applied sciences

MDPI

Article

Memory-Enhanced Dynamic Multi-Objective Evolutionary Algorithm Based on L_p Decomposition

Xinxin Xu [1], Yanyan Tan [1,2,*], Wei Zheng [3] and Shengtao Li [1,2]

1 School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China; 2016020920@stu.sdnu.edu.cn (X.X.); shengtaoli1985@163.com (S.L.)
2 Institute of Data Science and Technology, Shandong Normal University, Jinan 250358, China
3 School of Mathematics and Statistics, Xi'an JiaoTong University, Xi'an 710049, China; zhengshu56@126.com
* Correspondence: yytan928@sdnu.edu.cn; Tel.: +86-0531-8961-0750

Received: 17 August 2018; Accepted: 4 September 2018; Published: 15 September 2018

Abstract: Decomposition-based multi-objective evolutionary algorithms provide a good framework for static multi-objective optimization. Nevertheless, there are few studies on their use in dynamic optimization. To solve dynamic multi-objective optimization problems, this paper integrates the framework into dynamic multi-objective optimization and proposes a memory-enhanced dynamic multi-objective evolutionary algorithm based on L_p decomposition (denoted by dMOEA/D-L_p). Specifically, dMOEA/D-L_p decomposes a dynamic multi-objective optimization problem into a number of dynamic scalar optimization subproblems and coevolves them simultaneously, where the L_p decomposition method is adopted for decomposition. Meanwhile, a subproblem-based bunchy memory scheme that stores good solutions from old environments and reuses them as necessary is designed to respond to environmental change. Experimental results verify the effectiveness of the L_p decomposition method in dynamic multi-objective optimization. Moreover, the proposed dMOEA/D-L_p achieves better performance than other popular memory-enhanced dynamic multi-objective optimization algorithms.

Keywords: multi-objective evolutionary optimization; memory enhancement; dynamic environment; decomposition method

1. Introduction

In the real world, many problems can be described as multi-objective optimization problems (MOPs), where multiple objectives often conflict with each other. Each objective has to compromise in optimization, and finally generate a set of balanced solutions, called the Pareto optimal set (POS) [1], which is provided to decision makers. Evolutionary multi-objective optimization, referred to EMO, focuses on using evolutionary algorithms for MOPs [2]. Its research has become a hot topic in the field of evolutionary computation. MOPs can be further divided into static and dynamic multi-objective problems (DMOPs). In the dynamic environment, the objective function, constraint function, and the related parameters of DMOPs can change with time [3]. Therefore, a dynamic evolutionary multi-objective optimization algorithm (DEMOA) must be able to automatically detect and respond to new changes with fast convergence, and track the time-varying POS in a timely fashion. As a result, this study brings new challenges to EMO [4]. In the past ten years, some researchers have had strong interest in dynamic EMO (DEMO) [5,6]. Based on genetic algorithms, artificial immune algorithms [7,8], particle swarm optimization algorithms [9,10], co-evolutionary algorithms [11], membrane computing [12,13], and other natural computing methods, they designed the corresponding DEMOAs [14]. DEMOAs have been preliminarily used in intelligent service [15], industrial design [16],

engineering management [17], scheduling control and optimization [18], and other fields. For their further researches have important theoretical significance and practical applied value.

How to make a rapid response to a new environment using an obtained optimal solution is very important in the design of a DEMOA. Memory methods can reuse the previous optimal solutions by remembering to help the algorithm make a good response to the new changes. There are some dynamic single objective evolutionary algorithms with memory methods to enhance the dynamic tracking performance [19,20]. So far, however, few memory methods are used in DEMOAs. In 2007, the famous dynamic non-dominated sorting genetic algorithm II (DNSGA-II) [16] was proposed by Deb et al., which was based on NSGA-II [21]. In 2009, Goh et al. proposed a collaborative multi-objective evolutionary algorithm for dynamic competition cooperation, named the dynamic competition-cooperation co-evolutionary algorithm (dCOEA) [12]. Koo and Goh et al. proposed a dynamic multi-objective evolutionary gradient search algorithm (dMO-EGS) [22] in 2010. Shang et al. proposed an immune clonal coevolutionary algorithm for dynamic multi-objective optimization (QICCA) [7] in 2014. All these DEMOAs have tried to use memory methods to quickly respond to new changes, but the actual effects are not ideal.

In addition to rapid response and dynamic tracking, another difficulty in designing a DEMOA is that the Pareto solutions set obtained by the algorithm should has good convergence and distribution in each process of environment change. At present, most DEMOAs follow the traditional approaches of static evolutionary multi-objective optimization and use the Pareto domination and elite reserved strategies to ensure the convergence. They also use the operators of diversity maintaining to ensure the diversity of the Pareto solutions. Zhang et al. proposed a new framework called the multi-objective evolutionary algorithm based on decomposition (MOEA/D) for solving static MOPs in 2007 [23]. MOEA/D decomposes a multi-objective problem into a number of single-objective sub-problems and solves these subproblems in parallel. MOEA/D has become very popular in solving static MOPs, but fewer decomposition-based DMOEAs are researched for DMOPs. In this paper, we introduce this framework and propose a memory-enhanced dynamic multi-objective algorithm based on L_p decomposition for solving DMOPs, where the L_p decomposition method is presented. The proposed algorithm is abbreviated as dMOEA/D-L_p.

The rest of this paper is organized as follows. Section 2 introduces some related works, including the basic definitions of dynamic optimization. Section 3 introduces the background knowledge on decomposition approaches and memory methods. Section 3.4 describes the framework of our proposed algorithm: dMOEA/D-L_p (memory-enhanced dynamic multi-objective algorithm based on L_p decomposition). Section 4 presents experimental studies on dMOEA/D-L_p. Finally, Section 5 concludes.

2. Related works

2.1. Problem Description and Basic Definitions

Without loss of generality, an MOP with n decision variables and m objective functions can be described as:

$$\begin{cases} \min\limits_{x\in\Omega} F(x) = (f_1(x), f_2(x), ..., f_m(x))^\top, \\ \text{s. t. } g_i(x) \le 0, i = 1, 2, ..., s; h_j(x) = 0, j = 1, 2, ..., q, \end{cases} \tag{1}$$

where $x = (x_1, x_2, ..., x_n)^T \in \Omega \subset R^n$ is a vector of decision variables, Ω is the n-dimensional decision space, and $F(x) = (f_1, f_2, ..., f_m)^T \in \Lambda \subset R^m$ is an m-dimensional vector of m objective functions. R^m is called the objective space. The evaluation function $F(x) : \Omega \to \Lambda$ defines m mapping from the decision space to the objective space. $g_i(x) \le 0 (i = 1, 2, ..., s)$ are some inequality constraints, and $h_j(x) = 0 (j = 1, 2, ..., q)$ are some q equality constraints [24].

The time variable of t is included in DMOP. Based on a static multi-objective problem, we can make an extension naturally. It involves an n-dimensional decision variable x, m-objective functions, and the constraint conditions. The objective functions and the constraint conditions are functions of the decision variable x. The mathematical model of a DMOP is expressed as:

$$\begin{cases} \min\limits_{x\in\Omega} F(x,t) = (f_1(x,t), f_2(x,t), ..., f_m(x,t))^\top, \\ \text{s. t. } g_i(x,t) \leq 0, i = 1,2,...,s; h_j(x,t) = 0, j = 1,2,...,q, \end{cases} \tag{2}$$

where the evaluation function $F(x,t)$, the inequality constraint function $g_i(x,t)$, and the equality constraint function $h_j(x,t)$ may change with time t.

Definition 1 (The relation of Pareto dominance). *For any two decision vectors u and v, the vector $u = (u_1, u_2,..., u_m)^T$ dominates vector $v = (v_1, v_2,..., v_m)^T$, denoted by $u \prec v$, only if : $\forall k \in \{1,2,...,m\}$, $u_k \leq v_k$ and $\exists l \in \{1,2,...,m\}$, $u_l < v_l$.*

Definition 2 (Pareto optimal solution). *The decision vector $x \in \Omega$ is the Pareto optimal solution, if and only if $\neg\exists x' \in \Omega$ satisfying $F(x',t) \prec F(x,t)$.*

Definition 3 (Pareto optimal set (POS(t))). *The Pareto optimal set (POS) is the set that consists of all Pareto optimal solutions. The POS of a DMOP is defined as $POS(t) : \{x \in \Omega \,|\neg\exists x' \in \Omega, F(x',t) \prec F(x,t)\}$.*

Definition 4 (Pareto optimal front (POF(t))). *The Pareto optimal front (**POF**) of a DMOP is defined as $POF(t) := \{F(x,t) \,|x \in POS(t)$. The Pareto optimal front (**POF**) is the set of all the Pareto optimal objective vectors.*

From the proposal of Farina et al. [3], there are four different types of DMOPs according to the changes affecting the Pareto optimal front and the Pareto optimal set.

- Type I: where **POS**(t) changes while **POF**(t) remains invariant.
- Type II: where both **POS**(t) and **POF**(t) change.
- Type III: where **POF**(t) changes while **POS**(t) remains invariant.
- Type IV: where both **POS**(t) and **POF**(t) remain invariant.

This classification shows the difficulty of solving DMOPs by describing the combination of changes in the Pareto set and front.

2.2. Decomposition Methods

Decomposition-based multi-objective evolutionary algorithms have become an extremely prevailing framework for multi-objective optimization. Their main idea is to decompose an MOP into a number of scalaring sub-problems and solve them in parallel, where the decomposition method plays an extremely important role. There are three commonly used decomposition methods [23]: weighted sum approach (WS) [25], Tchebycheff approach (TCH) [25] and the penalty-based boundary intersection approach (PBI) [23,26], described as follows.

- Weighted Sum (WS) approach

The DMOPs of Formula (2) are decomposed into N dynamic scalar optimization subproblems by using N different uniformly distributed weight vectors $\lambda^i = (\lambda_1^i, \lambda_2^i, ..., \lambda_m^i)(i = 1,2,...,N)$, where for all $i = 1,2,..,N$, $\lambda_j^i \geq 0$ and $\sum\limits_{j=1}^{m} \lambda_j^i = 1$.

With this approach, the decomposed scalar subproblem can be described as follows:

$$\begin{cases} \min\limits_{x\in\Omega} g^{ws}(x,t\,|\lambda\,) = \sum\limits_{i=1}^{m} \lambda_i f_i(x,t), \\ \text{s. t. } g_i(x,t) \le 0, i = 1,2,...,s; h_j(x,t) = 0, j = 1,2,...,q. \end{cases} \tag{3}$$

$x = (x_1, x_2, ..., x_n)^T$ are n variables to be optimized. We can use different weight vectors to generate a set of different scalar optimization problems. $g_i(x,t)$ and $h_j(x,t)$ are s inequality and q equality constraint functions, respectively.

- Tchebycheff (TCH) approach

Referring to the standard Tchebycheff formula in the static multi-objective problem, under the condition of adding time, a dynamic Tchebycheff scalar optimization problem can be described as:

$$\begin{cases} \min\limits_{x\in\Omega} g^{tch}(F(x,t)\,|\lambda, z^*(t)\,) = \max\limits_{1\le i\le m}\{\lambda_i\,|f_i(x,t) - z^*(t)|\}, \\ \text{s. t. } g_i(x,t) \le 0, i = 1,2,...,s; h_j(x,t) = 0, j = 1,2,...,q, \end{cases} \tag{4}$$

where $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)^T$ is the same as the WS approach, t is a time variable, and $z^*(t) = (z_1^*(t), ..., z_m^*(t))^T$ is a reference point. That is, for each $i = 1, ..., m$, $z_i^*(t) = \min\{f_i(x,t)\,|g(x,t) \le 0; h(x,t) = 0; x \in \Omega\}$. For each Pareto optimal solution x^* of MOPs in Formula (2), there is a corresponding weight vector λ, so that x^* is the optimal solution of Formula (4). Each optimal solution of Formula (4) is also the Pareto optimal solution [23] of MOPs in Formula (2). Therefore, different Pareto optimal solutions can be obtained by changing the weight vectors. Thus, MOPs can be decomposed and transformed into multiple Tchebycheff scalar optimization subproblems with different weight vectors.

- Penalty-based boundary intersection (PBI) approach

A dynamic PBI scalar optimization problem can be described as:

$$\begin{cases} \min\limits_{x\in\Omega} g^{pbi}(x,t\,|\lambda, z^*(t)\,) = d_1 + \theta d_2, \\ d_1 = \dfrac{\left\|(z^*(t) - F(x,t))^T\lambda\right\|}{\|\lambda\|}, d_2 = \|F(x,t) - (z^*(t) - d_1\lambda)\|, \\ \text{s. t. } g_i(x,t) \le 0, i = 1,2,...,s; h_j(x,t) = 0, j = 1,2,...,q, \end{cases} \tag{5}$$

where $g^{pbi}(x,t\,|\lambda, z^*(t)\,)$ is dynamic scalar objective function. z^* is the ideal reference point as defined in (4), and θ is the default penalty parameter. $F(x,t)$ is the evaluation function in Formula (2), d_1 and d_2 are two distance values [23], t is a time variable. According to previous experience [23], when using the same distributed weighted vectors, the PBI approach has a slight advantage over the TCH approach when solving MOPs with more than two objectives. However, the benefits come with the price that the penalty parameter θ needs to be adjusted properly.

2.3. Memory-Enhanced Algorithm for Environmental Change

Many practical optimization problems show periodic change approximation, such as city traffic changes [20], where the optimal solution in the new environment may return to the previously searched locations. Then, with a memory method reusing the previously searched solutions, a dynamic evolutionary algorithm will have a better tracking performance. According to how much information needs to be stored, memory methods can commonly be divided into short-term memory and medium-term memory (given the limited computing resources, long-term memory methods are rarely used). Short-term memory methods only need to remember the optimal solutions of the last changed environment. On the contrary, in medium-term memory methods, the optimal solutions of several previous environmental changes need to be remembered.

2.3.1. DEMO Algorithm with Short-Term Memory

(1) The immune clonal coevolutionary algorithm for dynamic multi-objective optimization (QICCA) algorithm is a short-term memory approach [7].

To ensure a good convergence rate, the QICCA algorithm makes the final antibody population as the initial population of the next moment. The algorithm preserves all the best solutions in the previous change environment and has an effect of short-term memory.

(2) DNSGA-II algorithm with short-term memory and diversity [23]

The DNSGA-II algorithm (this paper named it short-term memory and diversity introduction, or SMDI) responds to new environmental change through a short-term memory and diversity introduction operation. SMDI first retains most of the individuals in the previous generation of environmental change. It has a short-term memory effect. Then, in order to respond to new changes, it randomly initializes some other individuals in the current population to introduce a small amount of diversity to the population.

The memory methods in the above two DEMOAs are simple and easy to use, but they can only remember the optimal solutions of the last previous environmental change, and thus their memory capacity is limited. Especially when the environments change sharply, the memory effect is basically lost.

2.3.2. DEMO Algorithm with Medium-Term Memory

If an evolutionary algorithm is added to such a medium-term memory method (memory pool + store procedure + retrieve process), among them the memory pool holds the best solutions of the past. The store procedure is responsible for storing the optimal solutions in the memory pool, and the retrieval process is responsible for retrieving the optimal solutions from the memory pool and inserting them into the population of the new environment. Then, the evolutionary algorithm can reuse the optimal solutions of several previous environmental changes and have a better response to the new change. According to Branke [27], there are three important issues in the design of medium-term memory methods.

(1) When should put the individuals deposited into the memory pool in the population?
(2) How many individuals should be stored in the memory pool, and which individuals should be replaced to make room for the memory pool to accommodate new individuals?
(3) Which individuals are retrieved from the memory pool and reinserted into the population?

3. Memory-Enhanced Dynamic Multi-Objective Evolutionary Algorithm Based on L_p Decomposition

A memory-enhanced dynamic multi-objective evolutionary algorithm based on L_p decomposition (dMOEA/D-L_p) is proposed in this paper. Specifically, dMOEA/D-L_p decomposes a dynamic multi-objective optimization problem into a number of dynamic scalar optimization subproblems and optimizes them simultaneously, where the L_p decomposition method is presented and used. Meanwhile, an improved environment detection operator is presented for dynamic environments. Additionally, a subproblem-based memory scheme that allows the evolutionary algorithm to store good solutions from old environments and reuse them as necessary is designed to respond to the environmental change. The overall framework of dMOEA/D-L_p can be seen at the end of this subsection. We present the L_p decomposition method in step 2 of the dMOEA/D-L_p. An improved environmental change detection operator and a subproblem-based bunchy memory scheme are presented in step 3 of the dMOEA/D-L_p. In step 4, we use differential evolutionary methods [28,29] to optimize the decomposed subproblems.

3.1. L_p Decomposition Used in dMOEA/D-L_p

In this paper, the L_p decomposition method is introduced and used in dMOEA/D-L_p. Its detailed description can be presented as follows:

$$\begin{cases} \min\limits_{x\in\Omega} g^{L_p}(x,t\,|\lambda,z^*(t)\,) = \sum_{i=1}^{m}\{\lambda_i(|f_i(x,t)-z_i^*(t)|^p)\}^{1/p}, \\ \text{s. t. } g_i(x,t)\le 0, i=1,2,...,s; h_j(x,t)=0, j=1,2,...,q, \end{cases} \tag{6}$$

where $g^{L_p}(x,t\,|\lambda,z^*(t)\,)$ is the scalar objective function. m is the number of objectives, $\lambda=(\lambda_1,...,\lambda_m)^T$, for all $i=1,...,m, \lambda_i\ge 0$ and $\sum_{i=1}^{m}\lambda_i=1$. $p\in(0,\infty]$, when $p=1$, g^{L_p} becomes the weighted sum method, and when $p=\infty$, g^{L_p} becomes the standard Tchebycheff function. The purpose of formula $\left|f_i(x,t)-z_i^*(t)\right|$ is to minimize the maximum deviation. We have drawn the contour lines of the improved function with different values in Figure 1, showing that with the increase of p, the search ability of L_p function decreases. All the g^{L_p} functions suffer from the geometry issue except for the Tchebycheff function. Especially, a g^{L_p} function cannot identify the whole Pareto optimal front of an MOP when the curvature of the Pareto optimal front is larger than the curvature of the contour line of the chosen g^{L_p} function. Since the curvature of the Tchebycheff function is ∞, it is able to identify Pareto optimal solutions for any type of geometry.

Overall, for a g^{L_p} scalar function, its search ability and its robustness on problem geometries is a trade-off. The higher the search ability, the lower the robustness. If we can estimate the curvature of the Pareto optimal front in advance, then we can easily select a suitable g^{L_p} function based on its search ability. For example, if the curvature of the Pareto optimal front is quadratic, the most suitable g^{L_p} function should be one using $p=2$. Alternatively, for a new problem or a problem having a complex geometry, the use of the Tchebycheff function is a good choice.

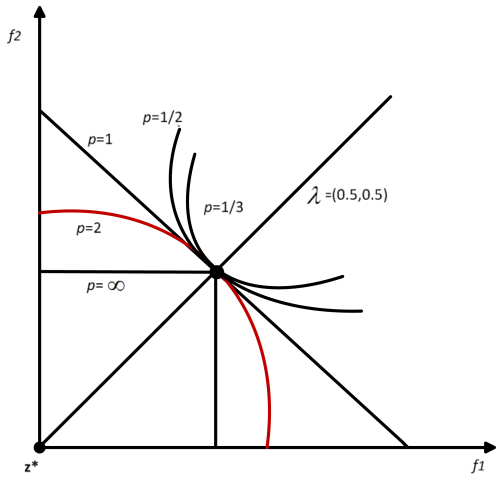

Figure 1. Contour lines of the g^{L_p} function.

3.2. Environmental Change Detection Operator

Many researchers have studied environmental change detection operators. However, the current detection operators either ignore the detection of the constraint functions or mistake the small noise generated by the objectives or constraint functions in the evaluation process for environmental changes [23]. A new and improved environment change detection operator [30] is introduced and adopted in our proposed dMOEA/D-L_p. It can detect the changes of objective functions and constraint functions at the same time, and set a threshold to eliminate the negative effects of noise as much as possible. The newly introduced operator is shown in Equation (7):

$$\delta(t) = \frac{\sum_i^K \left(\frac{\|F(x^i,t)-F(x^i,t-1)\|}{\|F(x^i,t-1)\|+\varepsilon} + \frac{\|g(x^i,t)-g(x^i,t-1)\|}{\|g(x^i,t-1)\|+\varepsilon} + \frac{\|h(x^i,t)-h(x^i,t-1)\|}{\|h(x^i,t-1)\|+\varepsilon} \right)}{K} > \tilde{\delta}, \tag{7}$$

where $F(x^i,t)$ is an evaluation function, $g(x^i,t)$ is an inequality constraint function, and $h(x^i,t)$ is an equality constraint function. Since they all may change with time t, we use the averaging method to detect three functions at the same time. ε is an arbitrary small positive number, $\|.\|$ represents Euclidean distance, K represents the number of individuals randomly selected from the population (generally available as $1 \leq K \leq 5$). According to [31], the Euclidean distance of the intergenerational target generally does not exceed 10^{-2} in the case of no change for environment. Therefore, we set a pre-threshold $\tilde{\delta}$ ($10^{-3} \leq \tilde{\delta} \leq 10^{-2}$). When $\delta(t) > \tilde{\delta}$, it means the environment has changed.

3.3. Subproblem-Based Bunchy Memory (SBM) Method to Respond to Environmental Change

A subproblem-based bunchy memory (SBM) method [30] is revised and applied in dMOEA/D-L_p for responding to environment change. The SBM method can be summarized as follows: whenever a change in the environment is detected, a series of representative solutions are first extracted from some subproblems to the memory pool. Then, it retrieves memory information and reuses the best solutions of previous environmental changes to respond to new changes. The memory pool has the same size as the population. Each string saved in the memory pool is considered as an organic whole, and the memory pool is made up of a series queue (string as the basic element of the queue). In keep with the three questions discussed in subsection, SBM includes three processes: extraction (sample), deposit, and retrieve. Figure 2 illustrates the working principle of SBM in detail.

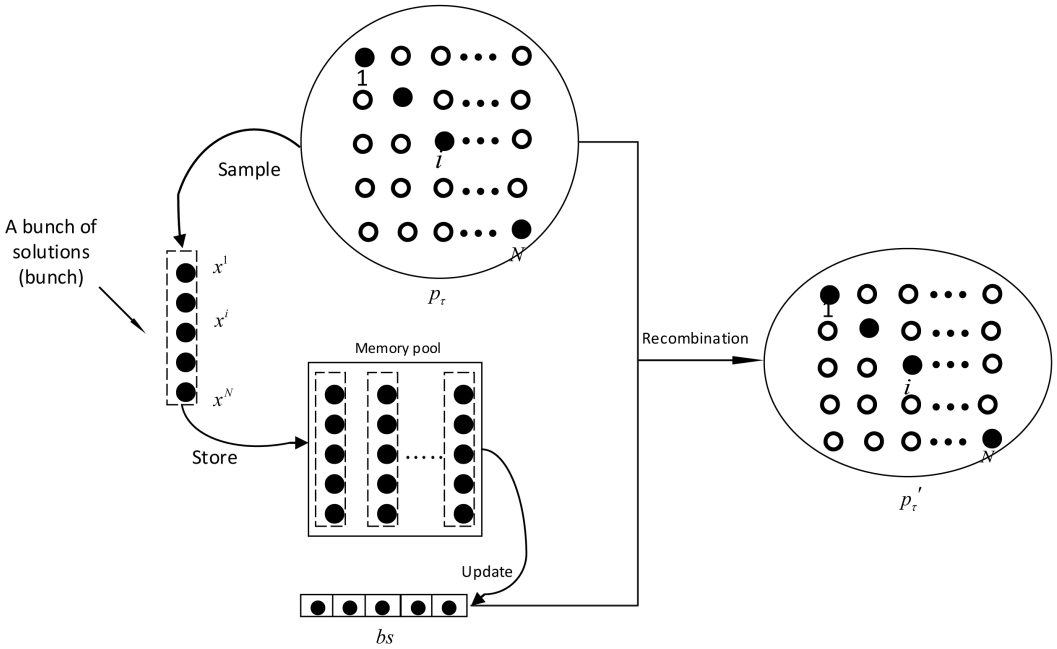

Figure 2. The working principle of the subproblem-based bunchy memory (SBM) method.

For question (1), the corresponding extraction process: SBM first selects *Bsize* even representative subproblems $\{i_1, ..., i_{Bsize}\}$ from the N subproblems, and then selects the optimal solutions of the corresponding individuals to be a string of individuals to be extracted.

For question (2), the corresponding deposit process: Insert the string of extracted individuals into the queue of the memory pool first. If the queue is full, it will be replaced according to the "first in, first out" strategy. Then, the memory pool is re-evaluated and updated in the new environment

(i.e., calculating the values of the object functions and the constraint functions). Finally, update the best individual array of $Bsize$, $bs[j](j = 1, ..., Bsize)$ is preserved as the best scalar objective function in the direction of λ^{i_j} (i rows and j columns of weight vector distribution) weight vector.

For question (3), the retrieval process: First, evaluate the individuals of p_τ in the new environment. Then, let $p'_\tau = p_\tau$. Finally, the optimal array bs competes with p_τ, which means for any subproblem i_j of the $Bsize$ subproblem $\{i_1, ..., i_{Bsize}\}$. If the scalar objective function of the individual $bs[j]$ is less than the scalar objective function value of the individual $p_\tau[i_j]$, set $p'_\tau[i_j] = bs[j]$, then the population is updated and the memory population p'_τ is obtained.

A detailed description can be seen in Algorithm 1.

Algorithm 1: SBM method

Input
  population P_τ; memory pool M;
Output
  population P'_τ;
Step 1 Extraction process:
  (1) A uniform selection of B_{size} representative sub-problems $\{i_1, ..., i_{B_{size}}\}$;
  (2) The current optimal solutions of these subproblems are saved as a *bunch* array.
Step 2 Deposit process:
  (1) Save the *bunch* to the end of the M queue; if the M is full, then the first string of the M queue is deleted;
  (2) Reevaluate the individual in M;
  (3) Initialize the optimal individual array bs: set bs as the first string of M, i.e., $bs = M(1)$;
  (4) Update bs: **for** $bunch = M(2)$ to the last string of M
    for $j = 1, ..., B_{size}$
      if $bunch[j].sof < bs[j].sof$ **do**
      $bs[j] = bunch[j]$;
      endif
    end
  end
Step 3 Retrieval process:
  (1) Reevaluate the individual of P_τ;
  (2) $P'_\tau = P_\tau$;
  (3) bs competes with P_τ: **for** $j = 1, ..., B_{size}$
    if $bs[j].sof < P_\tau[i_j].sof$ **do**
    endif
  end
    $P_\tau[i_j] = bs[j]$;
Step 4 Output P'_τ

The sample process of SBM is easy to extract a series of more evenly distributed individuals and maintain a better diversity in the objective space. For example, when N = 100 and $Bsize$ = 5, the sample process first selects 5 from 100 subproblems $\{i \,|i = 1, 25, 50, 75, 100\}$. Then, five solutions of $\{x^1, x^{25}, ..., x^{100}\}$ are extracted from population P_τ. These five solutions in the objective space are $\{F(x^1, t), F(x^{25}, t), ..., F(x^{100}, t)\}$. Because of the uniform distribution of the $Bsize$ representative problems, these five mapping points in the objective space are also evenly distributed.

3.4. Detailed Description of dMOEA/D-L_p and Its Time Complexity Analysis

Based on what we have discussed above, a detailed description of dMOEA/D-L_p can be concluded in Algorithm 2.

Algorithm 2: The overall framework of dMOEA/D-L_p

Input:
DMOP; Stop criterion;
N: The number of uniformly distributed weight vectors;
T: The number of neighbors;
τ_T: Frequency of environmental change;
n_T: Severity of environmental change.
Output:
Population P_τ;
Step 1 Initialization:
(1) population: iterative counter τ=0, $P_\tau = \{x^1, ..., x^n\}$;
(2) reference point: $z(t) = (z_1(t), ..., z_m(t))$;
for each $j = 1$ **to** m, **do**
$z_j(t) = \min_{1 \le i \le N} f_j(x^i, t)$;
end
(3) memory pool: $M = \phi$;
Step 2 Decomposition process:
(1) According to Equation (6), the DMOP is decomposed into N dynamic scalar optimization subproblems;
(2) Calculate Euclidean distance between any two weight vectors:
for each $i = 1$ **to** N, **do**
$B(i) = \{i_1, ..., i_T\}$, /*($\lambda^{i_1}, ..., \lambda^{i_T}$) are the T nearest weight vectors to the weight vector λ^i */;
end
Step 3 Detect and respond to environmental changes:
Detect the environmental changes as Equation (7):
if $\delta(t) > \tilde{\delta}$ **then**
Preserve and output the population P_τ before environmental change;
Apply Algorithm 1 to acquire new population $P_\tau{}'$, and set $P_\tau = P_\tau{}'$
endif
Step 4 Update of subproblems:
For each $i = 1$ **to** N **do**
$\bar{y} \leftarrow x^i + F(x^k, x^l)$ /* k and l are randomly selected from $B(i)$ */
$y_j \leftarrow \begin{cases} \bar{y}_j & if(rand(j) \le CR) \\ x_j & otherwise \end{cases} \quad j = 1, 2, ..., n;$
if $y = (y_1, y_2, ..., y_n)$ does not satisfy the constraints, **then** $y \leftarrow \text{repair}(y)$;
if $F(y, t) < F(x^i, t)$ **then** $F(x^i, t) = F(y, t)$;
For each $j = 1$ **to** m **do**
if $z_j(t) < f_j(y, t)$ **then** $z_j(t) = f_j(y, t)$;
end
For each $j \in B(i)$ **do**
if $g^{L_p}(y, t|\lambda^j, z(t)) < g^{L_p}(x^j, t|\lambda^j, z(t))$ **then**
$x^j = y$;
$FV^j(t) = F(y, t)$;
endif
end
end
Step 5 Stopping criterion:
if $\tau \ge G$ **then** stop and output P_τ,
otherwise $\tau = \tau + 1$ go to **Step3**.

The time cost of dMOEA/D-L_p is mainly focused on two steps: first detect and respond to environmental change, then optimize the sub-problems.

(1) Detection and response steps: the time complexity of the detection operation is $O(K)$, K is the number of individuals used in Formula (7), and the time complexity of responding environment change (i.e., calling SBM) is $O(N)$. Because $K < N$, the time complexity of this step is $O(N)$.
(2) The evolutionary optimization step of subproblems: because N subproblems are involved, and the neighborhood size of each subproblem is T, the time complexity of this step is $O(mNT)$, where m is the number of objectives.

Comprehensively analyzing the above two steps, the time complexity of dMOEA/D-L_p is $O(mNT)$, which is the same as of MOEA/D [23].

4. Experiments

In order to solve dynamic multi-objective optimization problems well, we propose dMOEA/D-L_p in this paper. For ease of expressions and comparisons, we call the version of dMOEA/D with the Tchebycheff decomposition approach as dMOEA/D-TCH, the version of dMOEA/D with the weighted sum decomposition approach as dMOEA/D-WS, the version of dMOEA/D with the PBI decomposition approach as dMOEA/D-PBI, and the version of dMOEA/D that uses L_p decomposition approach as dMOEA/D-L_p. There are two main sets of experimental studies in this section: (1) We compare dMOEA/D-L_p with dMOEA/D-TCH, dMOEA/D-WS, and dMOEA/D-PBI to test the effectiveness of the L_p decomposition method for dynamic multi-objective optimization; (2) We compare dMOEA/D-L_p with two other state-of-the-art algorithms (DNSGA-II [23], QICCA [7]) to test the overall performance of the algorithm.

4.1. Test Problems

We used FDA benchmark test instances [3,16] and their improved versions [30,32] as our test problems. Detailed definitions of these test problems are shown in Table 1. When the intensity of environmental change (n_T) is relatively small, the environmental change is less intense, and the number of POF(t) is less in each cycle. When the n_T value is large, the environmental change is more intense, and there will be many different POF(t) in each cycle.

Table 1. Test problems used in our experiments.

Problems	Objective Functions	Variable Bounds	n
FDA1	$f_1(x_1) = x_1, f_2(x) = g \cdot h$ $g(x_{II}) = 1 + \sum_{x_i \in x_{II}} (x_i - G(t))^2, h(f_1, g) = 1 - \sqrt{f_1/g}$ $G(t) = \sin(0.5\pi t), t = \lfloor \tau/\tau_T \rfloor / n_T$	$x_I = (x_1) \in [0,1]$ $x_{II} = (x_2, ..., x_n) \in [-1,1]$	20
FDA2	$f_1(x_1) = x_1, f_2(x) = g \cdot h$ $g(x_{II}) = 1 + \sum_{x_i \in x_{II}} x_i^2, h(x_{III} f_1, g) = 1 - (f_1/g)^{2^{\left(H(t) + \sum_{x_i \in x_{III}} (x_i - H(t)/4)^2\right)}}$ $H(t) = 2\sin(0.5\pi t - 1), t = \lfloor \tau/\tau_T \rfloor / n_T$	$x_I = (x_1) \in [0,1]$ $x_{II} = (x_2, ..., x_6) \in [-1,1]$ $x_{III} = (x_7, ..., x_n) \in [-1,1]$	20
FDA3	$f_1(x_1) = x_1^{F(t)}, f_2(x) = g \cdot h$ $g(x_{II}) = 1 + G(t) + \sum_{x_i \in x_{II}} (x_i - G(t))^2, h(f_1, g) = 1 - \sqrt{f_1/g}$ $G(t) = \lvert\sin(0.5\pi t)\rvert, F(t) = 10^{2\sin(0.5\pi t)}, t = \lfloor \tau/\tau_T \rfloor / n_T$	$x_I = (x_1) \in [0,1]$ $x_{II} = (x_2, ..., x_n) \in [-1,1]$	30
FDA4	$f_1(x) = (1 + g(x_{II})) \prod_{i=1}^{m-1} \cos(0.5\pi t x_i)$ $f_k(x) = (1 + g(x_{II})) \left(\prod_{i=1}^{m-k} \cos(0.5\pi t x_i)\right) \sin(0.5\pi t x_{m-k+1}), k = 2 : m-1$ $f_m(x) = (1 + g(x_{II})) \sin(0.5\pi t x_1)$ $g(x_{II}) = \sum_{x_i \in x_{II}} (x_i - G(t))^2, G(t) = \lvert\sin(0.5\pi t)\rvert, t = \lfloor \tau/\tau_T \rfloor / n_T$	$x_{II} = (x_m, ..., x_n)$ $x_i \in [0,1], i = 1 : n$	12

Table 1. *Cont.*

Problems	Objective Functions	Variable Bounds	*n*
FDA5	$f_1(x) = (1+g(x_{II}))\prod_{i=1}^{m-1}\cos(0.5\pi t y_i)$ $f_k(x) = (1+g(x_{II}))\left(\prod_{i=1}^{m-k}\cos(0.5\pi t y_i)\right)\sin(0.5\pi t y_{m-k+1}), k = 2:m-1$ $f_m(x) = (1+g(x_{II}))\sin(0.5\pi y_1)$ $g(x_{II}) = G(t) + \sum_{x_i \in x_{II}}(x_i - G(t))^2, G(t) = \lvert\sin(0.5\pi t)\rvert$ $y_i = x_i^{F(t)}, i = 1,..,(m-1), F(t) = 1+100\sin^4(0.5\pi t)$ $t = \lfloor \tau/\tau_T \rfloor / n_T$	$x_{II} = (x_m,...,x_n)$ $x_i \in [0,1], i = 1:n$	12

4.2. Performance Metric

The dynamic inverse generation distance rGD(t) [33] and dynamic generation distance GD(t) [34,35] are used as the performance evaluation metrics of the DEMOAs, and are defined as follows, respectively:

$$\mathrm{rGD}(t) = \frac{\sum_{i=1}^{|POF^*(t)|} d_i}{|POF^*(t)|}, d_i = \min_{k=1}^{|Q(t)|} \sqrt{\sum_{j=1}^{m} \left(f_j^{*(i)} - f_j^{(k)}\right)^2}, \tag{8}$$

$$\mathrm{GD}(t) = \frac{\sqrt{\sum_{i=1}^{|Q(t)|} (d_i')^2}}{|Q(t)|}, d_i' = \min_{k=1}^{|POF^*(t)|} \sqrt{\sum_{j=1}^{m} \left(f_j^{(i)} - f_j^{*(k)}\right)^2}, \tag{9}$$

where $POF^*(t)$ is a set of uniformly distributed points in the objective space along the POF at time t, $Q(t)$ is an approximation set to the POF obtained by the algorithm at the same time t, $f_j^{*(i)}$ is the j-th objective function value of the i-th sampling point in $POF^*(t)$, and $f_j^{(i)}$ is the j-th objective function value of the i-th point in $Q(t)$.

Moreover, d_i is the minimum Euclidean distance between point i of $POF^*(t)$ and the points in $Q(t)$. In a sense, if the $|POF^*(t)|$ is sufficiently large to represent the $POF^*(t)$ very well, rGD(t) can measure both the diversity and convergence of $Q(t)$. To have a low value of rGD(t), $Q(t)$ must be very close to the $POF^*(t)$, and cannot miss any part of the $POF^*(t)$. Therefore, the value of rGD(t) is smaller, which means that the convergence and diversity of the obtained Pareto front is better.

d_i' is the minimum Euclidean distance between point i of $Q(t)$ and the points in $|POF^*(t)|$. GD(t) measures the distance of $Q(t)$ to $POF^*(t)$. The value of GD(t) is smaller, which means that the convergence and diversity of the obtained Pareto front is better. The perfect situation is that GD(t) = 0, which means that all solutions in $Q(t)$ are Pareto optimal solutions.

4.3. Setting of Experimental Parameters

Individuals used real vector encoding in all test problems. Differential evolution (DE) crossover and polynomial mutation operators [30] are adopted in the five compared algorithms (i.e., dMOEAD-L_p, dMOEA/D-TCH, dMOEA/D-WS, dMOEA/D-PBI and DNSGA-II) [16], where crossover rate $CR = 0.9$, scaling factor $F = 0.5$. The whole clone (clone ratio of 5) and non-uniform mutation is adopted in QICCA [7]; diversified introduction ratio (ζ) was set to 0.2 in DNSGA-II [7]; memory pool size was set to 100. All of number of neighbors in four compared algorithms (dMOEAD-L_p, dMOEA/D-TCH, dMOEA/D-WS, and dMOEA/D-PBI) was set to 20; *Bsize* was set to 5 for 2-objective problems, and to 15 for 3-objective problems; the penalty parameter (θ) was set to 5 in dMOEA/D-PBI. For parameters used in the environmental monitoring operator, we took $k = 2$, $\tilde{\delta} = 0.002$ for problems with two objectives, and $k = 3$, $\tilde{\delta} = 0.006$ for 3-objective problems. Table 2 displays the common parameter settings for the compared algorithms, where n is the dimension of the decision variable. To study the effects of various dynamic changes in uncertain environments, different combinations of (τ_T, n_T) were set according to different DMOPs. Thirty independent runs were done for each combination of the various problems. In each experiment, various algorithms uniformly

tracked 100 times of environmental change. The number of generations was the combination of the environmental change time of each algorithm and the frequency of environmental change.

Table 2. Public parameter settings in the experiments.

Parameter	Value
Population size (N)	Two objectives: N = 100; Three objectives: N = 300
Crossover probability	0.9
Mutation probability	$1/N$
Frequency of change (τ_T)	5, 10, 15, 20, 25, 35
Severity of change (n_T)	5, 10

To choose a proper p for each test instance, we executed dMOEA/D-L_p on FDA1–FDA5 with different p values: $p = 0$, $p = 1/3$, $p = 1/2$, $p = 1$, $p = 2$. Experimental results are shown in Table 3. The best performance is highlighted with bold font. It can be seen from the table that $p = 2$ generally performEd the best for dMOEA/D-L_p in dealing with the FDA series problems. This may be because the POFs curvature of FDA1–FDA5 are quadratic, and $p = 2$ in the L_p function was the most suitable according to the geometric shape of the L_p function in Figure 1.

Table 3. Verification of p value in the L_p decomposition method for the FDA series problem. The best performance is highlighted with bold font. rGD(t): dynamic inverse generation distance.

Problem	(τ_T,n_T)	Statistic	rGD(t)				
			$p = \infty$	$p = 1/2$	$p = 1/3$	$p = 1$	$p = 2$
FDA1(2)	(25,5)	Min	$\mathbf{(6.03 \times 10^{-1})}$	(8.30×10^{-1})	(8.30×10^{-1})	(1.98×10^{0})	(6.75×10^{-1})
		Mean	(9.81×10^{-1})	(1.07×10^{0})	(1.07×10^{0})	(2.02×10^{0})	$\mathbf{(1.06 \times 10^{-1})}$
		Std	(2.55×10^{-1})	(8.98×10^{-2})	(8.98×10^{-2})	(8.11×10^{-2})	$\mathbf{(2.41 \times 10^{-2})}$
FDA2(2)	(15,5)	Min	(1.11×10^{-1})	(9.33×10^{-1})	(9.33×10^{-1})	(2.10×10^{0})	$\mathbf{(6.33 \times 10^{-2})}$
		Mean	(1.83×10^{-1})	(1.56×10^{0})	(1.56×10^{0})	(2.91×10^{0})	$\mathbf{(7.34 \times 10^{-2})}$
		Std	(7.47×10^{-2})	(5.30×10^{-1})	(5.30×10^{-1})	(4.20×10^{-1})	$\mathbf{(5.04 \times 10^{-3})}$
FDA3(2)	(35,5)	Min	$\mathbf{(4.13 \times 10^{-1})}$	(6.94×10^{-1})	(6.94×10^{-1})	(5.01×10^{-1})	(4.79×10^{-1})
		Mean	$\mathbf{(4.60 \times 10^{-1})}$	(3.33×10^{0})	(3.33×10^{0})	(6.23×10^{-1})	(5.16×10^{-1})
		Std	(1.84×10^{-1})	(1.34×10^{0})	(1.34×10^{0})	(8.71×10^{-2})	$\mathbf{(2.73 \times 10^{-2})}$
FDA4(3)	(25,5)	Min	(2.02×10^{-1})	(8.31×10^{0})	(8.31×10^{0})	(8.91×10^{-1})	$\mathbf{(1.30 \times 10^{-1})}$
		Mean	(4.64×10^{-1})	(8.36×10^{0})	(8.36×10^{0})	(1.03×10^{0})	$\mathbf{(1.35 \times 10^{-1})}$
		Std	(2.64×10^{-2})	(2.71×10^{-2})	(2.71×10^{-2})	(6.38×10^{-2})	$\mathbf{(2.51 \times 10^{-2})}$
FDA5(3)	(25,5)	Min	(8.43×10^{-2})	(8.32×10^{0})	(8.32×10^{0})	(1.01×10^{-1})	$\mathbf{(5.01 \times 10^{-2})}$
		Mean	(1.02×10^{-1})	$(1.26 \times 10^{+1})$	$(1.26 \times 10^{+1})$	(1.69×10^{-1})	$\mathbf{(5.69 \times 10^{-2})}$
		Std	(8.43×10^{-4})	(2.37×10^{0})	(2.37×10^{0})	(1.86×10^{-2})	$\mathbf{(3.05 \times 10^{-4})}$

4.4. Experimental Results and Analysis

In this subsection, we carry out the two parts of the experimental results and analyses. In the experiments, the statistics of the performance metrics were based on 30 independent runs.

(1) Compare dMOEA/D-L_p with dMOEA/D-WS, dMOEA/D-TCH, and dMOEA/D-PBI

To test the effectiveness of the L_p decomposition method, dMOEA/D-L_p was compared with dMOEA/D-WS, dMOEA/D-TCH, and dMOEA/D-PBI on the FDA series problems. Except for the decomposition method, all the other operators were the same for all four compared algorithms. The final rGD(t)-metric statistical results are shown in Table 4, and the final GD(t)-metric statistical results are shown in Table 5. Each table includes the best, mean, and standard deviation (std) of the performance metric values, and the best performance of the metric value is highlighted in bold. As shown in Table 4, in terms of the rGD(t)-metric, we can see that dMOEA/D-L_p achieved the best

performance out of these four algorithms. In terms of the GD(t)-metric, the same situations occur in Table 5. The best and the mean metric values in Tables 4 and 5 are mainly focused on dMOEA/D-L_p, indicating that dMOEA/D-L_p behaved the best in almost all the test instances. The minimal std values obtained by dMOEA/D-L_p presented the best stability of the algorithm. We can conclude that the L_p decomposition method is effective and has some advantages over TCH, WS, and PBI decomposition approaches for decomposition-based dynamic multi-objective optimization.

In the L_p decomposition method, $p \in (0, \infty]$, when $p = 1$, g^{L_p} becomes the weighted sum method, and when $p = \infty$, g^{L_p} becomes the standard Tchebycheff function. Figure 1 draws the contour lines of the L_p function with different p values showing that the search ability of L_p function decreased with the increase of p. All the g^{L_p} functions suffered from the geometry issue except for the Tchebycheff function. Especially, a g^{L_p} function cannot identify the whole Pareto optimal front of an MOP when the curvature of the Pareto optimal front is larger than the curvature of the contour line of the chosen g^{L_p} function. Since the curvature of the Tchebycheff function is ∞, it is able to identify Pareto optimal solutions for any type of geometry. Overall, for a g^{L_p} scalar function, its search ability and its robustness on problem geometries is a trade-off. The higher the search ability, the lower the robustness. If we can estimate the curvature of the Pareto optimal front in advance, then we can easily select a suitable g^{L_p} function based on its search ability. Based on our trial experiments on p for the FDA series problem in Section 4.3, p was set to 2 in advance for dMOEA/D-L_p in all experiments, which is consistent with the observation that if the curvature of the Pareto optimal front is quadratic, the most suitable g^{L_p} function should be the one using $p = 2$ [36]. Alternatively, for a new problem or a problem having a complex geometry, the use of the Tchebycheff function is a good choice.

(2) Compare dMOEA/D-L_p with DNSGA-II [16] and QICCA [7]

In order to test the performance of the proposed dMOEA/D-L_p, we tested dMOEA/D-L_p on each test instance, and compared the results obtained by dMOEA/D-L_p with those obtained by the other two state-of-the-art algorithms: DNSGA-II [16] and QICCA [7]. Referring to the literature with open source codes of NSGA-II [21] and MOEA/D [23], we realized the DNSGA-II, QICCA, and dMOEAD-L_p with C++ language programming. The final rGD(t)-metric statistical results are shown in Table 6, and the final GD(t)-metric statistical results are shown in Table 7. Each table includes the best, mean, and standard deviation (std) of the performance metric values, and the best performance of each metric value is marked in bold. For intuitive analysis of the performance of three algorithms, Figures 3–7 describe the approximation set to the Pareto front in different times. As shown in Tables 6 and 7, in terms of the rGD and GD-metric, dMOEA/D-L_p achieved the best performance in almost all of the test instances. Compared with DNSGA-II and QICCA, dMOEA/D-L_p was competitive in solving this kind of dynamic multi-objective optimization problem.

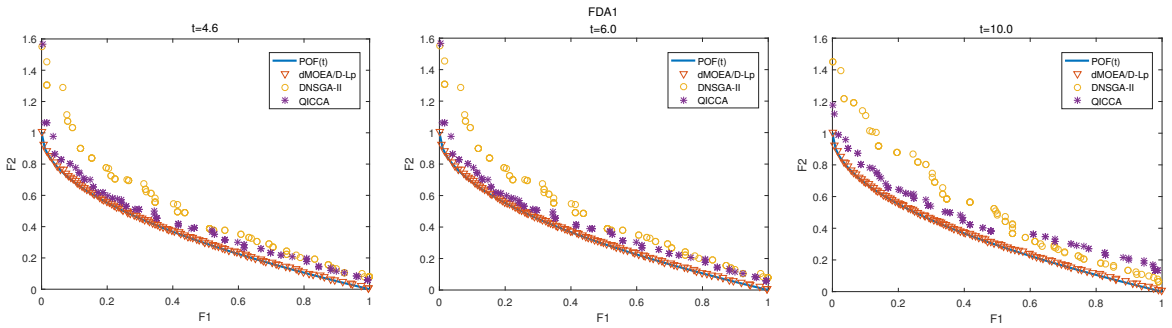

Figure 3. Solution sets obtained by three algorithms at different times on FDA1 in case of $\tau_T = 10$, $n_T = 10$. dMOEA/D-L_p: memory-enhanced dynamic multi-objective evolutionary algorithm based on L_p decomposition; DNSGA-II: dynamic non-dominated sorting genetic algorithm II; POF: Pareto optimal front; QICCA: immune clonal coevolutionary algorithm for dynamic multi-objective optimization.

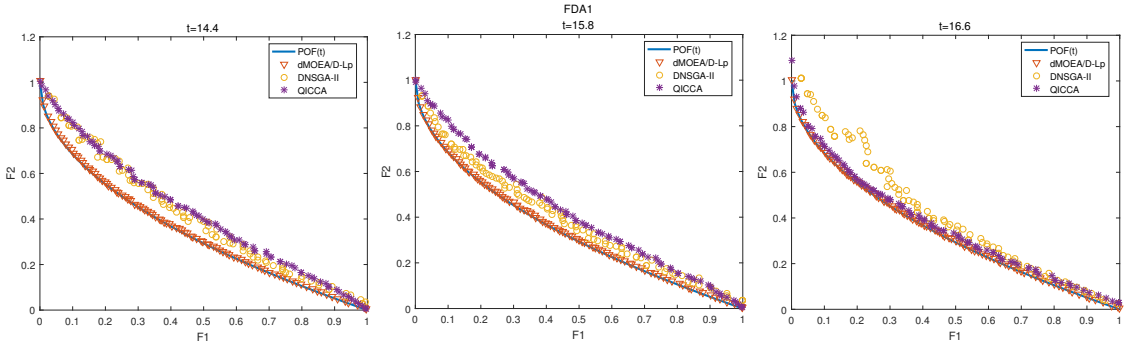

Figure 4. Solution sets obtained by three algorithms at different times on FDA1 in case of $\tau_T = 25$, $n_T = 5$.

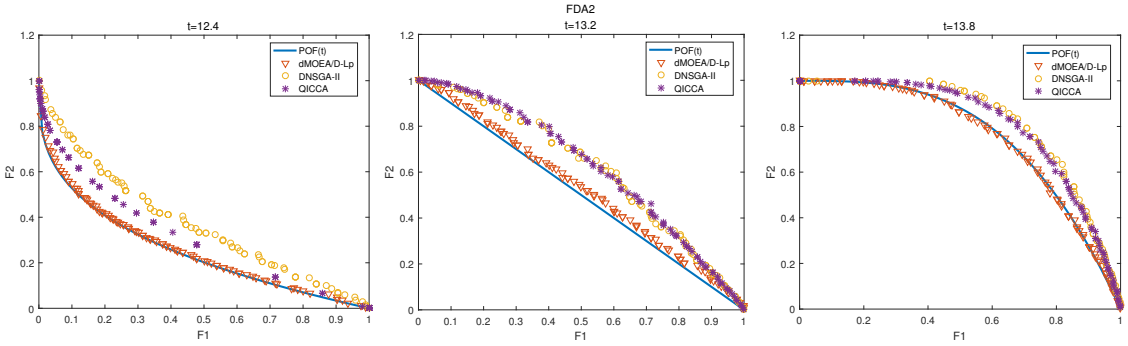

Figure 5. Solution sets obtained by three algorithms at different times on FDA2 in case of $\tau_T = 15$, $n_T = 5$.

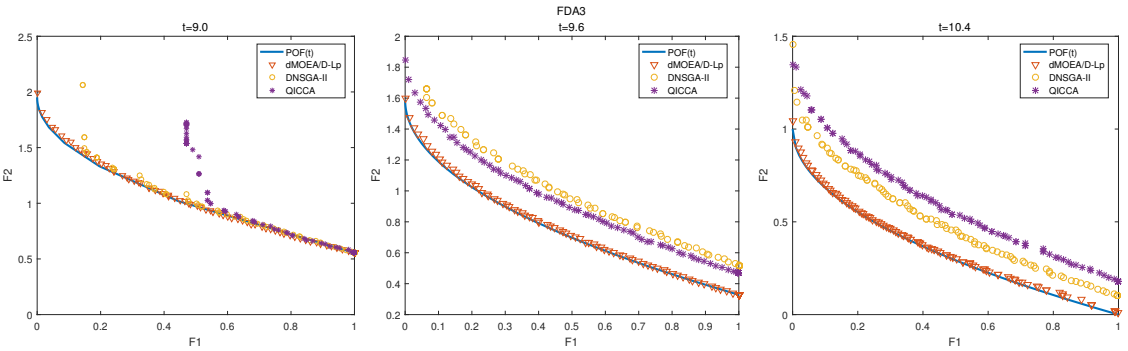

Figure 6. Solution sets obtained by three algorithms at different times on FDA3 in case of $\tau_T = 35$, $n_T = 5$.

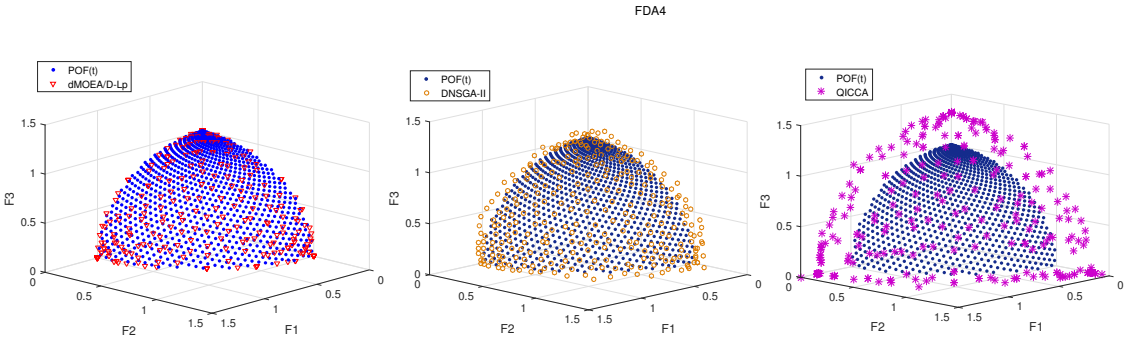

Figure 7. Solution sets obtained by three algorithms at $t = 13.2$ on FDA4 in case of $\tau_T = 25$, $n_T = 5$.

Table 4. The best, mean, and standard deviation of rGD(t) metric values obtained by dMOEA/D-L_p and the other versions of dMOEA/D on test problems based on 30 independent runs. The best performance of each metric value is marked in bold. dMOEA/D-PBI: dMOEA/D with the PBI decomposition approach; dMOEA/D-TCH: dMOEA/D with the Tchebycheff decomposition approach; dMOEA/D-WS: dMOEA/D with the weighted sum decomposition approach.

Problem	(τ_T,n_T)	Statistic	rGD(t) Metrics			
			dMOEA/D-L_p	dMOEA/D-TCH	dMOEA/D-WS	dMOEA/D-PBI
FDA1(2)	(10, 10)	Min	(6.69×10^{-1})	$\mathbf{(5.76 \times 10^{-1})}$	(1.94×10^{0})	(8.01×10^{-1})
		Mean	$\mathbf{(1.05 \times 10^{0})}$	(1.12×10^{0})	(2.05×10^{0})	(1.16×10^{0})
		Std	$\mathbf{(2.39 \times 10^{-2})}$	(3.20×10^{-1})	(1.27×10^{-1})	(5.13×10^{-1})
	(25,10)	Min	$\mathbf{(6.75 \times 10^{-1})}$	(7.33×10^{-1})	(1.99×10^{0})	(8.01×10^{-1})
		Mean	$\mathbf{(7.27 \times 10^{-1})}$	(1.96×10^{0})	(2.01×10^{0})	(8.74×10^{-1})
		Std	$\mathbf{(2.03 \times 10^{-2})}$	(2.32×10^{-1})	(3.49×10^{-2})	(2.11×10^{-1})
	(25,5)	Min	(6.75×10^{-1})	$\mathbf{(6.03 \times 10^{-1})}$	(1.98×10^{0})	(8.85×10^{-1})
		Mean	$\mathbf{(1.06 \times 10^{-1})}$	(9.81×10^{-1})	(2.02×10^{0})	(9.13×10^{-1})
		Std	$\mathbf{(2.41 \times 10^{-2})}$	(2.55×10^{-1})	(8.11×10^{-2})	(2.32×10^{-1})
FDA2(2)	(5,10)	Min	$\mathbf{(2.25 \times 10^{-1})}$	(1.79×10^{0})	(1.05×10^{0})	(2.33×10^{-1})
		Mean	$\mathbf{(3.32 \times 10^{-1})}$	(2.75×10^{0})	(2.44×10^{0})	(1.13×10^{0})
		Std	$\mathbf{(5.78 \times 10^{-2})}$	(6.91×10^{0})	(6.01×10^{-1})	(5.25×10^{-1})
	(15,10)	Min	$\mathbf{(1.21 \times 10^{-1})}$	(3.49×10^{-1})	(1.77×10^{0})	(2.33×10^{-1})
		Mean	$\mathbf{(1.30 \times 10^{-1})}$	(7.33×10^{-1})	(2.74×10^{0})	(1.13×10^{0})
		Std	$\mathbf{(5.71 \times 10^{-2})}$	(5.69×10^{-1})	(6.92×10^{-1})	(5.25×10^{-1})
	(15,5)	Min	$\mathbf{(6.33 \times 10^{-2})}$	(1.11×10^{-1})	(2.10×10^{0})	(1.00×10^{0})
		Mean	$\mathbf{(7.34 \times 10^{-2})}$	(1.83×10^{-1})	(2.91×10^{0})	(1.77×10^{0})
		Std	$\mathbf{(5.04 \times 10^{-3})}$	(7.47×10^{-2})	(4.20×10^{-1})	(7.14×10^{-1})
FDA3(2)	(25,10)	Min	$\mathbf{(1.00 \times 10^{-1})}$	(2.64×10^{-1})	(5.74×10^{-1})	(1.51×10^{-1})
		Mean	$\mathbf{(1.03 \times 10^{-1})}$	(4.03×10^{-1})	(6.56×10^{-1})	(5.15×10^{-1})
		Std	$\mathbf{(2.05 \times 10^{-3})}$	(2.38×10^{-3})	(2.10×10^{-2})	(2.54×10^{-1})
	(35,10)	Min	(4.04×10^{-1})	(6.06×10^{-1})	(6.13×10^{-2})	$\mathbf{(2.17 \times 10^{-1})}$
		Mean	$\mathbf{(6.05 \times 10^{-1})}$	(7.56×10^{-1})	(7.23×10^{-2})	(6.06×10^{-1})
		Std	(2.73×10^{-2})	$\mathbf{(2.05 \times 10^{-2})}$	(1.34×10^{-1})	(2.31×10^{-1})
	(35,5)	Min	(4.79×10^{-1})	(4.13×10^{-1})	(5.01×10^{-1})	$\mathbf{(1.35 \times 10^{-1})}$
		Mean	(5.16×10^{-1})	$\mathbf{(4.60 \times 10^{-1})}$	(6.23×10^{-1})	(5.33×10^{-1})
		Std	$\mathbf{(2.73 \times 10^{-2})}$	(1.84×10^{-1})	(8.71×10^{-2})	(2.08×10^{-1})
FDA4(3)	(20,10)	Min	$\mathbf{(1.29 \times 10^{-2})}$	(7.01×10^{-1})	(1.96×10^{0})	(4.05×10^{-1})
		Mean	$\mathbf{(2.43 \times 10^{-2})}$	(8.00×10^{-1})	(2.74×10^{0})	(4.14×10^{-1})
		Std	$\mathbf{(5.67 \times 10^{-3})}$	(2.34×10^{-1})	(6.97×10^{-1})	(1.13×10^{-1})
	(25,10)	Min	(1.26×10^{-1})	(7.01×10^{-1})	$\mathbf{(1.00 \times 10^{-1})}$	(1.01×10^{-1})
		Mean	$\mathbf{(1.40 \times 10^{-1})}$	(8.00×10^{-1})	(2.98×10^{-1})	(3.87×10^{-1})
		Std	$\mathbf{(3.36 \times 10^{-2})}$	(2.34×10^{-1})	(6.30×10^{-1})	(3.20×10^{-1})
	(25,5)	Min	(1.30×10^{-1})	(2.02×10^{-1})	(8.91×10^{-1})	$\mathbf{(1.14 \times 10^{-1})}$
		Mean	$\mathbf{(1.35 \times 10^{-1})}$	(4.64×10^{-1})	(1.03×10^{0})	(2.32×10^{-1})
		Std	$\mathbf{(2.51 \times 10^{-2})}$	(2.64×10^{-2})	(6.38×10^{-2})	(1.03×10^{-1})
FDA5(3)	(20,10)	Min	$\mathbf{(8.22 \times 10^{-2})}$	(7.37×10^{-1})	(6.03×10^{-1})	(7.79×10^{-1})
		Mean	$\mathbf{(9.05 \times 10^{-2})}$	(9.06×10^{-1})	(8.96×10^{-1})	(9.16×10^{-1})
		Std	$\mathbf{(8.43 \times 10^{-4})}$	(2.76×10^{-3})	(1.09×10^{-3})	(1.75×10^{-3})
	(25,10)	Min	$\mathbf{(7.83 \times 10^{-2})}$	(8.21×10^{-2})	(8.43×10^{-2})	(8.03×10^{-2})
		Mean	(8.86×10^{-2})	$\mathbf{(8.35 \times 10^{-2})}$	(8.65×10^{-2})	(8.43×10^{-2})
		Std	$\mathbf{(9.75 \times 10^{-5})}$	(1.07×10^{-4})	(1.09×10^{-3})	(1.10×10^{-4})
	(25,5)	Min	$\mathbf{(5.01 \times 10^{-2})}$	(8.43×10^{-2})	(1.01×10^{-1})	(8.01×10^{-2})
		Mean	$\mathbf{(5.69 \times 10^{-2})}$	(1.02×10^{-1})	(1.69×10^{-1})	(1.05×10^{-1})
		Std	$\mathbf{(3.05 \times 10^{-4})}$	(8.43×10^{-4})	(1.86×10^{-2})	(6.09×10^{-2})

Table 5. The best, mean, and standard deviation of GD(t) metric values obtained by dMOEA/D-L_p and the other versions of dMOEA/D on test problems based on 30 independent runs. The best performance of each metric value is marked in bold.

Problem	(τ_T,n_T)	Statistic	GD(t) Metrics			
			dMOEA/D-L_p	dMOEA/D-TCH	dMOEA/D-WS	dMOEA/D-PBI
FDA1(2)	(10,10)	Min	$\mathbf{(5.55 \times 10^{-4})}$	(5.96×10^{-4})	(7.74×10^{-4})	(9.72×10^{-4})
		Mean	$\mathbf{(9.14 \times 10^{-4})}$	(6.45×10^{-3})	(9.61×10^{-3})	(1.01×10^{-2})
		Std	$\mathbf{(4.79 \times 10^{-4})}$	(4.45×10^{-3})	(1.77×10^{-2})	(6.49×10^{-3})
	(25,10)	Min	$\mathbf{(4.18 \times 10^{-4})}$	(4.92×10^{-4})	(1.20×10^{-3})	(6.71×10^{-4})
		Mean	$\mathbf{(1.26 \times 10^{-3})}$	(1.62×10^{-3})	(1.66×10^{-3})	(3.00×10^{-3})
		Std	$\mathbf{(5.18 \times 10^{-4})}$	(6.00×10^{-4})	(1.49×10^{-3})	(1.24×10^{-3})
	(25,5)	Min	$\mathbf{(5.97 \times 10^{-4})}$	(6.26×10^{-4})	(1.12×10^{-3})	(7.64×10^{-4})
		Mean	$\mathbf{(9.13 \times 10^{-4})}$	(2.26×10^{-3})	(3.35×10^{-3})	(4.12×10^{-3})
		Std	$\mathbf{(4.79 \times 10^{-4})}$	(1.29×10^{-3})	(6.37×10^{-3})	(1.86×10^{-3})
FDA2(2)	(5,10)	Min	$\mathbf{(7.89 \times 10^{-20})}$	(1.20×10^{-6})	(3.16×10^{-18})	(8.04×10^{-4})
		Mean	$\mathbf{(5.10 \times 10^{-5})}$	(3.31×10^{-3})	(1.55×10^{-4})	(5.79×10^{-3})
		Std	(4.92×10^{-3})	(2.54×10^{-3})	$\mathbf{(1.26 \times 10^{-3})}$	(3.43×10^{-3})
	(15,10)	Min	$\mathbf{(6.16 \times 10^{-4})}$	(6.70×10^{-4})	(1.02×10^{-3})	(6.49×10^{-4})
		Mean	(2.43×10^{-3})	(2.46×10^{-3})	$\mathbf{(1.32 \times 10^{-3})}$	(2.57×10^{-3})
		Std	$\mathbf{(4.92 \times 10^{-4})}$	(1.22×10^{-3})	(1.31×10^{-3})	(1.54×10^{-3})
	(15,5)	Min	(7.56×10^{-4})	(5.99×10^{-4})	$\mathbf{(6.02 \times 10^{-9})}$	(6.60×10^{-4})
		Mean	$\mathbf{(4.75 \times 10^{-3})}$	(1.91×10^{-2})	(7.66×10^{-3})	(1.98×10^{-2})
		Std	$\mathbf{(2.53 \times 10^{-3})}$	(1.40×10^{-2})	(1.28×10^{-2})	(1.36×10^{-2})
FDA3(2)	(25,10)	Min	$\mathbf{(1.00 \times 10^{-3})}$	(8.07×10^{-3})	(1.00×10^{-2})	(3.37×10^{-2})
		Mean	$\mathbf{(3.48 \times 10^{-2})}$	(4.92×10^{-2})	(2.20×10^{-1})	(3.71×10^{-2})
		Std	(3.91×10^{-2})	(7.34×10^{-2})	(1.08×10^{-1})	(4.17×10^{-2})
	(35,10)	Min	$\mathbf{(1.02 \times 10^{-3})}$	(7.56×10^{-1})	(5.90×10^{-3})	(3.23×10^{-3})
		Mean	$\mathbf{(3.15 \times 10^{-2})}$	(2.16×10^{0})	(6.89×10^{-2})	(3.36×10^{-2})
		Std	$\mathbf{(3.48 \times 10^{-2})}$	(1.06×10^{0})	(5.35×10^{-2})	(3.87×10^{-2})
	(35,5)	Min	$\mathbf{(1.06 \times 10^{-3})}$	(9.18×10^{-3})	(1.11×10^{-2})	(7.72×10^{-3})
		Mean	(3.64×10^{-2})	(3.90×10^{-2})	(6.67×10^{-2})	$\mathbf{(3.63 \times 10^{-2})}$
		Std	$\mathbf{(3.39 \times 10^{-2})}$	(3.89×10^{-2})	(5.53×10^{-2})	(3.48×10^{-2})
FDA4(3)	(20,10)	Min	$\mathbf{(2.13 \times 10^{-2})}$	(2.52×10^{-2})	(3.24×10^{-2})	(2.31×10^{-2})
		Mean	$\mathbf{(2.16 \times 10^{-2})}$	(2.56×10^{-2})	(4.64×10^{-2})	(2.56×10^{-2})
		Std	$\mathbf{(1.68 \times 10^{-4})}$	(8.54×10^{-4})	(2.48×10^{-2})	(2.04×10^{-4})
	(25,10)	Min	(2.51×10^{-2})	$\mathbf{(1.52 \times 10^{-2})}$	(3.22×10^{-2})	(2.31×10^{-2})
		Mean	(2.55×10^{-2})	(2.59×10^{-2})	(5.18×10^{-2})	$\mathbf{(2.34 \times 10^{-2})}$
		Std	$\mathbf{(1.48 \times 10^{-4})}$	(3.08×10^{-3})	(2.53×10^{-2})	(1.64×10^{-4})
	(25,5)	Min	$\mathbf{(1.12 \times 10^{-2})}$	(2.53×10^{-2})	(3.21×10^{-2})	(2.31×10^{-2})
		Mean	$\mathbf{(1.47 \times 10^{-2})}$	(2.56×10^{-2})	(5.33×10^{-2})	(2.34×10^{-2})
		Std	$\mathbf{(6.04 \times 10^{-5})}$	(1.07×10^{-4})	(3.62×10^{-2})	(1.64×10^{-4})
FDA5(3)	(20,10)	Min	(2.69×10^{-2})	$\mathbf{(2.20 \times 10^{-2})}$	(3.23×10^{-2})	(5.76×10^{-2})
		Mean	(4.26×10^{-2})	(3.99×10^{-2})	(7.14×10^{-2})	$\mathbf{(3.25 \times 10^{-2})}$
		Std	$\mathbf{(8.35 \times 10^{-3})}$	(1.16×10^{-2})	(3.11×10^{-2})	(7.09×10^{-2})
	(25,10)	Min	(2.68×10^{-2})	$\mathbf{(1.96 \times 10^{-2})}$	(3.42×10^{-2})	(2.56×10^{-2})
		Mean	(4.23×10^{-2})	(3.62×10^{-2})	(6.38×10^{-2})	$\mathbf{(3.83 \times 10^{-2})}$
		Std	$\mathbf{(8.30 \times 10^{-4})}$	(7.29×10^{-3})	(2.28×10^{-3})	(1.81×10^{-2})
	(25,5)	Min	(2.86×10^{-2})	$\mathbf{(2.22 \times 10^{-2})}$	(3.26×10^{-2})	(2.82×10^{-2})
		Mean	(4.21×10^{-2})	(4.19×10^{-2})	(9.34×10^{-2})	$\mathbf{(3.89 \times 10^{-2})}$
		Std	$\mathbf{(7.85 \times 10^{-4})}$	(8.84×10^{-4})	(5.04×10^{-2})	(8.11×10^{-3})

Table 6. The best, mean, and standard deviation of rGD(t) metric values obtained by dMOEA/D-L_p and the other two compared algorithms on test problems based on 30 independent runs. The best performance of each metric value is marked in bold.

Problem	(τ_T,n_T)	Statistic	rGD(t) Metrics		
			dMOEA/D-L_p	DNSGA-II	QICCA
FDA1(2)	(10,10)	Min	$\mathbf{(6.69 \times 10^{-1})}$	(1.115×10^{0})	(1.05×10^{0})
		Mean	$\mathbf{(1.05 \times 10^{0})}$	(1.21×10^{0})	(1.08×10^{0})
		Std	$\mathbf{(2.39 \times 10^{-2})}$	(3.11×10^{-1})	(2.45×10^{-1})
	(25,10)	Min	$\mathbf{(6.75 \times 10^{-1})}$	(1.07×10^{0})	(1.05×10^{0})
		Mean	$\mathbf{(7.27 \times 10^{-1})}$	(1.08×10^{0})	(1.06×10^{0})
		Std	$\mathbf{(2.03 \times 10^{-2})}$	(5.48×10^{-2})	(4.43×10^{-2})
	(25,5)	Min	$\mathbf{(6.75 \times 10^{-1})}$	(1.06×10^{0})	(1.06×10^{0})
		Mean	$\mathbf{(1.06 \times 10^{0})}$	(1.09×10^{0})	(1.07×10^{0})
		Std	$\mathbf{(2.41 \times 10^{-2})}$	(1.36×10^{-1})	(7.44×10^{-2})
FDA2(2)	(5,10)	Min	$\mathbf{(2.25 \times 10^{-1})}$	(1.50×10^{0})	(1.07×10^{0})
		Mean	$\mathbf{(3.32 \times 10^{-1})}$	(3.65×10^{0})	(3.32×10^{0})
		Std	$\mathbf{(5.78 \times 10^{-2})}$	(7.53×10^{-1})	(7.11×10^{-1})
	(15,10)	Min	$\mathbf{(1.21 \times 10^{-1})}$	(1.34×10^{0})	(1.47×10^{0})
		Mean	$\mathbf{(1.30 \times 10^{-1})}$	(4.30×10^{0})	(3.95×10^{0})
		Std	$\mathbf{(5.71 \times 10^{-2})}$	(5.95×10^{-1})	(6.36×10^{-1})
	(15,5)	Min	$\mathbf{(6.33 \times 10^{-2})}$	(1.88×10^{0})	(1.40×10^{0})
		Mean	$\mathbf{(7.34 \times 10^{-2})}$	(3.75×10^{0})	(3.63×10^{0})
		Std	$\mathbf{(5.04 \times 10^{-3})}$	(6.33×10^{-1})	(6.95×10^{-1})
FDA3(2)	(25,10)	Min	$\mathbf{(1.00 \times 10^{-1})}$	(1.17×10^{-1})	(1.40×10^{-1})
		Mean	$\mathbf{(1.03 \times 10^{-1})}$	(6.36×10^{-1})	(6.40×10^{-1})
		Std	$\mathbf{(2.05 \times 10^{-3})}$	(2.61×10^{-2})	(2.63×10^{-2})
	(35,10)	Min	$\mathbf{(4.04 \times 10^{-1})}$	(6.30×10^{-1})	(6.35×10^{-1})
		Mean	$\mathbf{(6.05 \times 10^{-1})}$	(6.35×10^{-1})	(6.38×10^{-1})
		Std	(2.73×10^{-2})	$\mathbf{(2.67 \times 10^{-2})}$	(2.68×10^{-2})
	(35,5)	Min	$\mathbf{(4.79 \times 10^{-1})}$	(6.06×10^{-1})	(6.09×10^{-1})
		Mean	$\mathbf{(5.16 \times 10^{-1})}$	(6.10×10^{-1})	(6.11×10^{-1})
		Std	(2.73×10^{-2})	(2.70×10^{-2})	$\mathbf{(2.52 \times 10^{-2})}$
FDA4(3)	(20,10)	Min	$\mathbf{(2.43 \times 10^{-2})}$	(1.39×10^{-1})	(1.41×10^{0})
		Mean	$\mathbf{(1.29 \times 10^{-2})}$	(1.40×10^{-1})	(1.77×10^{0})
		Std	$\mathbf{(5.67 \times 10^{-3})}$	(5.19×10^{-1})	(3.13×10^{-1})
	(25,10)	Min	$\mathbf{(1.26 \times 10^{-1})}$	(1.27×10^{-1})	(1.38×10^{-1})
		Mean	$\mathbf{(1.40 \times 10^{-1})}$	(1.42×10^{-1})	(1.45×10^{-1})
		Std	$\mathbf{(3.36 \times 10^{-2})}$	(4.38×10^{-1})	(3.83×10^{-2})
	(25,5)	Min	$\mathbf{(1.30 \times 10^{-1})}$	(1.32×10^{-1})	(2.02×10^{-1})
		Mean	$\mathbf{(1.35 \times 10^{-1})}$	(1.38×10^{-1})	(2.29×10^{0})
		Std	$\mathbf{(2.51 \times 10^{-2})}$	(3.08×10^{-2})	(5.98×10^{-2})
FDA5(3)	(20,10)	Min	(8.22×10^{-2})	(7.31×10^{-1})	$\mathbf{(8.03 \times 10^{-2})}$
		Mean	$\mathbf{(9.05 \times 10^{-2})}$	(9.73×10^{-1})	(9.13×10^{-2})
		Std	$\mathbf{(8.43 \times 10^{-4})}$	(3.16×10^{-3})	(4.86×10^{-3})
	(25,10)	Min	(8.43×10^{-2})	$\mathbf{(6.36 \times 10^{-2})}$	(7.91×10^{-2})
		Mean	(9.16×10^{-2})	(8.65×10^{-2})	$\mathbf{(8.43 \times 10^{-2})}$
		Std	$\mathbf{(1.07 \times 10^{-4})}$	(1.55×10^{-4})	(1.10×10^{-4})
	(25,5)	Min	$\mathbf{(5.01 \times 10^{-2})}$	(8.00×10^{-2})	(9.38×10^{-2})
		Mean	$\mathbf{(5.69 \times 10^{-2})}$	(1.06×10^{-1})	(1.05×10^{-1})
		Std	$\mathbf{(3.05 \times 10^{-4})}$	(5.43×10^{-4})	(6.09×10^{-4})

Table 7. The best, mean, and standard deviation of GD(t) metric values obtained by dMOEA/D-L_p and the other two compared algorithms on test problems based on 30 independent runs. The best performance of each metric value is marked in bold.

Problem	(τ_T, n_T)	Statistic	GD(t) Metrics		
			dMOEA/D-L_p	DNSGA-II	QICCA
FDA1(2)	(10,10)	Min	$\mathbf{(5.55 \times 10^{-4})}$	(6.91×10^{-3})	(5.76×10^{-3})
		Mean	$\mathbf{(9.14 \times 10^{-4})}$	(8.08×10^{-3})	(6.49×10^{-3})
		Std	$\mathbf{(4.79 \times 10^{-4})}$	(7.00×10^{-3})	(6.08×10^{-2})
	(25,10)	Min	$\mathbf{(4.18 \times 10^{-4})}$	(1.81×10^{-3})	(1.93×10^{-3})
		Mean	$\mathbf{(1.26 \times 10^{-3})}$	(2.02×10^{-3})	(2.05×10^{-3})
		Std	$\mathbf{(5.18 \times 10^{-4})}$	(6.42×10^{-4})	(1.37×10^{-3})
	(25,5)	Min	$\mathbf{(5.97 \times 10^{-4})}$	(2.56×10^{-3})	(2.76×10^{-3})
		Mean	$\mathbf{(9.13 \times 10^{-4})}$	(2.97×10^{-3})	(2.92×10^{-3})
		Std	$\mathbf{(4.79 \times 10^{-4})}$	(3.75×10^{-3})	(2.56×10^{-3})
FDA2(2)	(5,10)	Min	(7.89×10^{-20})	$\mathbf{(1.52 \times 10^{-20})}$	(5.10×10^{-5})
		Mean	$\mathbf{(5.10 \times 10^{-5})}$	(1.50×10^{-3})	(2.51×10^{-3})
		Std	(4.92×10^{-3})	$\mathbf{(2.33 \times 10^{-3})}$	(2.97×10^{-3})
	(15,10)	Min	(6.16×10^{-4})	$\mathbf{(8.80 \times 10^{-5})}$	(9.11×10^{-5})
		Mean	$\mathbf{(2.43 \times 10^{-3})}$	(4.37×10^{-3})	(7.08×10^{-3})
		Std	$\mathbf{(4.92 \times 10^{-4})}$	(3.82×10^{-3})	(5.44×10^{-3})
	(15,5)	Min	(7.56×10^{-4})	$\mathbf{(1.02 \times 10^{-7})}$	(2.43×10^{-7})
		Mean	$\mathbf{(4.75 \times 10^{-3})}$	(7.55×10^{-3})	(1.80×10^{-2})
		Std	$\mathbf{(2.53 \times 10^{-3})}$	(5.70×10^{-3})	(1.29×10^{-2})
FDA3(2)	(25,10)	Min	$\mathbf{(1.00 \times 10^{-3})}$	(8.07×10^{-3})	(1.00×10^{-2})
		Mean	$\mathbf{(3.48 \times 10^{-2})}$	(4.92×10^{-2})	(2.20×10^{-1})
		Std	$\mathbf{(3.91 \times 10^{-2})}$	(7.34×10^{-2})	(1.08×10^{-1})
	(35,10)	Min	$\mathbf{(1.02 \times 10^{-3})}$	(3.02×10^{-2})	(3.40×10^{-2})
		Mean	$\mathbf{(3.15 \times 10^{-2})}$	(3.23×10^{-2})	(3.68×10^{-2})
		Std	$\mathbf{(3.48 \times 10^{-2})}$	(3.50×10^{-2})	(4.32×10^{-2})
	(35,5)	Min	$\mathbf{(1.06 \times 10^{-3})}$	(3.79×10^{-2})	(4.05×10^{-2})
		Mean	$\mathbf{(3.64 \times 10^{-2})}$	(3.92×10^{-2})	(4.27×10^{-2})
		Std	$\mathbf{(3.39 \times 10^{-2})}$	(3.42×10^{-2})	(4.55×10^{-2})
FDA4(3)	(20,10)	Min	$\mathbf{(2.13 \times 10^{-2})}$	(3.57×10^{-2})	(2.54×10^{-2})
		Mean	$\mathbf{(2.16 \times 10^{-2})}$	(3.88×10^{-2})	(2.66×10^{-2})
		Std	$\mathbf{(1.68 \times 10^{-4})}$	(2.03×10^{-4})	(1.83×10^{-4})
	(25,10)	Min	(2.51×10^{-2})	(2.56×10^{-2})	$\mathbf{(2.43 \times 10^{-2})}$
		Mean	$\mathbf{(2.55 \times 10^{-2})}$	(3.11×10^{-2})	(3.66×10^{-2})
		Std	$\mathbf{(1.48 \times 10^{-4})}$	(1.75×10^{-3})	(2.02×10^{-4})
	(25,5)	Min	$\mathbf{(1.12 \times 10^{-2})}$	(1.53×10^{-2})	(1.99×10^{-2})
		Mean	$\mathbf{(1.47 \times 10^{-2})}$	(1.67×10^{-2})	(2.47×10^{-2})
		Std	$\mathbf{(6.04 \times 10^{-5})}$	(1.44×10^{-4})	(1.68×10^{-4})
FDA5(3)	(20,10)	Min	(2.69×10^{-2})	$\mathbf{(2.20 \times 10^{-2})}$	(3.23×10^{-2})
		Mean	(4.26×10^{-2})	$\mathbf{(3.99 \times 10^{-2})}$	(7.14×10^{-2})
		Std	$\mathbf{(8.35 \times 10^{-3})}$	(1.16×10^{-2})	(3.11×10^{-2})
	(25,10)	Min	$\mathbf{(2.68 \times 10^{-2})}$	(3.44×10^{-2})	(3.67×10^{-2})
		Mean	$\mathbf{(4.23 \times 10^{-2})}$	(4.62×10^{-2})	$\mathbf{(4.87 \times 10^{-2})}$
		Std	$\mathbf{(8.30 \times 10^{-4})}$	(8.53×10^{-3})	(8.28×10^{-3})
	(25,5)	Min	(2.86×10^{-2})	(3.15×10^{-2})	$\mathbf{(2.22 \times 10^{-2})}$
		Mean	$\mathbf{(4.21 \times 10^{-2})}$	(4.36×10^{-2})	(3.38×10^{-1})
		Std	(7.85×10^{-4})	$\mathbf{(7.32 \times 10^{-4})}$	(1.04×10^{-2})

It can be seen from Table 1 that some components (x_{II}) of the Pareto optimal solutions in FDA1 varied sinusoidally with the function $G(t)$, so the $POS(t)$ of FDA1 changed with time. However, at any time t, $POF(t)$ is $F_2 = 1 - \sqrt{F_1}$ and did not change with time. Figure 3 shows the solution sets obtained by three algorithms for solving FDA1 at three different times in the tenth cycle. The corresponding environmental change factors in these three moments were $\tau_T = 10$, $n_T = 10$. Figure 4 describes the solution sets obtained by three algorithms, also for FDA1, under the parameters $\tau_T = 25$ and $n_T = 10$. From the two figures, it can be seen clearly that the solution sets obtained by dMOEA/D-L_p were the most uniformly convergent to $POF(t)$ in different environments. The results of the other two algorithms in mapping significantly deviated from $POF(t)$ and the distributions were uneven.

Some components (x_{III}) of the Pareto optimal solutions in FDA2 varied sinusoidally with the function $H(t)/4$, so the $POS(t)$ of the FDA2 changed with time. Furthermore, the $POF(t)$ of FDA2 also changed with time sinusoidally because $f_2 = 1 - (f_1/g)^{2^{H(t)}}$. Figure 5 shows the solution sets obtained by three algorithms for solving FDA2 at three different times in the tenth cycle. Environmental change factors were set to $\tau_T = 15$ and $n_T = 5$. It can be seen from Figure 5 that the Pareto front shape of the problem changed from convex to concave. The solutions sets obtained by dMOEA/D-L_p at three different times were closest to $POF(t)$, and the convergence distribution of dMOEA/D-L_p was better than the other two algorithms.

The $POS(t)$ of the FDA3 also changed with time. Its $POF(t)$ was $f_2 = (1 + G(t)) \times (1 - \sqrt{f_1/(1 + G(t))})]$, and also changed with $G(t)$. Moreover, the density of the $POF(t)$ also varied with time. Figure 6 shows the distributions of the solution sets obtained by three algorithms at three different times. From left to right, the corresponding environmental change amplitude of the problem in Figure 6 varied from small to large, and the distributions of $POF(t)$ shifted from top to bottom. All three plots of Figure 6 show that the distribution and convergence of dMOEA/D-L_p were the best.

Some components of FDA5's optimal variables (x_{II}) changed with $G(t)$, so its $POS(t)$ changed with time. When the number of objectives m was 3, the $POF(t)$ of FDA5 was 1/8 of a sphere at any time, but the radius of the sphere varied between 1 and 2 with time t. Figures 8–10 plot the solution sets obtained by three algorithms on FDA5 at three different times. The environmental change amplitude of the problem varied from large to small in the three times, and the spherical radius of the $POF(t)$ grew from 1 to 2. Figures 8–10 show that dMOEA/D-L_p could best track the $POF(t)$, and the convergence and distribution of the solution sets obtained by dMOEA/D-L_p were obviously superior to those obtained by the other two algorithms.

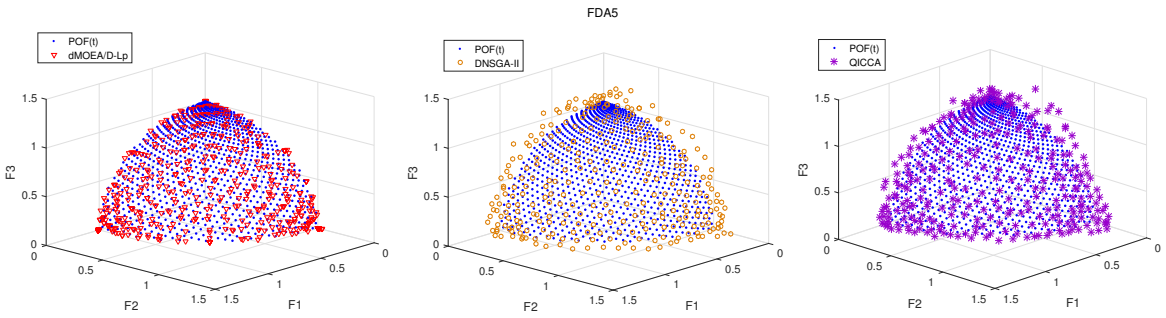

Figure 8. Solution sets obtained by three algorithms at t = 10.2 on FDA5 in case of $\tau_T = 15$, $n_T = 5$.

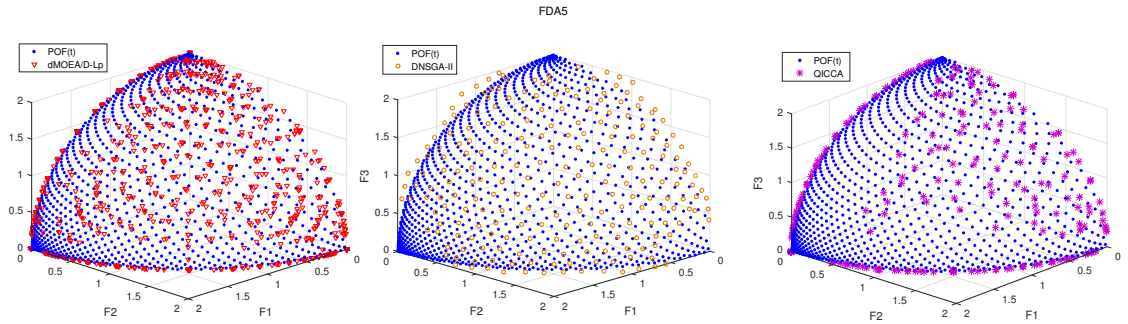

Figure 9. Solution sets obtained by three algorithms at t = 10.8 on FDA5 in case of $\tau_T = 15$, $n_T = 5$.

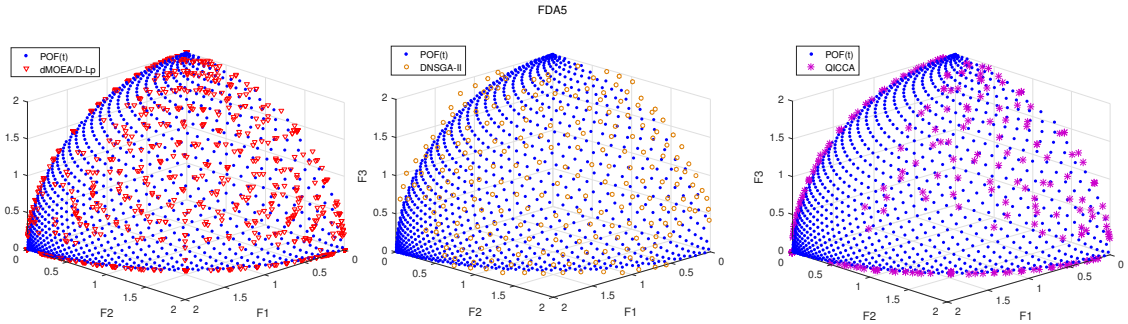

Figure 10. Solution sets obtained by three algorithms at t = 13.2 on FDA5 in case of $\tau_T = 15$, $n_T = 5$.

The $POS(t)$ and $POF(t)$ of FDA4 were similar to FDA5, except that the spherical radius of the $POF(t)$ remained the same value (one), not varying with time t. Similar comparing results happen for FDA4, which can be seen in Figure 7. It indicates that dMOEA/D-L_p performed with obvious superiority over the other two algorithms.

5. Conclusions

In order to handle dynamic multi-objective optimization problems well, a memory-enhanced dynamic multi-objective evolutionary algorithm based on L_p decomposition (denoted by dMOEA/D-L_p) is proposed in this paper. Specifically, dMOEA/D-L_p decomposes a dynamic multi-objective optimization problem into a number of dynamic scalar optimization subproblems and optimizes them simultaneously. The L_p decomposition method is referred and modified to use. To skillfully monitor environmental change and quickly respond to the new environment, an improved environment detection operator is presented and a subproblem-based memory scheme is adopted which allows the evolutionary algorithm to store good solutions from old environments and reuse them as necessary.

We conducted two sets of experimental studies . In the first set of experiments, we compared dMOEA/D-L_p against dMOEA/D-TCH, dMOEA/D-WS, and dMOEA/D-PBI to test the effectiveness of the L_p decomposition method for dynamic multi-objective optimization. In the second set of experiments, we compared dMOEA/D-L_p with two other popular algorithms (DNSGA-II and QICCA) to test the overall performance of the algorithm. Simulation results demonstrated the effectiveness of the L_p decomposition method for decomposition-based dynamic multi-objective optimization. Comparison results also revealed that the proposed dMOEA/D-L_p had quick tracking performance and better convergence, it was superior and preferable in solving dynamic multi-objective optimization problems.

Author Contributions: Conceptualization, X.X. and Y.T.; Formal analysis, S.L.; Funding acquisition, Y.T.; Investigation, X.X.; Methodology, X.X.; Project administration, W.Z.; Validation, Y.T. and W.Z.; Writing—original draft, X.X.; Writing—review & editing, Y.T.

Funding: This research was funded by the [National Natural Science Foundation of China] grant numbers [61401260, 61572298, 61602283], and the [Natural Science Foundation of Shandong, China] grant numbers [BS2014DX006, ZR2016FB10].

Acknowledgments: This research was funded by the National Natural Science Foundation of China (grant numbers: 61401260, 61572298, 61602283), and the Natural Science Foundation of Shandong, China (grant numbers: BS2014DX006, ZR2016FB10).

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

References

1. Qu, B. Evolutionary Algorithms for Solving Multi-Modal and Multi-Objective Optimization Problems. Ph.D. Thesis, Nanyang Technological University, Singapore, 2011.
2. Sindhya, K.; Miettinen, K.; Deb, K. A Hybrid Framework for Evolutionary Multi-Objective Optimization. *IEEE Trans. Evolut. Comput.* **2013**, *17*, 495–511. [CrossRef]
3. Farina, M.; Deb, K.; Amato, P. Dynamic multiobjective optimization problems: Test cases, approximations, and applications. *IEEE Trans. Evolut. Comput.* **2004**, *8*, 425–442. [CrossRef]
4. Wu, Y.; Jin, Y.; Liu, X. A directed search strategy for evolutionary dynamic multiobjective optimization. *Soft Comput.* **2015**, *19*, 3221–3235. [CrossRef]
5. Jin, Y.; Branke, J. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evolut. Comput.* **2005**, *9*, 303–317. [CrossRef]
6. Liu, C.; Wang, Y. Dynamic Multi-objective Optimization Evolutionary Algorithm. *Nat. Sci. J. Hainan Univ.* **2010**, *4*, 456–459.
7. Shang, R.; Jiao, L.; Ren, Y.; Li, L.; Wang, L. Quantum immune clonal coevolutionary algorithm for dynamic multiobjective optimization. *Nat. Comput. Int. J.* **2014**, *13*, 421–445. [CrossRef]
8. Zhang, Z. Multiobjective Optimization Immune Algorithm In Dynamic Environments And Its Application To Greenhouse Control. *Appl. Soft Comput.* **2008**, *8*, 959–971. [CrossRef]
9. Li, X.; Branke, J.; Kirley, M. On performance metrics and particle swarm methods for dynamic multiobjective optimization problems. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 25–28 September 2007; pp. 576–583.
10. Li, Y.; Zhan, Z.H.; Lin, S.; Zhang, J.; Luo, X. Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. *Inf. Sci.* **2015**, *293*, 370–382. [CrossRef]
11. Zheng, X.W.; Lu, D.J.; Wang, X.G.; Liu, H. A cooperative coevolutionary biogeography-based optimizer. *Appl. Intell.* **2015**, *43*, 95–111. [CrossRef]
12. Goh, C.K.; Tan, K.C. A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization. *IEEE Trans. Evolut. Comput.* **2009**, *13*, 103–127.
13. Jiang, S.; Yang, S. A Steady-state and Generational Evolutionary Algorithm for Dynamic Multiobjective Optimization. *IEEE Trans. Evolut. Comput.* **2017**, *21*, 65–82. [CrossRef]
14. Wang, Y.; Li, B. Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. *Memet. Comput.* **2010**, *2*, 3–24. [CrossRef]
15. Chen, J.H.; Cheng, C.W. Multi-objective evolutionary optimization of dynamic service facility location problems. In Proceedings of the 2011 IEEE Southeastcon, Nashville, TN, USA, 17–20 March 2011; pp. 333–338.
16. Deb, K.; Udaya, B.R.N.; Karthik, S. Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Matsushima, Japan, 5–8 March 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 803–817.
17. Hutzschenreuter, A.K.; Bosman, P.A.N.; Han, L.P. Evolutionary Multiobjective Optimization for Dynamic Hospital Resource Management. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Nantes, France, 7–10 April 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 320–334.
18. Li, J.Q.; Sang, H.Y.; Han, Y.Y.; Wang, C.G.; Gao, K.Z. Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *J. Clean. Prod.* **2018**, *181*, 584–598. [CrossRef]

19. Barlow, G.J.; Smith, S.F. A Memory Enhanced Evolutionary Algorithm for Dynamic Scheduling Problems. In *Workshops on Applications of Evolutionary Computation*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4974, pp. 606–615.
20. Yang, S.; Yao, X. Population-Based Incremental Learning with Associative Memory for Dynamic Environments. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 542–561. [CrossRef]
21. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolut. Comput.* **2002**, *6*, 182–197, [CrossRef]
22. Koo, W.T.; Chi, K.G.; Tan, K.C. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. *Memet. Comput.* **2010**, *2*, 87–110. [CrossRef]
23. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evolut. Comput.* **2007**, *11*, 712–731. [CrossRef]
24. Coello, C.A.C.; Lamont, G.B.; Van Veldhuizen, D.A. *Evolutionary Algorithms For Solving Multi-Objective Problems*; Springer: New York, NY, USA, 2007.
25. Miettinen, K. *Nonlinear Multiobjective Optimization*; Kluwer Academic Publishers: Boston, MA, USA, 1999.
26. Bai, J.; Liu, H. Multi-objective artificial bee algorithm based on decomposition by PBI method. *Appl. Intell.* **2016**, *45*, 976–991. [CrossRef]
27. Branke, J. Memory enhanced evolutionary algorithms for changing optimization problems. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1875–1882.
28. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evolut. Comput.* **2011**, *15*, 4–13. [CrossRef]
29. Tan, Y.Y.; Jiao, Y.C.; Li, H.; Wang, X.K. A modification to MOEA/D-DE for multiobjective optimization problems with complicated Pareto sets. *Inf. Sci.* **2012**, *213*, 14–38. [CrossRef]
30. Min, L. Memory Enhanced Dynamic Multi-Objective Evolutionary Algorithm Based on Decomposition. *J. Softw.* **2013**, *24*, 1571–1588.
31. Liu, R.; Niu, X.; Fan, J. An orthogonal predictive model-based dynamic multi-objective optimization algorithm. *Soft Comput*. **2015**, *19*, 3083–3107. [CrossRef]
32. Sola, M.C. Parallel Processing for Dynamic Multi-Objective Optimization. Ph.D. Thesis, University of Granada, Granada, Spain, 2010.
33. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.M.; Da Fonseca, V.G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evolut. Comput.* **2003**, *7*, 117–132. [CrossRef]
34. Veldhuizen, D.A.V.; Lamont, G.B. Evolutionary Computation and Convergence to a Pareto Front. Late Breaking Papers at the Genetic Programming 1998 Conference, Madison, WI, USA, 22–25 July 1998; pp. 221–228.
35. Veldhuizen, D.A.V.; Lamont, G.B. Multiobjective evolutionary algorithm test suites. In Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, TX, USA, 28 February–2 March 1999; pp. 351–357.
36. Okimoto, T.; Schwind, N.; Clement, M. Lp-Norm based algorithm for multi-objective distributed constraint optimization. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, Paris, France, 5–9 May 2014; Volume 18, pp. 1427–1428.

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).