) ≈	2.4277e+0 (1.43e−1) +	3.2972e+0 (2.62e−1)
WFG2	10	8.0643e+0 (1.55e+0) −	5.5784e+0 (3.05e+0) −	1.6248e+0 (1.81e−1) ≈	2.4826e+0 (3.88e−1) −	5.7470e+0 (1.74e+0) −	1.7920e+0 (3.61e−1)
	15	1.7357e+1 (5.54e+0) −	1.1222e+1 (1.89e+0) −	1.5628e+0 (3.78e−1) ≈	1.2642e+0 (2.13e−1) +	1.2789e+1 (1.38e+0) −	2.3050e+0 (7.98e−1)
WFG3	10	1.1189e+1 (1.28e−4) −	9.1908e−1 (6.98e−2) +	2.1021e+0 (1.88e−1) +	1.1879e+0 (8.56e−2) +	1.1060e+0 (1.36e−1) +	2.4613e+0 (2.26e−1)
	15	1.6953e+1 (3.81e−5) −	2.2492e+0 (1.82e−1) +	4.0070e+0 (3.24e−1) ≈	2.1674e+0 (2.62e−1) +	1.6673e+0 (3.86e−1) +	4.2693e+0 (1.23e+0)
WFG4	10	1.3196e+1 (6.43e+0) −	4.4372e+0 (8.31e−2) −	4.7969e+0 (9.82e−1) ≈	4.2424e+0 (5.27e−2) ≈	4.1316e+0 (4.22e−2) ≈	4.2136e+0 (1.04e−1)
	15	2.8206e+1 (9.78e−1) −	8.3714e+0 (4.08e−1) −	7.6517e+0 (8.31e−2) ≈	8.1285e+0 (9.26e−2) −	7.4599e+0 (5.49e−2) +	7.7824e+0 (1.11e−1)
WFG5	10	1.8727e+1 (1.34e−1) −	4.4440e+0 (3.06e−2) −	4.1452e+0 (7.06e−2) −	4.1626e+0 (2.03e−2) −	4.2241e+0 (3.41e−2) −	4.0344e+0 (3.86e−2)
	15	3.0024e+1 (6.03e−8) −	7.8352e+0 (2.05e−1) ≈	7.4354e+0 (4.90e−2) ≈	7.7203e+0 (4.80e−2) −	7.1981e+0 (1.28e−1) +	7.5313e+0 (6.01e−2)
WFG6	10	1.4944e+1 (4.03e+0) −	4.5720e+0 (2.87e−2) ≈	4.2313e+0 (3.21e−2) ≈	4.2511e+0 (3.52e−2) ≈	4.4157e+0 (6.37e−2) ≈	4.3695e+0 (2.07e−1)
	15	2.8561e+1 (2.44e+0) −	1.0013e+1 (1.75e+0) ≈	7.4634e+0 (1.60e−1) +	7.7827e+0 (7.71e−2) ≈	7.3120e+0 (1.30e−1) +	8.0243e+0 (1.90e−1)
WFG7	10	1.6823e+1 (1.73e+0) −	4.5121e+0 (1.30e−1) −	4.0027e+0 (2.17e−2) +	4.1460e+0 (1.77e−2) ≈	4.2551e+0 (5.57e−2) ≈	4.1518e+0 (6.56e−2)
	15	3.0346e+1 (1.61e−3) −	8.8360e+0 (3.39e−1) −	7.2612e+0 (7.48e−2) +	8.2064e+0 (1.16e−1) −	7.1798e+0 (1.01e−1) +	7.6130e+0 (8.70e−2)
WFG8	10	1.9093e+1 (1.36e−1) −	5.1188e+0 (5.07e−1) −	4.6629e+0 (2.17e−2) −	4.8276e+0 (3.38e−2) −	4.6012e+0 (3.56e−2) −	4.2410e+0 (4.91e−2)
	15	2.8903e+1 (1.53e+0) −	9.5721e+0 (7.98e−1) −	7.9078e+0 (6.22e−2) −	8.6835e+0 (1.83e−1) −	7.7107e+0 (6.01e−2) −	7.3664e+0 (2.52e−1)
WFG9	10	1.7540e+1 (2.19e+0) −	4.1996e+0 (7.49e−2) ≈	4.3883e+0 (3.91e−2) −	4.2512e+0 (3.14e−2) −	4.3247e+0 (2.94e−2) −	4.1677e+0 (2.88e−2)
	15	2.9935e+1 (1.05e−1) −	8.1123e+0 (2.73e−1) −	7.7758e+0 (5.19e−2) −	8.0385e+0 (1.01e−1) −	7.7769e+0 (1.58e−1) −	7.2405e+0 (1.43e−1)
	−	18	10	7	9	7	
	+	0	3	4	3	8	
	≈	0	5	7	6	3	

As shown in Tables 3 and 4, in terms of HV, the proposed DL-TPCEA was significantly better than the other five algorithms on 26 out of 45 test instances, and performed similarly to them on two test instances. Specifically, DL-TPCEA generated higher HV values than MOEA/D-PaS, NSGA-III, CMOPSO, Two_Arch2, and DLEA on 39, 33, 39, 38, and 36 out of the 45 test instances, respectively. As shown in Tables 5 and 6, in terms of IGD, the proposed DL-TPCEA was significantly better than the other five algorithms on 21 out of 45 test instances, and performed similarly to them on one test instance. DL-TPCEA generated smaller IGD values than MOEA/D-PaS, NSGA-III, CMOPSO, Two_Arch2, and DLEA on 44, 31, 26, 29, and 29 out of the 45 test instances, respectively. The results demonstrated that it was a promising way to approximate the PFs of WFGs via coevolution and dynamic learning strategy in the proposed DL-TPCEA. CMOPSO and DLEA showed better results for IGD values than for HV values, indicating that these two algorithms also had a good ability to maintain the trade-off between convergence and diversity. In addition, from the comprehensive results of HV values and IGD values, NSGA-III was also a good algorithm. However, compared with these three MaOEAs, the proposed DL-TPCEA also showed much better performance with respect to both convergence and diversity.

The superiority of DL-TPCEA can be explained as follows. The other five comparative algorithms, with the exception of Two_Ach2, attempted to simulate PFs of WFGs through balanced convergence and diversity using a single population. However, as the number of objectives increases, the balance between convergence and diversity became more difficult. This was because the increasing number of objectives led to more serious conflicts on multiple objectives, so that the selection pressure of the MOEAs was not as good as when there were fewer objectives. When the number of objectives kept increasing, the solutions generated by these algorithms may only be single convergence-related solutions or diversity-related solutions, but there was no compromise between convergence and diversity over the whole PF. In addition, Two_Arch2 used two different external archives to store convergence-related solutions or diversity-related solutions, respectively. Moreover, Two_Arch2 promoted the evolution between the two archives so that the population maintained a compromise between convergence and diversity. However, these two external archiving methods had poor performance in dealing with MaOPs, especially the objective conflicts were serious. The proposed DL-TPCEA used two populations for coevolution and the shortcomings of each population will be compensated by BCE. It kept a good balance between convergence and diversity, and used dynamic learning strategy to further strengthen the balance. As a result, the proposed DL-TPCEA did not degrade the performance because of the conflicts caused by the number of objectives increased.

From the HV values in Tables 3 and 4, we can see that the HV value obtained by other related algorithms except MOEA/D-PaS on 8-, 10-, and 15-objective WFG3 was zero. This was caused by the calculation of the HV results using a set of reference points set on the corresponding test instance. When the corresponding algorithms failed to obtain any candidate solution dominating the reference point on those test instances, the value of the hypervolume (HV value) formed by the non-dominated population and the reference point was zero. In these three test instances, only MOEA/D-PaS obtained HV results, which also indicates that MOEA/D-PaS had its own advantages in dealing with WFG3. In addition to the three test instances, all the algorithms obtained HV values (non-zero) on the other test instances.

It can also be concluded from the results that DL-TPCEA mainly showed poor performance on WFG1 and WFG3. In terms of the characteristics of the problem, WFG1 is convex and mixed, while WFG3 is linear and degenerate. DL-TPCEA may not be able to find boundary individuals well on such problems like WFG1 or WFG3, thus, resulting in poor performance. In addition, WFG2 is convex and disconnected. However, DL-TPCEA was still able to obtain the optimal HV results, indicating that the two-population coevolution of DL-TPCEA is capable of dealing with disconnected MaOPs. Finally, WFG4-9 is concave, and DL-TPCEA also has the best performance. The performance of DL-TPCEA on 10- and

15-objective WFG4-9 was lower than DLEA, indicating that dynamic learning strategies played a significant role in dealing with concave MaOPs.

In order to more intuitively observe the ability of the six algorithms to balance convergence and diversity on the WFG test suite, the parallel coordinates of the solution set obtained by the six algorithms on the 5-objective WFG2 and 10-objective WFG9 were given in Figures 3 and 4, respectively. In parallel coordinates, the ordinate represents the objective value, and the convergence information can be obtained. An algorithm has good convergence if it can converge to the range of PF. At the same time, the vertical height can also reflect the performance in the diversity. The horizontal axis corresponds to each objective, which can reflect the diversity information of MaOEAs. It is an algorithm that maintains solutions for every objective, and the denser the lines, the better the diversity. Therefore, using the parallel coordinates of the solution set can better compare the performance of MaOEAs.

For 5-objective WFG2, the range of PF on each objective dimension m is from 0 to $m * 2$ ($m = 1, \ldots, M$). As shown in Figure 3, although the solution sets obtained by all the six algorithms can successfully converged to the range of the corresponding objective dimension on PF, their diversity was significantly different. Among the six algorithms, MOEA/D-PaS and DLEA had the worst performance in diversity. MOEA/D-PaS had a poor diversity in the second objective, while DLEA had a poor diversity in the second to fourth objectives. By contrast, NSGA-III, CMOPSO, and Two_Arch2 algorithms performed better in diversity. However, the diversity of the solution sets obtained by these three algorithms was not as good as that of DL-TPCEA. It can be seen from in Figure 3f that the diversity of the solution set obtained by DL-TPCEA was good in each objective dimension. This indicated that the output solution set of the proposed DL-TPCEA was better than the other five comparative algorithms in terms of convergence and diversity. This result was also consistent with the maximum HV value and minimum IGD value of DL-TPCEA on 5-objective WFG2, as shown in Tables 3 and 5.

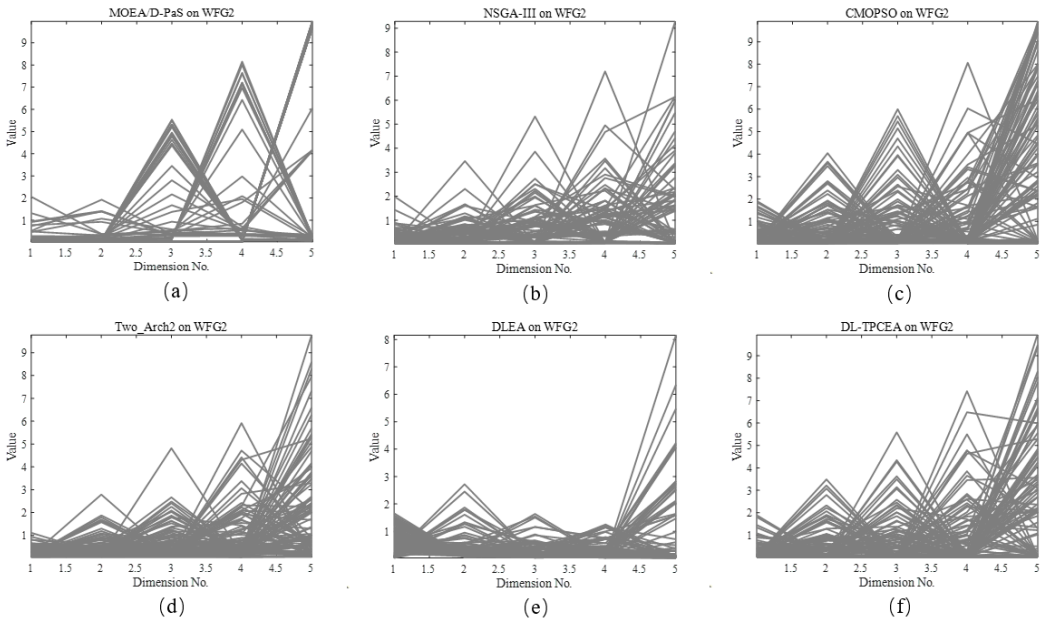

Figure 3. The final solution set obtained by the six MOEAs on 5-objectives WFG2, shown by parallel coordinates. (**a**) The final solution set obtained by MOEA/D-Pas; (**b**) The final solution set obtained by NSGA-III; (**c**) The final solution set obtained by CMOPSO; (**d**) The final solution set obtained by Two_Arch2; (**e**) The final solution set obtained by DLEA; (**f**) The final solution set obtained by DL-TPCEA.

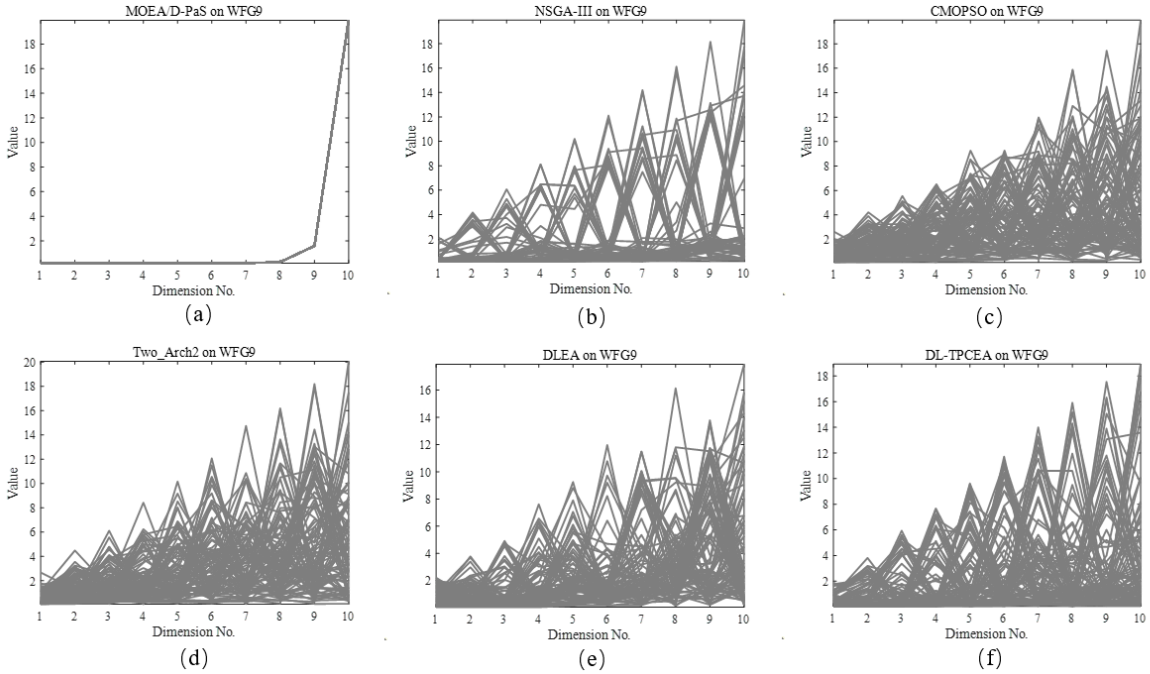

Figure 4. The final solution set obtained by the six MOEAs on 10-objectives WFG9, shown by parallel coordinates. (**a**) The final solution set obtained by MOEA/D-Pas; (**b**) The final solution set obtained by NSGA-III; (**c**) The final solution set obtained by CMOPSO; (**d**) The final solution set obtained by Two_Arch2; (**e**) The final solution set obtained by DLEA; (**f**) The final solution set obtained by DL-TPCEA.

As shown in Figure 4, the solution set obtained by DL-TPCEA was superior to the five comparative algorithms in terms of convergence and diversity. For 10-objective WFG9, the range of PF on each objective dimension m is from 0 to $m * 2$ ($m = 1, \ldots, M$). Among the six algorithms, MOEA/D-PaS converged to few solutions on PF of 10-objective WFG9, so it cannot approach PF well. As shown in Figure 4c,e, the solution set obtained by CMOPSO had a relatively poor diversity on the sixth objective, while DLEA had a relatively poor diversity on the seventh and ninth objectives. NSGA-III and Two_Arch2 algorithms performed well, second only to the convergence and diversity of the solution set obtained by DL-TPCEA on 10-objective WFG9. This was consistent with the HV values and IGD values in Tables 4 and 6.

Figure 5 showed the IGD value trajectories obtained by running six algorithms on the 5-objective WFG test suite. The algorithm for each trajectory was identified in the bottom legend, and DL-TPCEA was specifically highlighted in red. Each subgraph was marked with a different problem, and its abscissa was the number of evaluations during algorithm iteration, and its ordinate was the IGD value. As can be seen from Figure 5, the IGD value trajectory of DL-TPCEA generally declines fastest (except for WFG1 and WFG3), which indicated that DLEA converged very quickly. In addition, from the final result, the IGD value obtained by DL-TPCEA is usually the minimum or not far from the minimum. On 5-objective WFG1, the final IGD value obtained by DL-TPCEA was second only to DLEA. And DLEA obtained the best IGD values on WFG3, while DL-TPCEA performed only at the mid-range level on this issue. Except for WFG1 and WFG3, DL-TPCEA performed very well on the other seven 5-objective WFGs. In general, DL-TPCEA had the best performance among the six algorithms.

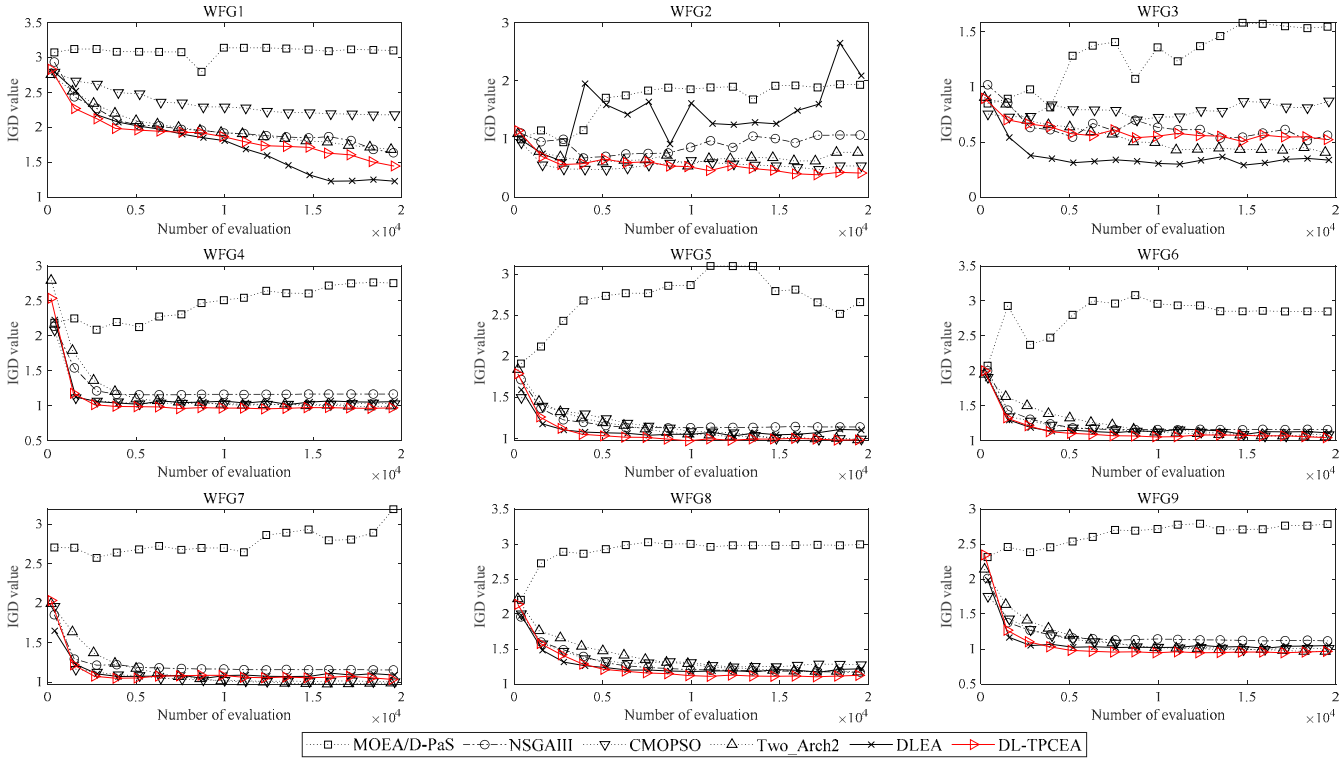

Figure 5. The IGD values convergence trajectories obtained by the six MOEAs on 5-objectives WFG test suite.

Combined with results of HV values and IGD values in Tables 3–6, convergence and diversity effects of solution sets in Figures 3 and 4, as well as IGD value trajectory shown in Figure 5, DL-TPCEA had the best performance in these six algorithms. DL-TPCEA had great advantages in many-objective optimization, both in terms of the convergence and diversity of the final solution set and the convergence speed.

Table 7 shows the average running time of the six comparative algorithms on 3-, 5-, 8-, 10-, and 15-objective WFG1. The last one is the results of Wilcoxon test (the smaller the value is, the shorter the corresponding running time is). The shortest time is NSGA-III, which is due to the simple structure. However, DL-TPCEA is only ranked fourth, which is also a shortcoming. However, given the performance gains, it is worth it, especially for problems that require a lot of accuracy.

Table 7. Comparison of runtime (s) among the six algorithms on 3-, 5-, 8-, 10-, and 15-objective WFG1.

Problem	M	MOEA/D-PaS	NSGA-III	CMOPSO	Two_Arch2	DLEA	DL-TPCEA
WFG1	3	1.7368e+1 (1.75e−1)	6.5904e−1 (3.59e−2)	3.0853e+0 (5.20e−1)	1.1769e+1 (4.49e−1)	1.2042e+1 (8.84e−2)	5.0030e+0 (2.58e−1)
	5	3.5334e+1 (3.27e−1)	1.5763e+0 (6.25e−2)	1.7703e+1 (2.45e+0)	3.8577e+1 (1.06e+0)	2.5697e+1 (1.80e−1)	1.9692e+1 (1.46e+0)
	8	7.2000e+1 (3.02e−1)	2.9553e+0 (5.70e−2)	3.3330e+1 (5.32e+0)	1.0961e+2 (2.00e+0)	6.1095e+1 (5.83e−1)	7.5986e+1 (3.63e+0)
	10	6.7838e+1 (6.60e−1)	3.1321e+0 (2.52e−1)	3.0180e+1 (7.89e+0)	1.1603e+2 (2.26e+0)	5.8485e+1 (2.66e−1)	8.3710e+1 (3.01e+0)
	15	1.0624e+2 (4.48e+0)	4.4098e+0 (1.35e−1)	1.2978e+2 (2.33e+1)	2.4883e+2 (4.10e+0)	1.0903e+2 (7.38e−1)	2.2197e+2 (6.09e+0)
	rank	4.20	1.00	2.40	5.60	3.60	4.20

5.4. Comparison Experiments of DL-TPCEA and Two Weight-Sum Based Algorithms

In this section, we compared DL-TPCEA with two weight-sum based algorithms. The weighted sum method is characterized by fast running speed, simple structure, and easy operation. MaOPs in industrial production tend to have more complex PF, so it may be very limited to solve such problems only by weighted sum method. For this purpose, DL-TPCEA is compared with the weight-sum based approach to verify the advantages of the proposed algorithm.

This paper provides two weight-sum based algorithms for comparison. The first algorithm is a modification of the classic NSGA-II framework, which is called WSEA. The

environment selection of the WSEA starts with a non-dominated sort, and then the rest of the solutions are selected by using weight-sum method in the layer *MaxFNo* mentioned in Section 4.1.2. The environment selection of the second algorithm only selects individuals by weight-sum method, which is called WSEA2. From the point of minimizing the problem, the way to select individuals here is to pick out the N individuals with smallest weighted sum to the next generation. In addition, both algorithms use crossover and mutation operators to generate offspring. Finally, since there is no preference for an objective, the weights of the two algorithms on each objective are set to the same value of $1/M$. However, in order to consider the impact of each objective size on the algorithm, a normalization operation should be carried out for each objective before calculating the weighted sum. DL-TPCEA is compared with the two weight-sum based algorithms mentioned above, and the results are shown in Tables 8–11.

Table 8. HV results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	3	2.5940e+1 (3.02e+0)	6.9900e+0 (5.88e+0)	2.7839e+1 (1.96e+0)
	5	4.3534e+3 (2.87e+2)	1.5758e+3 (9.81e+2)	2.6535e+3 (2.70e+2)
	8	1.8437e+7 (1.20e+6)	1.1020e+7 (7.58e+6)	6.9852e+6 (1.19e+6)
WFG2	3	4.4022e+1 (1.02e+0)	1.0113e+1 (4.37e+0)	5.8855e+1 (2.83e−1)
	5	4.2104e+3 (9.81e+2)	2.2314e+3 (1.04e+3)	6.1122e+3 (1.83e+1)
	8	1.6107e+7 (2.25e+6)	1.6424e+7 (2.47e+6)	2.1889e+7 (8.02e+4)
WFG3	3	4.7170e+0 (3.09e−1)	1.4324e+0 (1.07e−2)	5.9102e+0 (9.30e−2)
	5	1.2563e+0 (9.19e−2)	1.0850e+0 (6.56e−3)	1.1142e+0 (3.37e−1)
	8	1.6284e−2 (2.65e−3)	1.3718e−2 (1.29e−3)	0.0000e+0 (0.00e+0)
WFG4	3	2.1540e+1 (2.61e−1)	5.8596e+0 (1.05e−1)	3.5106e+1 (1.47e−1)
	5	1.5536e+3 (1.18e+2)	7.4547e+2 (1.23e+2)	4.8905e+3 (2.30e+1)
	8	5.1626e+6 (1.10e+6)	4.0105e+6 (8.00e+5)	2.0259e+7 (1.20e+5)
WFG5	3	5.6176e+0 (3.55e−1)	4.9062e+0 (2.20e−4)	3.3072e+1 (2.45e−1)
	5	4.8517e+2 (7.18e−4)	4.8516e+2 (2.96e−6)	4.5798e+3 (1.63e+1)
	8	1.7489e+6 (4.13e+0)	1.7489e+6 (4.42e+0)	1.8993e+7 (7.80e+4)
WFG6	3	1.9143e+1 (8.51e−1)	5.2439e+0 (9.21e−1)	3.1280e+1 (8.81e−1)
	5	1.4861e+3 (4.26e+2)	8.3328e+2 (4.21e+2)	4.3856e+3 (8.73e+1)
	8	4.0300e+6 (1.18e+6)	2.3463e+6 (1.30e+5)	1.8534e+7 (4.34e+5)
WFG7	3	1.7538e+1 (1.22e+0)	9.4479e+0 (3.78e+0)	3.5763e+1 (9.24e−2)
	5	1.5602e+3 (3.41e+2)	8.5188e+2 (3.01e+2)	4.9534e+3 (8.29e+0)
	8	6.0797e+6 (7.26e+5)	4.7256e+6 (1.71e+6)	2.0675e+7 (3.51e+4)
WFG8	3	1.6581e+1 (2.50e+0)	5.6755e+0 (1.60e−1)	2.9372e+1 (2.48e−1)
	5	6.8821e+2 (6.54e+1)	6.1305e+2 (7.08e+1)	3.8241e+3 (4.35e+1)
	8	2.2857e+6 (6.42e+5)	2.0356e+6 (3.05e+5)	1.6523e+7 (2.10e+5)
WFG9	3	8.3332e+0 (1.99e+0)	3.4207e+0 (4.40e−5)	3.4084e+1 (2.54e−1)
	5	6.5479e+2 (9.07e+1)	4.4075e+2 (9.90e+1)	4.6044e+3 (4.08e+1)
	8	3.0759e+6 (7.92e+5)	2.8697e+6 (5.56e+5)	1.8621e+7 (1.34e+5)

Table 9. HV results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	10	8.3451e+9 (3.40e+8)	8.0823e+9 (8.38e+8)	3.0894e+9 (9.09e+7)
	15	1.3885e+17 (1.57e+14)	1.3151e+17 (7.93e+15)	3.9888e+16 (6.02e+15)
WFG2	10	7.7232e+9 (5.92e+8)	5.1805e+9 (1.83e+9)	9.5531e+9 (2.77e+7)
	15	1.2874e+17 (2.49e+16)	1.0902e+17 (2.21e+16)	1.7777e+17 (1.70e+14)
WFG3	10	3.8860e−5 (5.32e−6)	3.8105e−5 (5.43e−6)	0.0000e+0 (0.00e+0)
	15	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)	0.0000e+0 (0.00e+0)
WFG4	10	2.0785e+9 (6.77e+8)	2.5030e+9 (7.61e+8)	9.1189e+9 (4.33e+7)
	15	4.9360e+16 (2.12e+16)	3.1410e+16 (1.15e+16)	1.7439e+17 (6.93e+14)
WFG5	10	7.6254e+8 (9.65e−1)	7.6254e+8 (4.37e+1)	8.5253e+9 (2.97e+7)
	15	1.4147e+16 (1.75e+12)	1.4146e+16 (1.83e+12)	1.6240e+17 (3.51e+14)
WFG6	10	1.8340e+9 (5.01e+8)	1.7364e+9 (7.30e+8)	8.1978e+9 (2.35e+8)
	15	4.5548e+16 (1.85e+16)	3.2527e+16 (1.02e+16)	1.5706e+17 (2.76e+15)
WFG7	10	2.8722e+9 (6.09e+8)	2.2976e+9 (9.57e+8)	9.2994e+9 (7.01e+6)
	15	4.9910e+16 (1.06e+16)	3.4728e+16 (9.38e+15)	1.7724e+17 (1.34e+14)
WFG8	10	8.7211e+8 (9.80e+7)	8.6477e+8 (2.00e+8)	7.8199e+9 (1.26e+8)
	15	3.6695e+16 (2.82e+16)	4.2377e+16 (9.52e+15)	1.5825e+17 (3.19e+15)
WFG9	10	1.6703e+9 (9.25e+8)	1.4451e+9 (5.64e+8)	8.4373e+9 (7.94e+7)
	15	5.4094e+16 (7.47e+15)	4.6128e+16 (2.02e+16)	1.5617e+17 (1.04e+16)

As can be seen from the results in Tables 8–11, DL-TPCEA obtained the optimal results in all the other instances except for the HV results of seven instances on WFG1 and WFG3. Regardless of HV or IGD indicator, DL-TPCEA has the best performance among the three algorithms. It also reflects that the complexity of MaOPs cannot be well adapted to only relying on a single weighted sum method. The reasons are as follows: without considering the objective preference, it is not guaranteed that the solution in the population will converge to PF only by the magnitude of the weighted sum. A smaller weighted sum may just be that the individual retains a smaller objective value for some objective, but whether the individual is a non-dominated solution is unknown. In addition, the method based on the weighted sum is linearly convergent. However, different MaOPs have different characteristics, making it difficult to apply this method to all problems.

From the point of the feature of the problem, WFG1 is convex and mixed, while WFG3 is linear and degenerate. WSEA has just obtained the optimal HV results in several examples of these two problems, indicating the weighted sum method is promising to deal with these problems. However, the IGD values obtained by WSEA and WSEA2 on these examples are very poor, which also indicates that the convergence ability is not strong. The calculation of HV will consider some boundary individuals in the population, so DL-TPCEA may not get good HV results because of the boundary individuals in the population. However, combining the results of the two indicators, DL-TPCEA performed best in 83 out of 90 instances, which is an overwhelming advantage. These results reflect the limitations of using the weighted sum method to solve MaOPs, and show that DL-TPCEA has good advantages.

Table 10. IGD results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 3, 5, and 8 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	3	1.2881e+0 (1.35e−1)	3.4165e+0 (9.29e−1)	1.1443e+0 (7.65e−2)
	5	2.0015e+0 (1.18e−1)	5.0632e+0 (1.85e+0)	1.5296e+0 (1.11e−1)
	8	2.4999e+0 (2.12e−1)	7.6394e+0 (4.79e+0)	2.2369e+0 (2.02e−1)
WFG2	3	7.7414e−1 (4.59e−2)	3.1113e+0 (3.99e−1)	1.1400e−1 (8.60e−3)
	5	1.6495e+0 (4.13e−1)	2.7955e+0 (9.76e−1)	5.1156e−1 (6.59e−2)
	8	3.8914e+0 (6.35e−1)	3.5795e+0 (5.84e−1)	1.0066e+0 (2.18e−1)
WFG3	3	1.4793e+0 (1.96e−1)	3.2014e+0 (2.81e−3)	1.1065e−1 (1.13e−2)
	5	5.2586e+0 (1.01e−1)	5.4443e+0 (4.88e−4)	5.6591e−1 (6.60e−2)
	8	8.5239e+0 (3.90e−1)	8.8671e+0 (3.27e−2)	1.6935e+0 (3.31e−1)
WFG4	3	1.4651e+0 (1.51e−2)	3.9049e+0 (7.37e−2)	1.7016e−1 (3.78e−3)
	5	5.7459e+0 (2.93e−1)	7.4360e+0 (3.14e−1)	9.7239e−1 (9.44e−3)
	8	1.1621e+1 (1.03e+0)	1.2737e+1 (4.56e−1)	2.5307e+0 (1.51e−2)
WFG5	3	3.6430e+0 (9.53e−2)	3.8534e+0 (6.71e−5)	1.8664e−1 (3.07e−3)
	5	7.9990e+0 (5.15e−6)	7.9990e+0 (2.13e−8)	9.9293e−1 (9.04e−3)
	8	1.4670e+1 (9.68e−6)	1.4670e+1 (1.06e−5)	2.6346e+0 (3.32e−2)
WFG6	3	1.5039e+0 (7.48e−2)	3.7653e+0 (2.26e−1)	2.4262e−1 (1.89e−2)
	5	5.3476e+0 (8.43e−1)	7.0610e+0 (9.93e−1)	1.0488e+0 (1.30e−2)
	8	1.2326e+1 (1.24e+0)	1.3945e+1 (1.32e−1)	2.8073e+0 (4.48e−2)
WFG7	3	1.8260e+0 (1.95e−1)	3.1263e+0 (6.97e−1)	1.7794e−1 (5.79e−3)
	5	5.1307e+0 (5.05e−1)	7.2260e+0 (7.30e−1)	1.0808e+0 (2.16e−2)
	8	1.1043e+1 (4.85e−1)	1.1873e+1 (1.83e+0)	2.7405e+0 (2.75e−2)
WFG8	3	1.5861e+0 (3.63e−1)	3.8409e+0 (2.16e−1)	2.9003e−1 (5.52e−3)
	5	5.5266e+0 (1.98e−1)	6.6281e+0 (8.22e−1)	1.1148e+0 (1.25e−2)
	8	1.1010e+1 (1.07e+0)	1.3450e+1 (1.43e+0)	2.9559e+0 (4.04e−2)
WFG9	3	3.1525e+0 (2.30e−1)	3.8736e+0 (1.04e−6)	1.6080e−1 (3.23e−3)
	5	7.5318e+0 (2.87e−1)	7.9566e+0 (6.88e−2)	9.5847e−1 (1.16e−2)
	8	1.3446e+1 (6.85e−1)	1.3468e+1 (5.75e−1)	2.6905e+0 (4.47e−2)

Table 11. IGD results of DL-TPCEA and two weight-sum based algorithms on benchmarks WFG1-WFG9 with 10 and 15 objectives.

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG1	10	2.7506e+0 (4.24e−1)	3.2357e+0 (1.43e+0)	2.3727e+0 (1.48e−1)
	15	3.3141e+0 (1.74e−1)	3.5682e+0 (7.23e−1)	3.2972e+0 (2.62e−1)
WFG2	10	5.1728e+0 (6.06e−1)	6.3742e+0 (1.88e+0)	1.7920e+0 (3.61e−1)
	15	1.2806e+1 (1.47e+0)	1.3168e+1 (2.04e+0)	2.3050e+0 (7.98e−1)
WFG3	10	1.1169e+1 (2.14e−2)	1.1187e+1 (2.10e−3)	2.4613e+0 (2.26e−1)
	15	1.5948e+1 (1.23e+0)	1.6847e+1 (1.76e−1)	4.2693e+0 (1.23e+0)

Table 11. *Cont.*

Problems	M	WSEA	WSEA2	DL-TPCEA
WFG4	10	1.5880e+1 (1.63e+0)	1.5367e+1 (1.51e+0)	4.2136e+0 (1.04e−1)
	15	2.5245e+1 (2.83e+0)	2.7162e+1 (1.53e+0)	7.7824e+0 (1.11e−1)
WFG5	10	1.8913e+1 (6.20e−9)	1.8913e+1 (6.95e−8)	4.0344e+0 (3.86e−2)
	15	3.0022e+1 (6.26e−4)	3.0023e+1 (6.52e−4)	7.5313e+0 (6.01e−2)
WFG6	10	1.5554e+1 (1.64e+0)	1.6302e+1 (1.94e+0)	4.3695e+0 (2.07e−1)
	15	2.4092e+1 (2.70e+0)	2.5533e+1 (2.47e+0)	8.0243e+0 (1.90e−1)
WFG7	10	1.3849e+1 (1.58e+0)	1.4887e+1 (2.41e+0)	4.1518e+0 (6.56e−2)
	15	2.2743e+1 (2.69e+0)	2.6167e+1 (2.09e+0)	7.6130e+0 (8.70e−2)
WFG8	10	1.4975e+1 (1.68e+0)	1.5039e+1 (1.52e+0)	4.2410e+0 (4.91e−2)
	15	2.4219e+1 (4.49e+0)	2.1602e+1 (2.28e+0)	7.3664e+0 (2.52e−1)
WFG9	10	1.6810e+1 (2.26e+0)	1.7056e+1 (1.58e+0)	4.1677e+0 (2.88e−2)
	15	2.3731e+1 (1.25e+0)	2.5088e+1 (3.06e+0)	7.2405e+0 (1.43e−1)

6. Conclusions

In recent years, in order to enable MOEAs to handle MaOPs with various characteristics, various MOEAs have been proposed. However, these MOEAs also had their own disadvantages. For example, MOEAs that rely on reference vectors cannot well represent the characteristics of the whole PF when generating reference vectors, which results in the performance degradation of MOEAs. This paper made full use of the advantages of DLS in many-objective optimization (better to maintain convergence and diversity), and proposed DL-TPCEA in combination with the BCE framework. The effective combination of the two strategies can further explore the entire decision space. At the same time, the convergence factor in DLS is further improved according to the evolutionary state of the population in BCE, and then the dynamic convergence factor is proposed to better use the important element of the evolutionary state of the population. This effective combination greatly improves the performance of DL-TPCEA. When compared with five state-of-the-art MOEAs, DL-TPCEA has significant advantages. Finally, in order to verify the performance advantage of DL-TPCEA over the weight-sum based algorithm, DL-TPCEA was compared with the two weight-sum based algorithms, and the results showed that DL-TPCEA still had significant advantages.

In addition, the original DLS used $I_{\varepsilon+}$ to maintain individual convergence and a diversity maintenance mechanism based on L_p-norm distance to maintain diversity. In this paper, the CV indicator is used to maintain individual convergence, and the comparison between CV and $I_{\varepsilon+}$ should be the future research direction. In addition, there are still many excellent strategies that can be used to maintain convergence and diversity, and this paper does not compare these strategies. The future direction of work can start from this point and be improved under the framework of DL-TPCEA to achieve better results. We used dynamic learning factors to combine DLS and BCE more effectively, but there are more ways to combine them more effectively in the future. In terms of the selection of the initial value of the dynamic convergence factor, suggestions in relevant paper [94] can also be referred to get a better initial value.

Author Contributions: Conceptualization, G.L. and G.-G.W.; methodology, G.L.; software, G.L.; validation, G.-G.W. and S.W.; formal analysis, S.W.; investigation, G.L.; resources, G.L. and G.-G.W.; data curation, G.L.; writing—original draft preparation, G.L.; writing—review and editing, G.-G.W. and S.W.; visualization, G.L.; supervision, G.L., G.-G.W. and S.W.; project administration, G.-G.W. and S.W.; funding acquisition, G.-G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number U1706218, 41576011, 41706010 and 61503165.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, H.; Cheng, R.; Wen, J.; Li, H.; Weng, J. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Inf. Sci.* **2020**, *509*, 457–469. [CrossRef]
2. Yang, S.; Li, M.; Liu, X.; Zheng, J. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2013**, *17*, 721–736. [CrossRef]
3. Ishibuchi, H.; Tsukamoto, N.; Nojima, Y. Evolutionary many-objective optimization: A short review. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 2419–2426.
4. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
5. Deb, K.; Jain, H. An evolutionary many-objective optimization slgorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **2014**, *18*, 577–601. [CrossRef]
6. Jain, H.; Deb, K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **2014**, *18*, 602–622. [CrossRef]
7. Zhang, X.; Tian, Y.; Jin, Y. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2015**, *19*, 761–776. [CrossRef]
8. Zitzler, E.; Künzli, S. Indicator-based selection in multiobjective search. In Proceedings of the International conference on parallel problem solving from nature, Birmingham, UK, 18–22 September 2004; pp. 832–842.
9. Bader, J.; Zitzler, E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **2011**, *19*, 45–76. [CrossRef]
10. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.M.; da Fonseca, V.G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comput.* **2003**, *7*, 117–132. [CrossRef]
11. While, L.; Hingston, P.; Barone, L.; Huband, S. A faster algorithm for calculating hypervolume. *IEEE Trans. Evol. Comput.* **2006**, *10*, 29–38. [CrossRef]
12. Russo, L.M.S.; Francisco, A.P. Quick hypervolume. *IEEE Trans. Evol. Comput.* **2014**, *18*, 481–502. [CrossRef]
13. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
14. Wang, R.; Zhang, Q.; Zhang, T. Decomposition-based algorithms using Pareto adaptive scalarizing methods. *IEEE Trans. Evol. Comput.* **2016**, *20*, 821–837. [CrossRef]
15. Potter, M.A.; Jong, K.A.D. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.* **2000**, *8*, 1–29. [CrossRef] [PubMed]
16. Wiegand, R.P.; Potter, M.A. Robustness in cooperative coevolution. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 369–376.
17. Wiegand, R.P.; Liles, W.C.; de Jong, K.A. Analyzing cooperative coevolution with evolutionary game theory. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 1600–1605.
18. De Jong, K.A.; Potter, M.A. Evolving complex structures via cooperative coevolution. In *Evolutionary Programming IV*; McDonnell, J.R., Reynolds, R.G., Fogel, D.B., Eds.; The MIT Press: Cambridge, MA, USA, 1995; pp. 307–317.
19. Sofge, D.; de Jong, K.; Schultz, A. A blended population approach to cooperative coevolution for decomposition of complex problems. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 413–418.
20. Yang, Z.; Tang, K.; Yao, X. Large scale evolutionary optimization using cooperative coevolution. *Inf. Sci.* **2008**, *178*, 2985–2999. [CrossRef]
21. Yang, Z.; Tang, K.; Yao, X. Multilevel cooperative coevolution for large scale optimization. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 1663–1670.
22. Liu, Y.; Yao, X.; Zhao, Q.; Higuchi, T. Scaling up fast evolutionary programming with cooperative coevolution. In Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, South Korea, 27–30 May 2001; pp. 1101–1108.
23. Bucci, A.; Pollack, J.B. On identifying global optima in cooperative coevolution. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 25–29 June 2005; pp. 539–544.
24. Chen, W.; Weise, T.; Yang, Z.; Tang, K. Large-scale global optimization using cooperative coevolution with variable interaction learning. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Kraków, Poland, 11–15 September 2010; pp. 300–309.
25. Tan, T.G.; Teo, J.; Lau, H.K. Performance scalability of a cooperative coevolution multiobjective evolutionary algorithm. In Proceedings of the 2007 IEEE International Conference on Computational Intelligence and Security, Washington, DC, USA, 15–19 December 2007; pp. 119–123.
26. Wang, R.; Purshouse, R.C.; Fleming, P.J. Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Trans. Evol. Comput.* **2012**, *17*, 474–494. [CrossRef]

27. Purshouse, R.C.; Jalbă, C.; Fleming, P.J. Preference-driven co-evolutionary algorithms show promise for many-objective optimisation. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Ouro Preto, Brazil, 5–8 April 2011; pp. 136–150.
28. Wang, R.; Purshouse, R.C.; Fleming, P.J. On finding well-spread Pareto optimal solutions by preference-inspired co-evolutionary algorithm. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 695–702.
29. Wang, R.; Purshouse, R.C.; Fleming, P.J. Preference-inspired co-evolutionary algorithms using weight vectors. *Eur. J. Oper. Res.* **2015**, *243*, 423–441. [CrossRef]
30. Liang, Z.; Wang, X.; Lin, Q.; Chen, F.; Chen, J.; Ming, Z. A novel multi-objective co-evolutionary algorithm based on decomposition approach. *Appl. Soft Comput.* **2018**, *73*, 50–66. [CrossRef]
31. Zhang, Y.-H.; Gong, Y.-J.; Gu, T.-L.; Yuan, H.-Q.; Zhang, W.; Kwong, S.; Zhang, J. DECAL: Decomposition-based coevolutionary algorithm for many-objective optimization. *IEEE Trans. Cybern.* **2017**, *49*, 27–41. [CrossRef] [PubMed]
32. Shu, Z.; Wang, W. Preference-inspired co-evolutionary algorithms with local PCA oriented goal vectors for many-objective optimization. *IEEE Access* **2018**, *6*, 68701–68715. [CrossRef]
33. Antonio, L.M.; Coello, C.A.C. Use of cooperative coevolution for solving large scale multiobjective optimization problems. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2758–2765.
34. Gong, D.; Xu, B.; Zhang, Y.; Guo, Y.; Yang, S. A similarity-based cooperative co-evolutionary algorithm for dynamic interval multi-objective optimization problems. *IEEE Trans. Evol. Comput.* **2019**, *24*, 142–156. [CrossRef]
35. Sun, J.; Miao, Z.; Gong, D.; Zeng, X.; Li, J.; Wang, G. Interval multiobjective optimization with memetic algorithms. *IEEE Trans. Cybern.* **2020**, *50*, 3444–3457. [CrossRef]
36. Yi, J.-H.; Xing, L.-N.; Wang, G.-G.; Dong, J.; Vasilakos, A.V.; Alavi, A.H.; Wang, L. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Inf. Sci.* **2020**, *509*, 470–487. [CrossRef]
37. González, J.; Ortega, J.; Damas, M.; Martín-Smith, P. Many-objective cooperative co-evolutionary feature selection: A lexicographic approach. In Proceedings of the International Work-Conference on Artificial Neural Networks, Gran Canaria, Spain, 12–14 June 2019; pp. 463–474.
38. Wang, F.; Li, Y.; Liao, F.; Yan, H. An ensemble learning based prediction strategy for dynamic multi-objective optimization. *Appl. Soft Comput.* **2020**, *96*, 106592. [CrossRef]
39. Wang, F.; Li, Y.; Zhang, H.; Hu, T.; Shen, X.-L. An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization. *Swarm Evol. Comput.* **2019**, *49*, 220–233. [CrossRef]
40. Shen, X.; Guo, Y.; Li, A. Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling. *Appl. Soft Comput.* **2020**, *88*, 106059. [CrossRef]
41. Liu, X.; Zhan, Z.; Gao, Y.; Zhang, J.; Kwong, S.; Zhang, J. Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization. *IEEE Trans. Evol. Comput.* **2019**, *23*, 587–602. [CrossRef]
42. Zhou, Y.; Yang, J.; Zheng, L. Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling. *IEEE Access* **2019**, *7*, 68–88. [CrossRef]
43. Li, M.; Yang, S.; Liu, X. Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization. *IEEE Trans. Evol. Comput.* **2015**, *20*, 645–665. [CrossRef]
44. Yuan, J.; Liu, H.-L.; Gu, F. A cost value based evolutionary many-objective optimization algorithm with neighbor selection strategy. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
45. Sang, H.-Y.; Pan, Q.-K.; Li, J.-Q.; Wang, P.; Han, Y.-Y.; Gao, K.-Z.; Duan, P. Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm Evol. Comput.* **2019**, *44*, 64–73. [CrossRef]
46. Gao, D.; Wang, G.G.; Pedrycz, W. Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 3265–3275. [CrossRef]
47. Li, J.-Q.; Pan, Q.-K.; Tasgetiren, M.F. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Appl. Math. Model.* **2014**, *38*, 1111–1132. [CrossRef]
48. Han, Y.-Y.; Gong, D.; Sun, X. A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Eng. Optimiz.* **2015**, *47*, 927–946. [CrossRef]
49. Sang, H.-Y.; Pan, Q.-K.; Duan, P.-Y.; Li, J.-Q. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *J. Intell. Manuf.* **2018**, *29*, 1337–1349. [CrossRef]
50. Wang, G.-G.; Gandomi, A.H.; Zhao, X.; Chu, H.C. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
51. Zhao, X.-C. A perturbed particle swarm algorithm for numerical optimization. *Appl. Soft Comput.* **2010**, *10*, 119–124.
52. Wang, H.; Yi, J.-H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Comput.* **2018**, *10*, 177–198. [CrossRef]
53. Liu, F.; Sun, Y.; Wang, G.-G.; Wu, T. An artificial bee colony algorithm based on dynamic penalty and Lévy flight for constrained optimization problems. *Arab. J. Sci. Eng.* **2018**, *43*, 7189–7208. [CrossRef]
54. Zhang, Y.; Gong, D.; Hu, Y.; Zhang, W. Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing* **2015**, *148*, 150–157. [CrossRef]
55. Wang, G.-G.; Deb, S.; Cui, Z. Monarch butterfly optimization. *Neural Comput. Appl.* **2019**, *31*, 1995–2014. [CrossRef]

56. Wang, G.-G.; Deb, S.; Zhao, X.; Cui, Z. A new monarch butterfly optimization with an improved crossover operator. *Oper. Res.* **2018**, *18*, 731–755. [CrossRef]
57. Feng, Y.; Wang, G.-G.; Li, W.; Li, N. Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem. *Neural Comput. Appl.* **2018**, *30*, 3019–3036. [CrossRef]
58. Feng, Y.; Deb, S.; Wang, G.-G.; Alavi, A.H. Monarch butterfly optimization: A comprehensive review. *Expert Syst. Appl.* **2021**, *168*, 114418. [CrossRef]
59. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
60. Zhao, D.; Liu, L.; Yu, F.; Heidari, A.A.; Wang, M.; Liang, G.; Muhammad, K.; Chen, H. Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2D Kapur entropy. *Knowl. Based Syst.* **2021**, *216*, 106510. [CrossRef]
61. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simulat.* **2012**, *17*, 4831–4845. [CrossRef]
62. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. An effective krill herd algorithm with migration operator in biogeography-based optimization. *Appl. Math. Model.* **2014**, *38*, 2454–2462. [CrossRef]
63. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2019**, *51*, 119–148. [CrossRef]
64. Wang, G.-G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
65. Li, W.; Wang, G.-G.; Alavi, A.H. Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowl. Based Syst.* **2020**, *195*, 105675. [CrossRef]
66. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.-G. Elephant Herding Optimization: Variants, Hybrids, and Applications. *Mathematics* **2020**, *8*, 1415. [CrossRef]
67. Li, W.; Wang, G.-G.; Gandomi, A.H. A Survey of learning-based intelligent optimization algorithms. *Arch. Comput. Method. E* **2021**, in press. [CrossRef]
68. Wang, G.-G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2019**, *49*, 542–555. [CrossRef] [PubMed]
69. Wang, F.; Li, Y.; Zhou, A.; Tang, K. An estimation of distribution algorithm for mixed-variable newsvendor problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 479–493. [CrossRef]
70. Rizk-Allah, R.M.; El-Sehiemy, R.A.; Deb, S.; Wang, G.-G. A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. *J. Supercomput.* **2017**, *73*, 1235–1256. [CrossRef]
71. Hemmelmayr, V.C.; Cordeau, J.-F.; Crainic, T.G. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Comput. Oper. Res.* **2012**, *39*, 3215–3228. [CrossRef]
72. Yu, H.; Li, W.; Chen, C.; Liang, J.; Gui, W.; Wang, M.; Chen, H. Dynamic Gaussian bare-bones fruit fly optimizers with abandonment mechanism: Method and analysis. *Eng. Comput. Germany* **2020**, in press. [CrossRef]
73. Yu, H.; Zhao, N.; Wang, P.; Chen, H.; Li, C. Chaos-enhanced synchronized bat optimizer. *Appl. Math. Model.* **2020**, *77*, 1201–1215. [CrossRef]
74. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comp. Syst.* **2020**, *111*, 300–323. [CrossRef]
75. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comp. Syst.* **2019**, *97*, 849–872. [CrossRef]
76. Hu, J.; Chen, H.; Heidari, A.A.; Wang, M.; Zhang, X.; Chen, Y.; Pan, Z. Orthogonal learning covariance matrix for defects of grey wolf optimizer: Insights, balance, diversity, and feature selection. *Knowl. Based Syst.* **2021**, *213*, 106684. [CrossRef]
77. Tu, J.; Chen, H.; Liu, J.; Heidari, A.A.; Zhang, X.; Wang, M.; Ruby, R.; Pham, Q.-V. Evolutionary biogeography-based whale optimization methods with communication structure: Towards measuring the balance. *Knowl. Based Syst.* **2021**, *212*, 106642. [CrossRef]
78. Zitzler, E.; Laumanns, M.; Thiele, L. SPEA2: Improving the strength Pareto evolutionary algorithm. In Proceedings of the Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, Athens, Greece, 19–21 September 2001; pp. 95–100.
79. Cheng, R.; Jin, Y.; Olhofer, M.; Sendhoff, B. Test problems for large-scale multiobjective and many-objective optimization. *IEEE Trans. Cybern.* **2017**, *47*, 4108–4121. [CrossRef] [PubMed]
80. Ma, X.; Liu, F.; Qi, Y.; Wang, X.; Li, L.; Jiao, L.; Yin, M.; Gong, M. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Trans. Evol. Comput.* **2015**, *20*, 275–298. [CrossRef]
81. Zhang, X.; Tian, Y.; Cheng, R.; Jin, Y. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Trans. Evol. Comput.* **2018**, *22*, 97–112. [CrossRef]
82. Liu, R.; Li, J.; Fan, J.; Jiao, L. A dynamic multiple populations particle swarm optimization algorithm based on decomposition and prediction. *Appl. Soft Comput.* **2018**, *73*, 434–459. [CrossRef]
83. Rambabu, R.; Vadakkepat, P.; Tan, K.C.; Jiang, M. A mixture-of-experts prediction framework for evolutionary dynamic multiobjective optimization. *IEEE Trans. Cybern.* **2020**, *50*, 5099–5112. [CrossRef] [PubMed]
84. Guo, Y.; Yang, H.; Chen, M.; Cheng, J.; Gong, D. Ensemble prediction-based dynamic robust multi-objective optimization methods. *Swarm Evol. Comput.* **2019**, *48*, 156–171. [CrossRef]

85. Muruganantham, A.; Tan, K.C.; Vadakkepat, P. Evolutionary dynamic multiobjective optimization via Kalman filter prediction. *IEEE Trans. Cybern.* **2015**, *46*, 2862–2873. [CrossRef] [PubMed]
86. Nebro, A.J.; Ruiz, A.B.; Barba-González, C.; García-Nieto, J.; Luque, M.; Aldana-Montes, J.F. InDM2: Interactive dynamic multi-objective decision making using evolutionary algorithms. *Swarm Evol. Comput.* **2018**, *40*, 184–195. [CrossRef]
87. Gee, S.B.; Tan, K.C.; Alippi, C. Solving multiobjective optimization problems in unknown dynamic environments: An inverse modeling approach. *IEEE Trans. Cybern.* **2016**, *47*, 4223–4234. [CrossRef]
88. Wang, H.; Jiao, L.; Yao, X. Two_Arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2014**, *19*, 524–541. [CrossRef]
89. Huband, S.; Barone, L.; While, L.; Hingston, P. *A Scalable Multi-Objective Test Problem Toolkit. Evolutionary Multi-Criterion Optimization*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 280–295.
90. Huband, S.; Hingston, P.; Barone, L.; While, L. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **2006**, *10*, 477–506. [CrossRef]
91. Das, I.; Dennis, J.E. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.* **1998**, *8*, 631–657. [CrossRef]
92. Zhang, X.; Zheng, X.; Cheng, R.; Qiu, J.; Jin, Y. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Inf. Sci.* **2018**, *427*, 63–76. [CrossRef]
93. Coello, C.A.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 1051–1056.
94. D'Angelo, G.; Palmieri, F. GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems. *Inf. Sci.* **2021**, *547*, 136–162. [CrossRef]