# A Rough-to-Fine Evolutionary Multiobjective Optimization Algorithm

Fangqing Gu, Hai-Lin Liu, *Senior Member, IEEE*, Yiu-Ming Cheung, *Fellow, IEEE*, and Minyi Zheng

***Abstract*—This article presents a rough-to-fine evolutionary multiobjective optimization algorithm based on the decomposition for solving problems in which the solutions are initially far from the Pareto-optimal set. Subsequently, a tree is constructed by a modified $k$-means algorithm on $N$ uniform weight vectors, and each node of the tree contains a weight vector. Each node is associated with a subproblem with the help of its weight vector. Consequently, a subproblem tree can be established. It is easy to find that the descendant subproblems are refinements of their ancestor subproblems. The proposed algorithm approaches the Pareto front (PF) by solving a few subproblems in the first few levels to obtain a rough PF and gradually refining the PF by involving the subproblems level-by-level. This strategy is highly favorable for solving problems in which the solutions are initially far from the Pareto set. Moreover, the proposed algorithm has lower time complexity. Theoretical analysis shows the complexity of dealing with a new candidate solution is $\mathcal{O}(M \log N)$, where $M$ is the number of objectives. Empirical studies demonstrate the efficacy of the proposed algorithm.**

***Index Terms*—Decomposition, evolutionary algorithm, incremental, multiobjective optimization, tree-like weight design.**

## I. Introduction

WITHOUT loss of generality, a multiobjective optimization problem (MOP) can be formulated as follows:

$$\min_{\mathbf{x}\in[l_i,u_i]^n} F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))^T \tag{1}$$

where $l_i$ and $u_i$ are the lower and upper bounds of $x_i$ for $i = 1, \ldots, n$. $F : [l_i, u_i]^n \to \mathbb{R}^M$ consists of $M$ real-value objective functions and $M \geq 2$ is the number of objectives.

Over the past few decades, researchers have proposed a variety of evolutionary multiobjective optimization (EMO) algorithms [1]–[6]. The population size of most existing evolutionary algorithms is kept constant for an entire optimization run [7], and plenty of solutions evolve simultaneously. These algorithms can work well for the problems, where the Pareto-optimal set (PS) is widely distributed in the decision space as shown in Fig. 1(a), because the length of the evolutionary path from any point $C$ in the decision space to the expected solution $A$ is comparable to that of the path from an existing Pareto-optimal $B$ to $A$. However, the PS may be within a small region of the decision space as shown in Fig. 1(b). It makes the solutions initially far from the PS. It is advantageous for approaching the Pareto front (PF) and obtaining a rough PF with a smaller population size because the length of the evolutionary path from a current Pareto-optimal $B$ we have obtained to the expected solution $A$ is fairly short compared to the path from $C$ to $A$ as shown in Fig. 1(b). Once the population gets close to the PF, a larger population will be required to refine the PF for the problems, where the PS is within a small region of the decision space. In the literature, only a few attempts have been made to develop EMO algorithms with a varied population size [8]–[12]. Nevertheless, how to maintain the representation and diversity of the population is a crucial

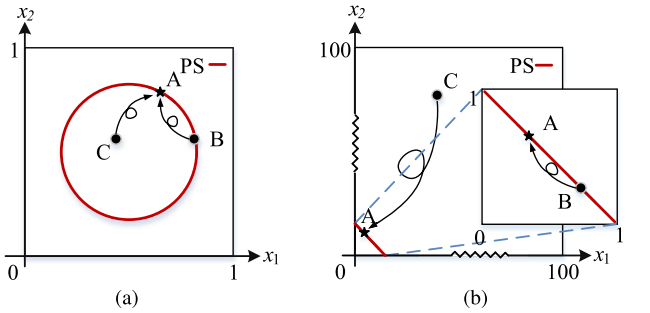


Fig. 1. Illustration of the challenges for solving the problems, where the PS concentrated in a small area. $A$ is the expected solution, $B$ is an existing Pareto-optimal solution, and $C$ is a point in the decision space. (a) PS widely distributed in the decision space. (b) PS within a small area of the decision space.

Manuscript received 12 November 2020; revised 18 February 2021 and 14 April 2021; accepted 7 May 2021. Date of publication 8 July 2021; date of current version 18 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61672444; in part by the Natural Science Foundation of Guangdong Province under Grant 2021A1515011839 and Grant 2020A1515011500; in part by the Programme of Science and Technology of Guangdong Province under Grant 2020A0505100056; in part by the Hong Kong Baptist University (HKBU), Research Committee, Initiation Grant, Faculty Niche Research Areas (IG-FNRA) 2018/19 under Grant RC-FNRA-IG/18-19/SCI/03; and in part by the Innovation and Technology Fund of Innovation and Technology Commission of the Government of the Hong Kong under Project ITS/339/18. This article was recommended by Associate Editor H. Ishibuchi. *(Corresponding authors: Hai-Lin Liu; Yiu-Ming Cheung.)*

Fangqing Gu is with the College of Applied Mathematics, Guangdong University of Technology, Guangzhou 510520, China, and also with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: fqgu@gdut.edu.cn).

Hai-Lin Liu and Minyi Zheng are with the School of Applied Mathematics, Guangdong University of Technology, Guangzhou 510520, China (e-mail: lhl@gdut.edu.cn).

Yiu-Ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: ymc@comp.hkbu.edu.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3081357.

Digital Object Identifier 10.1109/TCYB.2021.3081357

issue, which is conducive to making the population close to different parts of the PS.

To this end, a rough-to-fine EMO algorithm, called REMOA, is presented in this article. In this algorithm, a tree is constructed on $N$ uniform weight vectors by a modified $k$-means algorithm. Each node contains a weight vector and is associated with a subproblem with the help of its containing weight vector in this node. The subproblems in the descendants are the refinements of the subproblems of their ancestors. Namely, the subdomains related to descendants are the further decompositions of the subdomains related to their ancestors. In the beginning, nodes in the first $L = 2$ levels are set to be active. In the evolution process, only the subproblems in the active nodes maintain a certain solution to compose the population. As the rate of improvement of the population is lower than a given value, we extend the population by increasing the value of $L$ to make more nodes become active. This mechanism approaches the PS, obtains a rough PF, and gradually refines the PF by involving more subproblems level-by-level. We compare the proposed algorithm with 12 state-of-the-art EMO algorithms on ZDT test suites [13]. Furthermore, since the PS shapes of ZDT test suites are simple and the number of objectives is only two, we, therefore, construct nine MOPs to investigate the performance of the proposed algorithm. The main characteristic of these test instances is that their PSs are within a very small area of the decision space. It makes the solutions initially far from the PS. The empirical results demonstrate the efficacy of the proposed algorithm. The main advantages of the proposed REMOA are as follows.

1) *Low Time Complexity:* In the proposed algorithm, a candidate solution only needs to be compared with the solutions along the path from the root to a leaf node of the tree. Thus, the computational complexity of dealing with a candidate solution is only $\mathcal{O}(M \log N)$.
2) *Efficiency:* We present a rough-to-fine subproblem organization paradigm. Therefore, the proposed algorithm approaches the PF by solving a few representative subproblems and gradually refines the PF. The proposed algorithm is efficient for solving the MOP whose PS is within a very small area of the decision space.

The remainder of this article is organized as follows. In Section II, we make an overview of the popular EMO algorithms and the evolutionary algorithms with varied population size. Section III gives a detailed description of the proposed REMOA, including a subproblem organization mechanism based on a tree structure, the individual updating strategy, and the mating selection strategy. Section IV conducts empirical studies of the proposed algorithm in comparison with the existing counterparts. Finally, we draw a conclusion in Section V.

## II. Related Works

In this section, we present an overview of the popular EMO algorithms and those evolutionary algorithms with the varied population sizes.

### A. EMO Algorithms

Researchers have proposed a wide variety of EMO algorithms over the past decades [14], [15]. These EMO algorithms concurrently evolve a population to approximate the PF in a single run and can be divided into three categories: 1) dominance-based algorithms; 2) decomposition-based algorithms; and 3) indicator-based algorithms [16].

Dominance-based algorithms, such as the nondominated sorting genetic algorithm II (NSGA-II) [2] and strength Pareto evolutionary Algorithm 2 (SPEA2) [1], use the Pareto-based ranking as the primary selection mechanism and the density-based selection criterion, for example, the crowding distance, as a secondary selection mechanism. Deb and Jain introduced a reference-point-based algorithm, called NSGA-III, in [17] to improve the uniformity of the solutions. In recent years, a number of modified dominance-based algorithms have been presented for the challenges on specific problems [18]–[20]. Moreover, nondominated sorting is computationally intensive, in particular, when the population size increases. Some research work has been dedicated to addressing this problem [21], [22]. A new and efficient implementation of nondominated sorting, called DDA-NS, was developed in [23]. It requires $\mathcal{O}(M\log(N))$ objective function value comparisons and $\mathcal{O}(MN)$ integer comparisons for dealing with a solution. Unless otherwise specified, the time complexity of the algorithm is the computational complexity for dealing with one candidate solution, not that of the entire algorithm, throughout this article.

Decomposition-based algorithms decompose a MOP into a series of subproblems with the help of a set of weight vectors [15], [24], [25]. As one of the most well-known decomposition-based EMO algorithms, MOEA/D was proposed in [3]. In this algorithm, each candidate solution is only compared with its neighborhood. Thus, its time complexity is $\mathcal{O}(MT)$, where $T$ is the number of neighborhoods. We proposed a decomposition-based EMO algorithm, called M2M, in [16], [26], and [27]. It decomposes a MOP into a number of simple multiobjective optimization subproblems and each subproblem owns a subpopulation. Its time complexity is $\mathcal{O}(M\sqrt{N})$. Due to the potential for balancing convergence and diversity, some high-performance decomposition-based EMO algorithms have been developed [28], [29]. Recently, some excellent decomposition-based algorithms were presented in [30] and [31]. A tree-structure decomposition method was presented in [32] and [33]. It can adaptively generate the weight vectors in MOEA/D according to the distribution of the population. MaOEA/C [34] decomposes the population into some clusters and obtains a promising result on problems with incomplete and irregular PF. Moreover, some approaches have been developed to balance the convergence and diversity of the evolutionary process by combining dominance-based and decomposition-based approaches [35]–[38].

Indicator-based EMO algorithms, for example, the $S$-metric selection evolutionary multiobjective algorithm (SMS-EMOA) [4] and HypE [39], directly use the contributions of individuals in a performance indicator as the selection criteria to select offspring. Subsequently, the well-converged and well-distributed solutions can be preserved. Nevertheless, they suffer from the high computational cost. The runtime complexity of SMS-EMOA [4] is exponential in the number of objectives. Currently, some efforts have been done to

reduce such algorithms' computational complexity [40]–[42]. These strategies improve the performance of indicator-based EMO algorithms to some extent.

### B. Evolutionary Algorithms With Varied Population Size

The population size of the above-mentioned EMO algorithms is constant. The population size usually has an enormous effect on the performance of evolutionary algorithms [43], [44]. Generally, a small population size may result in premature convergence. In contrast, if the population size is too large, it will waste considerable resources and decrease the search efficiency of the algorithm [44]. Thus, a dynamic/adaptive population size might help in improving the performance of the algorithms. The existing related studies [8], [11], [12] mainly focus on adapting the population size. Unfortunately, there are few studies regarding how to preserve the representativeness of the solutions in the population with a small size.

Specifically, many efforts have been devoted to control the population size for single-objective optimization evolutionary algorithms [45], [46]. Several theoretical analyses regarding the choice of the population size are presented in [44] and [47]. These papers discuss the relationship between the population size and algorithm performance for some problems involving single-objective evolutionary algorithms. Moreover, some experimental studies have been presented to investigate the influence of the population size on algorithm performance. An adaptive differential evolution (DE) algorithm with a varied population size was proposed in [48] and [49], where the population size increases or decreases according to the search status of the current population. A genetic algorithm with a reducing population size scheme was proposed for solving single-objective optimization problems in [50] and [51]. For single-objective evolutionary algorithms, a larger population size in the primary stage of the evolution process is beneficial to identify the basin of the globally best solution and preserve the diversity of the population; whereas, a smaller population size is sufficient to identify the best solution within this basin.

However, only a few attempts have been made to extend automatic population growth control to multiobjective optimization [10]. Tan *et al.* [8] presented an incrementing EMO algorithm with a dynamic population size that is computed according to the discovered PF at each generation. Nag *et al.* [11] proposed an archive-based steady-state EMO algorithm. It bounds the archive (population) size between a minimum and a maximum. Glasmachers *et al.* [10] proposed an EMO algorithm, whose population size starts small and grows large as the generation increases. As claimed in [10], for EMO algorithms, only a rather limited population size is sufficient to approach the PF. Once the PF is reached, it needs a larger population size to well exploit and cover the PF. The experimental results demonstrate that increasing the population size similar to this can save fitness function evaluation compared to a fixed population size. Brog proposed in [9] adapted the population size according to the size of the $\epsilon-$box dominance archive. Recently, an unbounded population EMO algorithm has been proposed in [12] and [52], which maintains all nondominated solutions (NSs). The algorithm bears the potential to converge to the true PF and obtains a promising result. Nevertheless, it is always a challenging issue to reduce the complexity of EMO algorithms with an unbounded/larger population.

**Algorithm 1:** General Framework of REMOA

**Input:**
- A stopping criterion.
- Genetic operators and their associated parameters.
- $N$: the upper bound of the population size.
- *max_gen*: the maximum number of generations.

**Output:** The solutions contained in the tree $\mathbb{T}.\mathbf{X}$.

```
Generate N uniform unit weight vectors V := {v_1, ..., v_N} and
  N initial solutions P.
Set L := 2, Δ := 0.1 and v^c := (1/√M, 1/√M, ..., 1/√M)^T.
Initialize Δ_0 := 1, t := 1 and the archive A := P.
T := CreateTree(v^c, V).
Each active node selects the best solution from P according to
  (5) to form population T.X.
N* := |T.X|.
while t < max_gen do
    Q := MatingSelection(T, A, N).
    Create N offsprings Y based on Q.
    if Δ_{t-1} < Δ && N* < N then
        P := Y ∪ T.X, L := L + 1, Δ_t := 1.
        Each active node selects the best solution from P
          according to (5) to form population T.X.
        N* := |T.X|.
    else
        [T, A] := UpdateX(T, Y, A).
        Compute Δ_t by Eq. (6).
    end
    t := t + 1.
end
```

## III. Proposed Evolutionary Algorithm

### A. Framework of the Proposed REMOA

Algorithm 1 describes the framework of the proposed REMOA. Let $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ be $N$ uniform unit weight vectors. Based on these weight vectors, a tree $\mathbb{T}$ is constructed on these weight vectors by using the algorithm illustrated in Section III-B. Each node of the tree contains a weight vector and associates with a subproblem with the help of its weight vector. The nodes in the first $L = 2$ levels are set to be active and others without active descendants are recorded as leaf nodes. In the beginning, $N$ solutions are randomly created in the decision space. Each subproblem in the active node selects the best individual from the initial solutions according to the value of the aggregation function defined in (5). These selected solutions form the initial population $\mathbb{T}.\mathbf{X}$. The population size $N^* = |\mathbb{T}.\mathbf{X}|$ is the cardinal number of solution set $\mathbb{T}.\mathbf{X}$, that is, the number of active nodes. Its upper bound is the value of weight vectors $N$. When the population size $N^* < 0.5N$, an archive $\mathbf{A}$, which stores some better solutions, is introduced to avoid a crucial loss of diversity. The archive is initialized as the initial solution set and updated by the solutions that survive when updating the population as described in Section III-C. In each generation, a solution set $Q$ is selected by the mating selection strategy, which is illustrated in Section III-D. Afterward, we apply the genetic operators on

$Q$ to generate offspring $\mathbf{Y}$. If the rate of improvement of the population is less than a given value and $N^* < N$ at the same time, which means not all nodes are active, then we extend the population as described in Section III-D. Otherwise, the population is updated as illustrated in Section III-B. In the following paragraphs, the implementation details of each component in REMOA will be explained step-by-step.

### *B. Tree Structure Subproblem Organization Paradigm: Rough-to-Fine*

*1) Construct Tree on the Weight Vectors:* The main idea of this article is to propose a tree structure subproblem organization paradigm. The tree is constructed as follows: each node of the tree contains a weight vector. Let $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$ with $\mathbf{v}_i \in \mathbb{R}^M_+, i = 1, \ldots, N$ be a set of uniform unit weight vectors. Algorithm 2 illustrates the detailed description on the creation of the tree, where $\mathbf{v}^c$ is the center vector and $\mathbf{V}$ is a weight vector set assigned to this node. The center vector may not be in the weight vector set $\mathbf{V}$. We first find the weight vector $\mathbf{v}$ in $\mathbf{V}$ closest to the center vector $\mathbf{v}^c$ as follows:

$$\mathbf{v} = \arg\min_{\mathbf{v}_j \in \mathbf{V}} \text{dist}(\mathbf{v}^c, \mathbf{v}_j). \tag{2}$$

This node contains the weight vector $\mathbf{v}$ and $\mathbf{v}$ is removed from the weight vector set $\mathbf{V}$. The center vector $\mathbf{v}_c$ is determined using different methods for the root, the nodes in the second level, and the other ones. Specifically, it is given as follows.

1) The center vector of the root is set as $\mathbf{v}_c = ([1/\sqrt{M}], (1/\sqrt{M}), \ldots, [1/\sqrt{M}])^T$.
2) The center vectors of the nodes in the second level are the extreme points in $\mathbb{R}^M_+$. That is

$$\mathbf{e}_i = \Big(\underbrace{\frac{1}{\sqrt{M-1}}, \ldots, \frac{1}{\sqrt{M-1}}}_{i-1}, 0 \underbrace{\frac{1}{\sqrt{M-1}}, \ldots, \frac{1}{\sqrt{M-1}}}_{M-i}\Big)^T.$$

   Note that the number of nodes in the second level is equal to the number of objectives. It aims to identify the extreme solutions in the population for each objective. For the objective $f_i$, a corner (extreme) solution that has the minimum value of $f_i$ will be obtained by solving a subproblem with the extreme weight vector $\mathbf{e}_i$. Therefore, these weight vectors are used to guide the population widespread.
3) For other nodes, the center vectors are computed by a modified $k$-means algorithm (see lines 7 to 12) with the number of clusters $k = 2$. Specifically, in order to balance each cluster, each weight vector $\mathbf{v}_j$ is assigned into the weighted closest cluster. Its cluster can be calculated as follows:

$$\mathbf{L}_j \leftarrow \arg\min_i \lambda_i \text{dist}(\mathbf{v}^c_i, \mathbf{v}_j) \tag{3}$$

   where $\lambda_i$ is the percentage of the weight vectors assigned into the $i$th cluster (see line 12) and it is initialized to 0.5 for all clusters (see line 7). $\mathbf{v}^c_i$ is the center of the $i$th cluster.

**Algorithm 2:** Creating the Tree `CreateTree`($\mathbf{v}^c$, $\mathbf{V}$)

**Input:**
- $\mathbf{v}^c$: the center vector.
- $\mathbf{V}$: the weight vectors assigned to this node.
- *maxIter*: maximum iterative number.

**Output:** The tree $\mathbb{T}$.

1 $\mathbf{v} \coloneqq \arg\min_{\mathbf{v}_j \in \mathbf{V}} \text{dist}(\mathbf{v}^c, \mathbf{v}_j)$.
2 $\mathbb{T}.\mathbf{v} \coloneqq \mathbf{v}$.
3 $\mathbf{V} \leftarrow \mathbf{V} \setminus \mathbf{v}$.
4 **if** $|\mathbf{V}| = N - 1$ **then**
5  $\mathbf{v}^c_i \coloneqq \mathbf{e}_i, i = 1, \ldots, M$.
6 **else**
7  $\lambda \coloneqq (0.5, 0.5)$.
8  Initialize centers $\mathbf{v}^c_1, \mathbf{v}^c_2$.
9  **for** $t = 1$ **to** *maxIter* **do**
10  $\mathbf{L}_j \coloneqq \arg\min_i \lambda_i \text{dist}(\mathbf{v}^c_i, \mathbf{v}_j)$ for $j = 1, \ldots, |\mathbf{V}|$.
11  $\mathbf{v}^c_i \coloneqq \frac{\sum_{j=1}^{|\mathbf{V}|} \mathbf{I}\{L_j = i\}\mathbf{v}_j}{\sum_{j=1}^{|\mathbf{V}|} \mathbf{I}\{L_j = i\}}, i = 1, 2.$
12  $\lambda_i \coloneqq \frac{\sum_{j=1}^{|\mathbf{V}|} \mathbf{I}\{L_j = i\}}{|\mathbf{V}|}, i = 1, 2.$
13  **end**
14 **end**
15 Compute $\mathbf{V}_i$ which is the weight vector set assigned to $\mathbf{v}^c_i$.
16 `CreateTree`($\mathbf{v}^c_i$, $\mathbf{V}_i$) for all $\mathbf{v}^c_i$.

The root has $M$ children and the other nodes have at most two children. The weight vectors are assigned to the closest center. Namely, for each weight vector $\mathbf{v}_j \in \mathbf{V}$, its closest center is calculated as follows:

$$k^* = \arg\min_i \langle \mathbf{v}^c_i, \mathbf{v}_j \rangle \tag{4}$$

where $\langle \mathbf{v}^c_i, \mathbf{v}_j \rangle$ is the acute angle between $\mathbf{v}^c_i$ and $\mathbf{v}_j$. The weight vector set assigned to the $i$th center $\mathbf{v}^c_i$ is denoted as $\mathbf{V}_i$. We can create the tree via the constructed function recursive call (see line 16).

This article uses the PBI approach [7] to aggregate a MOP into a scalar optimization problem. Each node with vector $\mathbf{v}_i$ is associated with the subproblem defined as follows:

$$g(\mathbf{x}|\mathbf{v}_i) = d_1 + \theta d_2 \tag{5}$$

where $d_1 = F(\mathbf{x})^T \mathbf{v_i}$ and $d_2 = \|F(\mathbf{x}) - d_1 \mathbf{v}_i\|$. We can find that the weight vectors of descendants are refinements of the weight vectors of their ancestors. Therefore, the descendant subproblems are a refinement of their ancestors' subproblem. For example, Fig. 2(b) shows the tree constructed via the aforementioned strategy for a 2-D problem. $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{12}\}$ is a uniformly distributed weight vector set as shown in Fig. 2(a). Taking $\mathbf{v}_3$ as an example, the subproblems with the weight vectors $\mathbf{v}_2$ and $\mathbf{v}_4$ are refinements of the subproblem with $\mathbf{v}_3$.

*2) Adaptively Organizing These Subproblems Based on the Tree Structure:* The nodes in the first $L = 2$ levels are initially set to be active. As shown in Fig. 2(b), nodes $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_{12}$ with the solid circle are active. The population is initialized as follows: let $\mathbf{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be an initial solution set. Each subproblem in the active nodes selects the best solution according to the value of (5) from $\mathbf{P}$. The selected solutions form the initial population. $\mathbf{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_5\}$ is, therefore, the

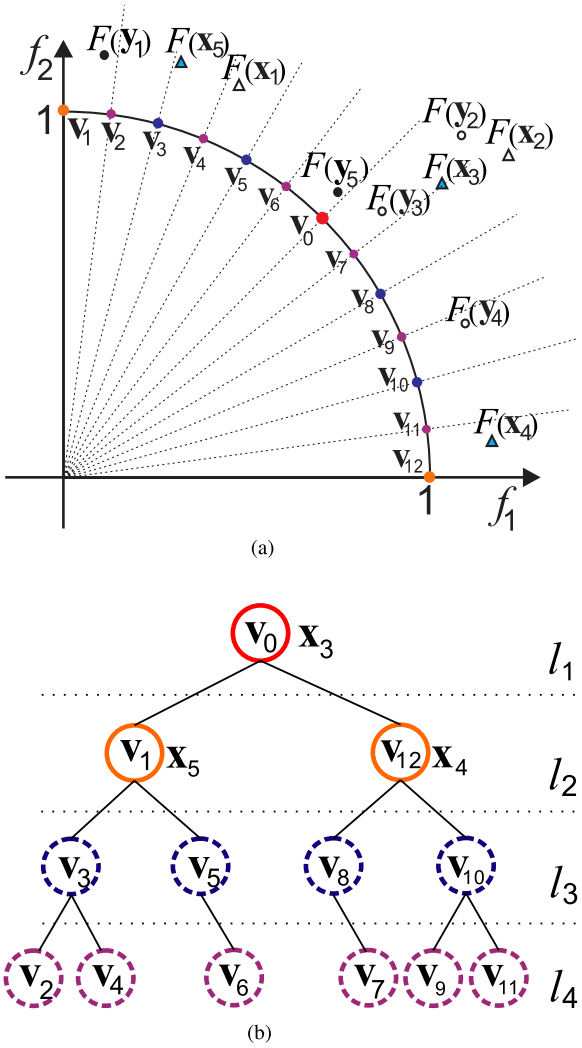


Fig. 2. Example of constructed tree and the illustration of the updating procedure. (a) Set of solutions of a simple example and a set of uniform weight vectors. (b) Constructed tree and the levels of the solutions.

solution set and their objective vectors are shown in Fig. 2(a). The solutions $\mathbf{x}_3, \mathbf{x}_4$, and $\mathbf{x}_5$ with solid triangles are selected to constitute the initial population. The population size $N^*$ is the number of the active nodes.

When $N^* < N$ and the rate of improvement of the population in convergence is less than a given value, we extend the population to improve the diversity of the population. Some metrics have been presented to measure the contributions of subspaces to the population convergence [53]. In this article, the rate of improvement of population is defined as the total improvement of individuals in the population. It is given as follows:

$$\Delta_t = \sum_{i=1}^{N^*} \max\left(g\left(\mathbf{x}_i^{(t-1)}|\mathbf{v}_i\right) - g\left(\mathbf{x}_i^{(t)}|\mathbf{v}_i\right), 0\right) \tag{6}$$

where $\mathbf{x}_i^{(t-1)}$ and $\mathbf{x}_i^{(t)}$ are solutions of the same subproblem in the $t-1$ and $t$ generations, respectively. If $\Delta_t < \Delta$, we set $L = L + 1$ and make the nodes in a new level be active. The population size increases to the number of active nodes.

**Algorithm 3:** Updating Population. `UpdateX`($\mathbb{T}$, $\mathbf{Y}$, $\mathbf{A}$)

**Input:**
- The tree $\mathbb{T}$.
- The solutions $\mathbf{Y}$.
- The archive $\mathbf{A}$.

**Output:** The updated tree $\mathbb{T}$ and the archive $\mathbf{A}$.

**foreach** $\mathbf{y}_i \in \mathbf{Y}$ **do**
  $\mathbb{P} = \mathbb{T}_0$.
  **while** *1* **do**
    **if** $g(\mathbf{y}_i|\mathbb{P}.\mathbf{v}) < g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v})$ **then** `// Region I`
      $\mathbb{P}.\mathbf{x} \rightleftharpoons \mathbf{y}_i$. `// Swapping solutions`
      $\mathbb{P}.\mathbf{s} = g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v})\mathbb{P}.\mathbf{v}$.
    **else if** $\mathbb{P}.\mathbf{s} \prec F(\mathbf{y}_i)$ **then** `// Region III`
      break.
    **else** `// Region II`
      **if** $\mathbb{P}$ *is a leaf node* **then**
        $\mathbf{y}_i$ randomly replaces a solution in $\mathbf{A}$.
        break.
      **else**
        $j^* = \arg\min_{j \in \mathbb{P}.\text{Children}} \langle \mathbb{T}_j.\mathbf{v}, F(\mathbf{y}_i) \rangle$.
        $\mathbb{P} = \mathbb{T}_{j^*}$.
      **end**
    **end**
  **end**
**end**

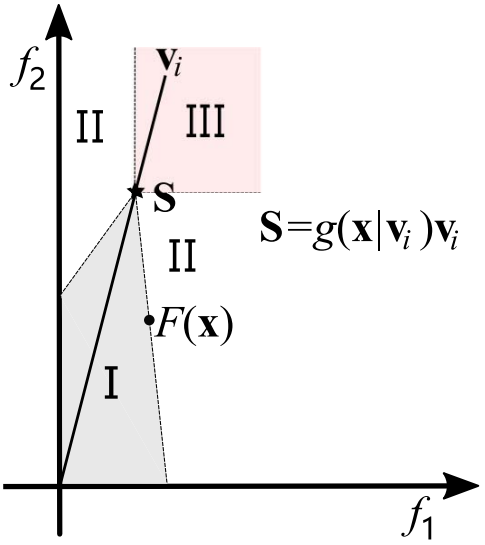


Fig. 3. Objective space partitioning for a give weight vector and solution.

Each subproblem in the active node reselects the best solution according to the value of (5) to yield the population.

### C. Updating of the Population

A top-down approach based on the tree structure is suggested to update the population. For each subproblem with weight vector $\mathbf{v}_i$, $\mathbf{x}$ is the current best solution. $\mathbf{s}$ is an equivalent point in the direction of weight vector $\mathbf{v}_i$ that has the same value as $F(\mathbf{x})$ for (5). It is given as follows:

$$\mathbf{s} = g(\mathbf{x}|\mathbf{v}_i)\mathbf{v}_i. \tag{7}$$

Thereby, the objective space is divided into three subregions I, II, and III by the equivalent point $\mathbf{s}$. A solution $\mathbf{y}$ is in region I if it has a value less than $\mathbf{x}$ according to (5), that is,

$g(\mathbf{y}|\mathbf{v}_i) < g(\mathbf{x}|\mathbf{v}_i)$. It is supposed to be better than $\mathbf{x}$. Region III denotes the regions in which the solution is dominated by $\mathbf{s}$. The solutions in this region are considered to be definitely worse than $\mathbf{x}$. Otherwise, the solution will be in region II. It is worse than $\mathbf{x}$ according to the subproblem with $\mathbf{v}_i$, but it is still a promising one. For instance, Fig. 3 shows the objective space partitioning for the subproblem with weight vector $\mathbf{v}_i$ and solution $\mathbf{x}$.

Consequently, Algorithm 3 contains the pseudocode of the updating procedure. In this algorithm, $\mathbb{T}_j$ is the $j$th node and $\mathbb{P}$ is a node cursor. $\mathbb{P}$ is initialized with the root. $\mathbb{P}.\mathbf{v}$ and $\mathbb{P}.\mathbf{x}$ denote the weight vector and solution related with node $\mathbb{P}$ and $\mathbb{P}.\mathbf{s} = g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v})\mathbb{P}.\mathbf{v}$. A node without active children is a leaf node. For each candidate solution $\mathbf{y}_i$ in a solution set $\mathbf{Y}$, we compare it with the solution related to the node cursor. There are three cases of comparison.

1) *Case I:* If $\mathbf{y}_i$ is in region I, that is, $g(\mathbf{y}_i|\mathbb{P}.\mathbf{v}) < g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v})$, $\mathbb{P}.\mathbf{x}$ is swapped with $\mathbf{y}_i$ and the equivalent point is updated by $\mathbb{P}.\mathbf{s} = g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v})\mathbb{P}.\mathbf{v}$ (see line 5).
2) *Case II:* If $\mathbf{y}_i$ is in region III, that is, $\mathbb{P}.\mathbf{s} \prec F(\mathbf{y}_i)$, no operation is required (see line 8).
3) *Case III:* $\mathbf{y}_i$ is otherwise in region II. If $\mathbb{P}$ is a leaf node, $\mathbf{y}_i$ randomly replaces a solution in the archive $\mathbf{A}$. Otherwise, $\mathbf{y}_i$ is transmitted to the closest child, which is calculated as follows:

$$j^* = \arg\min_{j \in \mathbb{P}.\text{Children}} \left\langle \mathbb{T}_j.\mathbf{v}, F(\mathbf{y}_i) \right\rangle \tag{8}$$

where $\langle \mathbb{T}_j.\mathbf{v}, F(\mathbf{y}_i)\rangle$ is the angle between vectors $\mathbb{T}_j.\mathbf{v}$ and $F(\mathbf{y}_i)$. The node cursor $\mathbb{P}$ is moved into the child node $\mathbb{T}_{j^*}$.

Fig. 2 shows an example of the updating procedure. A new solution set $\{\mathbf{y}_1, \ldots, \mathbf{y}_5\}$, whose objective vectors are shown in Fig. 2(a), is used to update the population one by one. Let us take $\mathbf{y}_1$ as an example. The node cursor $\mathbb{P}$ initially refers to the root. We first compare $\mathbf{y}_1$ with solution $\mathbf{x}_3$ contained in $\mathbb{P}$. Since $\mathbf{y}_1$ has been in the region II of the subproblem with weight vector $\mathbf{v}_0$ and $\mathbf{x}_3$, $\mathbf{y}_1$ is assigned into the child with weight vector $\mathbf{v}_1$ by (8) and the node cursor $\mathbb{P}$ redirects to the node with weight vector $\mathbf{v}_1$. $\mathbf{y}_1$ is compared with $\mathbf{x}_5$ contained in the node cursor. $\mathbf{y}_1$ is in the region I of the subproblem with $\mathbf{v}_1$ and $\mathbf{x}_5$. Therefore, $\mathbb{P}.\mathbf{x}$ is swapped with $\mathbf{y}_1$, and the equivalent point is updated by $\mathbb{P}.\mathbf{s} = g(\mathbb{P}.\mathbf{x}|\mathbb{P}.\mathbf{v}_1)\mathbb{P}.\mathbf{v}_1$. Since the node cursor refers to a leaf node, the updating procedure of $\mathbf{y}_1$ is finished. $\mathbf{x}_5$ is not dominated by the equivalent point. Hence, $\mathbf{x}_5$ randomly replaces a solution in the archive. In this manner, we can obtain the population $\{\mathbf{y}_5, \mathbf{y}_1, \mathbf{x}_4\}$ after the update.

### *D. Mating Selection Based on the Lifetime and the Acute Angle*

In general, if the straight line $(f_1/v_1) = (f_2/v_2) = \cdots = (f_M/v_M)$, taking $f_1, f_2, \ldots, f_M$ as variables, intersects with the PF, the coordinate of the intersection point is the objective vector of the optimal solution of the scalar problem (5) with weight vector $\mathbf{v} = (v_1, \ldots, v_M)^T$ [54]. This means that the angle between the weight vector $\mathbf{v}$ and the objective vector of the Pareto-optimal solution of its corresponding scalar problem will be 0. Therefore, the solution has a greater acute angle between the weight vector and objective vector. It generally is further away from the optimal solution of the scalar problem. Hence, we assume that the solution has more capacity for improvement. However, if the line does not intersect with the PF, this assumption will be invalid. A lifetime of the solution is introduced to fix this flaw. The longer the lifetime of a solution is, the lower the probability of it being selected.

Accordingly, we present a mating selection strategy based on the lifetime and the acute angle between its objective vector and its corresponding weight vector. Specifically, for each solution $\mathbf{x}_i$ in the population, we record the number of generations from producing to present as $\text{age}_i$ and compute the acute angle between its objective vector and its corresponding weight vector as $\text{angle}_i$. $N$ solutions are selected from the population to form $parent1 = \{\mathbf{x}_1^{p1}, \mathbf{x}_2^{p1}, \ldots, \mathbf{x}_N^{p1}\}$ by using tournament selection. Solutions with smaller age and greater angle are preferred in the tournament selection. For two solutions, we compare the parameter age first. The solution whose value for age is less will be selected. If these two solutions have the same value for age, the solution with the larger value of angle will be selected.

For each parent $\mathbf{x}_i^{p1} \in parent1$, the other parent $\mathbf{x}_i^{p2}$ corresponding to $\mathbf{x}_i^{p1}$ is selected by different methods according to the population size.

1) In the primary stage of the algorithm, when the population size $N^* \leq N/2$, if only the individuals in the population are involved in recombination, the algorithm may suffer from loss of diversity and premature convergence. Therefore, an archive is introduced to promote diversity, and $\mathbf{x}_i^{p2}$ is randomly selected from the archive $\mathbf{A}$ to favor exploration.
2) In the late stage of the algorithm, when the population size $N^* > N/2$, the recombination of two similar solutions is beneficial for exploitation. Thus, $\mathbf{x}_i^{p2}$ is randomly selected from the neighbors of $\mathbf{x}_i^{p1}$, which are defined on the tree. For each active node $\mathbb{T}_i$, its neighbors are the active nodes $\mathbb{T}_j$ with the length of the edge from $\mathbb{T}_i$ to $\mathbb{T}_j$ that is less than a given value. This value is set to 2 in this article. The regions of these neighbors are different for different weight vectors. The neighbors of the root are distributed in the entire first octant because its neighbors include the extreme weight vectors. Whereas the neighbors of a leaf will focus on a smaller area. This can provide a balance between exploration and exploitation. As shown in Fig. 2(a), we can see that the neighbors of $\mathbf{v}_1$ are the nodes containing weight vectors $\{\mathbf{v}_0, \mathbf{v}_{12}, \mathbf{v}_3, \mathbf{v}_2, \mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_6\}$ as all nodes are active.

The simulated binary crossover (SBX) and polynomial mutation [7] operators are performed on each pair of parents $\mathbf{x}_i^{p1} \in parent1$ and $\mathbf{x}_i^{p2}$ for generating offspring.

### *E. Discussion*

*1) Computational Complexity Analysis:* From the framework of the proposed algorithm, we can see that the difference in the computational complexity is due to the population updating. Given a MOP with $M$ objectives and a tree with

size $N$, the maximum number of levels is $\lfloor \log N \rfloor + 1$ because the constructed tree is comparatively balanced. Here, $\lfloor \log N \rfloor$ is the largest integer less than or equal to $\log N$. The time complexity of updating one candidate solution of REMOA is as follows: let $L \leq \lfloor \log N \rfloor + 1$ be the number of levels of active nodes. We need to compare the candidate solutions with the solutions along the path from the root to a leaf node in the worst case according to the dominance relationship. The time complexity of computing the dominance relationship of two solutions is $\mathcal{O}(M)$. Thus, it requires a computationally $\mathcal{O}(ML)$ comparison. Furthermore, we need to assign the candidate solution to the child node by computing the inner product of objective vectors and weight vectors. It also requires $\mathcal{O}(ML)$ multiplication. Thus, the time complexity of updating one candidate solution is $\mathcal{O}(M \log N)$ because $L \leq \lfloor \log N \rfloor + 1$. Compared to state-of-the-art algorithms, for example, NSGA-II [2] requires $\mathcal{O}(MN)$ and MOEA/D [3] requires $\mathcal{O}(MT)$ for updating one candidate solution, the proposed algorithm has lower computational complexity. Especially, in the primary stage of the algorithm, there are only a few nodes are active. The level $L$ of active nodes is much less than $\log N$. The computational complexity of the proposed REMOA is very low.

*2) Proposed REMOA Has Higher Convergence Rate:* In the primary stage of the evolution process, solutions in the population are generally far away from the Pareto-optimal solutions. The proposed REMOA decomposes the search domain of the MOP into a few subdomains related to the subproblems in the active nodes at the beginning. The proposed REMOA concentrates the computational resources (evaluation function) to solve these few subproblems. The count of evaluation functions, spent in the standard EMO algorithms with constant population size evolving the population for one generation, is equivalent to optimizing these few subproblems many times. Hence, the population of the proposed algorithm can quickly approach the PF by optimizing these few scalar subproblems. Nevertheless, this makes the diversity and representativeness of the population even more important. The algorithm with a small population size is easy to run into prematurity.

*3) Proposed Algorithm Can Preserve the Diversity and Uniformity of the Population:* In the proposed REMOA, an archive is introduced to store some better solutions that are not dominated by the solutions in the population. It can effectively use the promising solution and enhance the diversity of the solutions involved in recombination. Moreover, the weight vectors are used to guide the search process. A set of uniform weight vectors leads to a set of uniform NSs [55]. Therefore, we only need to analyze the uniformity of the weight vectors of the active nodes. To facilitate the description, we denote the weight vectors with level $L_i$ as $\bar{\mathbf{V}}_i$ when it causes no confusion. Since the counts of weight vectors in the clusters obtained by the modified $k$-means algorithm are close, the constructed tree is comparatively balanced. This is an excellent characteristic. It makes the proposed algorithm have lower computational complexity. Moreover, for any $i$ that is less than the maximum level, the constructed tree has two following properties.

1) The weight vectors in any $\bar{\mathbf{V}}_i$ are relatively uniform.
2) Unions of the weight vectors $\bigcup_{k=1}^{i} \bar{\mathbf{V}}_k$ are also relatively uniform.

This means that solutions in any of the same levels are fairly uniform. Moreover, solutions from the first level to any level are also fairly uniform.

## IV. Experimental Study

In this section, we will focus on experimental studies to investigate the performance of the proposed algorithm. First, we give a brief introduction om the test problems and the compared algorithms used in this article. Second, we present performance metrics used in numerical experiments and parameter settings of the compared algorithms. Finally, we provide the experimental results and analyses.

### A. Test Problems

We compared the proposed algorithm with other algorithms on the benchmarks ZDT1–ZDT4, ZDT6 [13] and nine test problems, MF1–MF9, constructed in this article. The PSs of ZDT test instances are concentrated in a line segment defined by $0 \leq x_1 \leq 1$ and $x_2 = \cdots = x_n = 0$. The parameter settings of the ZDT test instances are used as described in [3]. In view of the PS shapes being simple and the number of objectives being only two, we construct nine MOPs: MF1–MF7, with complicated PS shapes, and MF8 and MF9, with three objectives. MF1–MF7 are modified from the test instances presented in [56], while MF8 and MF9 are modified from DTLZ1 and DTLZ3 [13]. A detailed description of these test problems can be found in the supplementary materials of this article.

### B. Compared Algorithms

We compared the proposed REMOA with the following 12 representative EMO algorithms: Pareto-based NSGA-II [2], indicator-based SMS-EMOA [4], indicator-based evolutionary algorithm (IBEA) [57], SRA2 [58], and SMS-EMOAc [10] which has an increasing population size, decomposition-based MOEA/D [3], M2M [27], RVEA [59], MaOEA/C [34], and MOEA/D-CMA [60], and hybrid EMO algorithms Two_Arch2 [61] and multistage evolutionary algorithm framework (MSEA) [38] which integrates multiple selection strategies in a framework. Here, we introduce the main idea of these algorithms.

*1) NSGA-II:* This method [2] is one of the most popular EMO algorithms for solving MOPs. It uses the Pareto-based ranking as the primary selection mechanism and the density-based selection criterion, for example, the crowding distance, as a secondary selection mechanism.

*2) SMS-EMOA:* SMS-EMOA [4] uses the contributions of individuals in hypervolume measure as a selection criterion to select offsprings. It aims to maximize the dominated hypervolume of the obtained solution set within the optimization process. It well preserves the convergence and diversity of the candidate solution set as much as possible.

*3) SMS-EMOAc:* A version of SMS-EMOA with adaptation population size, denoted as SMS-EMOAc, was proposed in [10]. The results obtained in this article showed that an increasing population size during an EMO algorithm run

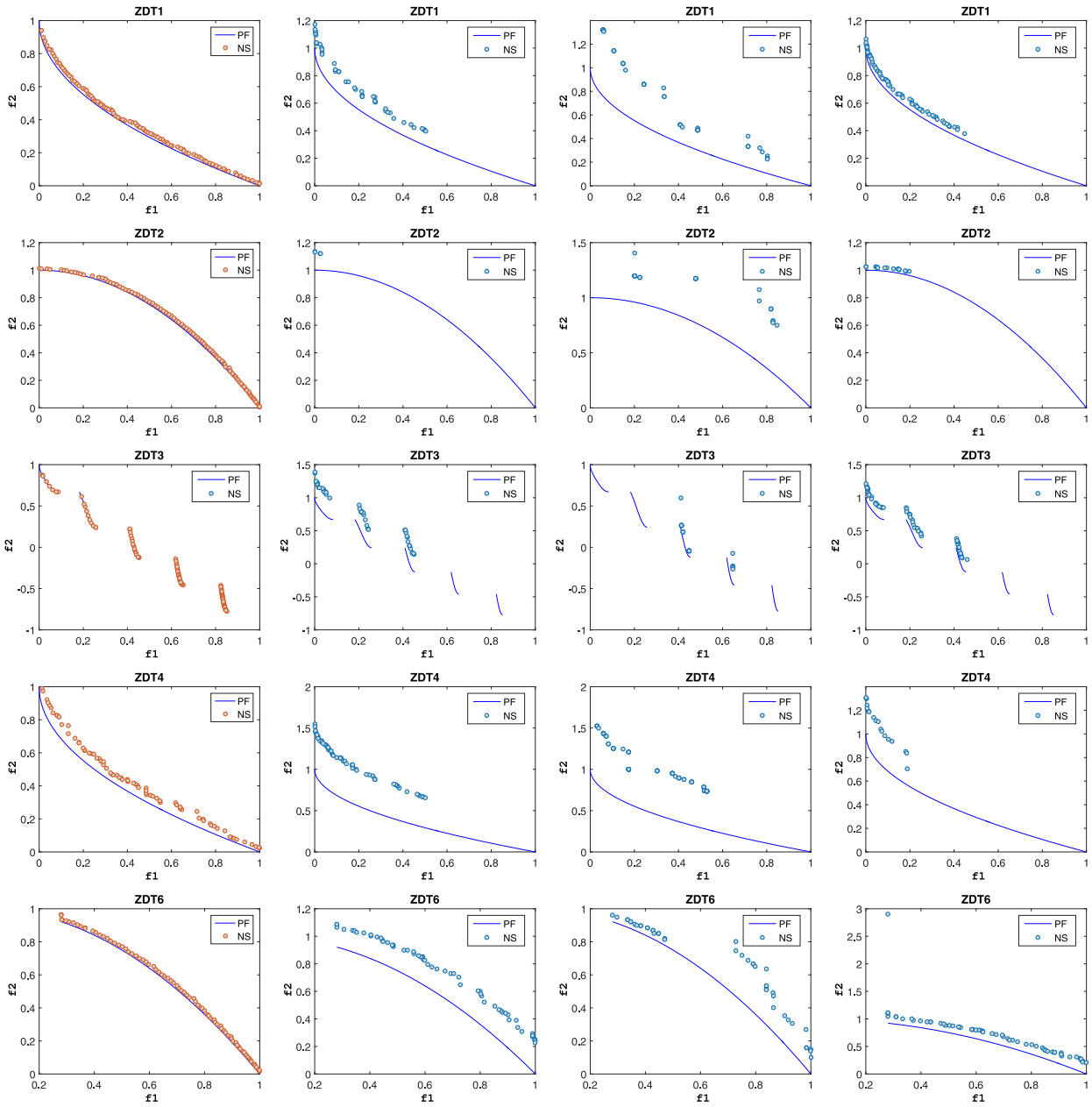


Fig. 4. Final NS with the median of the IGD value found by REMOA (the first column), NSGA-II (the second column), MOEA/D (the third column), and SMS-EMOA (the fourth column) for ZDT1–ZDT4 and ZDT6.

saves fitness function evaluations compared to a fixed population size.

*4) IBEA:* Zitzler and Künzli [57] first proposed a general indicator-based EMO algorithm framework called the IBEA. IBEA uses indicators to compare solutions and choose the next-generation population, which requires no additional diversity-preserving mechanism. The binary additive $\epsilon-$indicator $I_{\epsilon+}$ is used in the simulation study in this article.

*5) SRA2:* The stochastic ranking-based multiindicator algorithm (SRA) adopts the stochastic ranking technique to balance the search biases of different indicators [58]. It further improves the performance of the algorithm by incorporating a direction-based archive to store well-converged solutions and maintain diversity.

*6) MOEA/D:* As one of the most well-known decomposition-based EMO algorithms, it decomposes the MOP into a number of scalar subproblems by a set of weight vectors and simultaneously optimizes them [3]. Since each subproblem is associated with a search direction, it can provide sufficient selection pressure toward the PF.

*7) M2M:* This approach decomposes a MOP into a number of simple multiobjective optimization subproblems and solves these subproblems in a collaborative manner [27]. It gives priority to diversity. Therefore, it is good at maintaining the population diversity and achieving good performance at solving the imbalanced MOPs.

*8) RVEA:* This method decomposes a MOP into a number of single-objective subproblems using a set of reference

vectors [59]. The angle-penalized distance is designed to dynamically balance convergence and diversity according to the number of objectives and the number of generations. Using this approach, we can measure the convergence by the distance between solutions and ideal points, and the diversity by the angles between solutions and reference vectors.

*9) Two_Arch2:* An improved two-archive EMO algorithm, called Two_Arch2, was proposed in [61]. In Two_Arch2, two archives (convergence archive and diversity archive) are maintained during the evolutionary search and two different selection principles (indicator-based and Pareto-based) are assigned to the two archives: these can play a balanced role in terms of convergence, diversity, and complexity.

*10) MaOEA/C:* MaOEA/C [34] decomposes the population into some clusters by a hierarchical clustering method, such that the population's distribution can be well portrayed. It obtains promising results on problems with incomplete and irregular PF.

*11) MOEA/D-CMA:* A decomposition-based EMO algorithm, which uses both DE and covariance matrix adaptation has been proposed in [60]. The experimental studies show that it is suitable for dealing with problems with biases.

*12) MSEA:* An MSEA proposed in [38] divided the optimization process into three stages and used different selection strategies in these stages. It can overcome the limitations of those selection criteria.

### C. Performance Metrics

To evaluate the performance of the compared algorithms, two widely used metrics: 1) the inverted generational distance (IGD) [62] and 2) hypervolume indicator (HV) [63], are used in this article. A brief introduction of these two metrics is given below.

*1) IGD:* Let $Q$ be an approximation of the PF obtained by an algorithm and $Q^*$ be a set of reference points that are uniformly distributed along the PF. The IGD metric is defined as follows:

$$\mathrm{IGD}(Q|Q^*) = \frac{\sum_{v\in Q^*} d(v, Q)}{|Q^*|} \tag{9}$$

where $d(v, Q)$ is the minimum Euclidean distance from the point $v$ to $Q$ and $|Q^*|$ is the size of $Q^*$. Since the exact PF of the test problems is known *a priori*, 10 000 reference points are sampled on the PF, as described in [35] and [61], to form $Q^*$ in this article. These points are the intersecting points of a set of uniform weight vectors and the PF. Since the number of objectives of the test problems is two or three, 10 000 points can cover the true PF very well. The IGD metric can provide a good measure of both the convergence and diversity of a solution set. The smaller the IGD, the better the approximation.

*2) HV:* Let $Q$ be the objective vectors of solutions obtained by an algorithm and $\mathbf{z}^r = (z_1^r, \ldots, z_M^r)$ be a reference point dominated by any Pareto-optimal objective vector. $HV(Q|\mathbf{z}^r)$ is the volume of the region that is dominated by $Q$ and dominates $\mathbf{z}^r$; it is given as follows:

$$HV(Q|\mathbf{z}^r) = \mathrm{Vol}\left(\bigcup_{\mathbf{x}\in Q}\left[f_1(\mathbf{x}), z_1^r\right] \times \cdots \times \left[f_M(\mathbf{x}), z_M^r\right]\right)$$

where Vol(.) indicates the Lebesgue measure. The greater the HV, the better the approximation. In this work, we set the reference point to 1.1 times of the upper bounds of the true PFs. That is, $\mathbf{z}^r = (1.1, 1.1)$ for the two-objective test instances and $\mathbf{z}^r = (1.1, 1.1, 1.1)$ for the three-objective test instances, MF8 and MF9.

### D. Parameter Settings

The experiments are conducted on PlatEMO[1] [64], which is a MATLAB-based framework for multiobjective and many-objective optimization. All of the compared algorithms were conducted 30 times independently for all test instances. Their parameters are as follows.

1) The population size $N$ of the compared algorithms is set to be 100 for two-objective test instances and 210 for three-objective test instances MF8 and MF9. The upper bound of the population size in the proposed REMOA, that is, the size of the tree, is the same as the population size of compared algorithms.
2) All algorithms stop when the number of generations reaches the maximum number. Specifically, the maximum number is set to be 100 for ZDT test instances, 500 for MF1-MF7, and 200 for MF8 and MF9.
3) The weight vectors used in SRA2, MOEA/D, M2M, RVEA, MOEA/D-CMA, and the proposed REMOA are generated via a systematic approach proposed in [35]. The number of weight vectors is set to be 100 and 210 for the problems with 2 and 3 objectives, separately.
4) The neighborhood size of each subproblem in MOEA/D and SRA2 [58] is set to 20, and the maximum replacement number is set to 2.

### E. Experimental Results and Analysis

Table I reports the average values of the IGD metric obtained by the algorithms in 30 independent runs for each test instance. Table II lists the average value of the HV metric obtained by the algorithms in 30 independent runs for each test instance. The results in bold are the best of those obtained by these algorithms in each test instance, and † indicates the corresponding EMO algorithm is worse than the proposed REMOA according to Wilcoxon's rank-sum test. In addition, due to space limitations, we only plotted the NSs with the median IGD value obtained by the proposed REMOA and Pareto-based NSGA-II, decomposition-based MOEA/D, and indicator-based SMS-EMOA in 30 independent runs. The number of solutions obtained by the proposed REMOA is less than that of the compared algorithms when there are some nodes that are not active in the tree. The NSs obtained by other algorithms are plotted in the supplementary material.

[1] PlatEMO can be downloaded at http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html.

TABLE I
PERFORMANCE COMPARISON OF REMOA WITH 12 STATE-OF-THE-ART ALGORITHMS IN TERM OF THE AVERAGE IGD VALUES ON ZDT TEST SUITES AND MF1–MF9. † INDICATES THAT THE CORRESPONDING EMO ALGORITHM IS WORSE THAN THE PROPOSED REMOA ACCORDING TO WILCOXON'S RANK SUM TEST

| Instances | REMOA | SMS-EMOA | SMS-EMOAc | NSGA-II | IBEA | SRA2 | MOEA/D | M2M | RVEA | Two_Arch2 | MaOEA/C | MOEA/D-CMA | MSEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ZDT1 | **0.0165** | 0.2005† | 4.2676† | 0.3417† | 0.3592† | 4.5343† | 0.1755† | 7.7742† | 1.6275† | 7.0779† | 0.2193† | 1.2126† | 14.540† |
| ZDT2 | **0.0161** | 0.4778† | 6.3292† | 0.6846† | 0.6790† | 5.8736† | 0.3560† | 9.7278† | 1.5445† | 9.7111† | 0.6093† | 2.3230† | 36.109† |
| ZDT3 | **0.0140** | 0.2127† | 5.0678† | 0.3172† | 0.4095† | 4.7458† | 0.2752† | 7.9762† | 1.4269† | 8.0299† | 0.1379† | 1.1289† | 16.331† |
| ZDT4 | **0.0527** | 0.5192† | 2.4422† | 0.5934† | 0.5166† | 2.3288† | 0.5074† | 2.4512† | 1.3049† | 1.0995† | 0.4320† | 5.7876† | 9.666† |
| ZDT6 | **0.0130** | 0.4913† | 0.2509† | 0.1456† | 0.0752† | 1.2183† | 0.0933† | 0.5902† | 0.5421† | 1.2530† | 0.2510† | 0.0403† | 0.7597† |
| MF1 | **0.0174** | 0.4513† | 0.0495† | 0.0308† | 0.3775† | 0.3979† | 0.1349† | 0.0331† | 0.1162† | 0.1787† | 0.0495† | 0.1828† | 0.0339† |
| MF2 | **0.0717** | 0.3871† | 0.1244† | 0.2711† | 0.3552† | 0.3557† | 0.5939† | 0.1508† | 0.3155† | 0.1449† | 0.1244† | 0.3072† | 0.2416† |
| MF3 | **0.0630** | 0.2505† | 0.2879† | 0.1609† | 0.2477† | 0.1599† | 0.2331† | 0.0864† | 0.1377† | 0.1279† | 0.2879† | 0.2191† | 0.1821† |
| MF4 | **0.0258** | 0.3896† | 0.1105† | 0.1011† | 0.3673† | 0.4064† | 0.3093† | 0.0452† | 0.1728† | 0.2582† | 0.1105† | 0.2092† | 0.0516† |
| MF5 | **0.0343** | 0.3593† | 0.2822† | 0.2230† | 0.3551† | 0.3475† | 0.5932† | 0.1523† | 0.2920† | 0.2347† | 0.2823† | 0.2819† | 0.3458† |
| MF6 | 0.0772 | 0.2092† | 0.2563† | **0.0360** | 0.2062† | 0.1091† | 0.2410† | 0.0589 | 0.1184† | 0.0968 | 0.2563† | 0.2211† | 0.2086† |
| MF7 | **0.0249** | 0.4692† | 0.0367† | 0.0238 | 0.3964† | 0.4035† | 0.1419† | 0.0681† | 0.1485† | 0.3034† | 0.0368† | 0.2553† | 0.0574† |
| MF8 | **0.0860** | 0.4216† | 0.1560† | 0.2664† | 0.6608† | 0.8164† | 0.5568† | 0.3196† | 0.4747† | 0.8810† | 0.1560† | 3.2801† | 0.1714† |
| MF9 | **0.1072** | 0.7284† | 4.2610† | 5.8879† | 0.8133† | 1.6482† | 0.9744† | 0.4030† | 0.6152† | 1.6697† | 4.2610† | 10.376† | 10.541† |

TABLE II
PERFORMANCE COMPARISON OF REMOA WITH 12 STATE-OF-THE-ART ALGORITHMS IN TERMS OF THE AVERAGE HV VALUES ON ZDT TEST SUITES AND MF1–MF9. † INDICATES THAT THE CORRESPONDING EMO ALGORITHM IS WORSE THAN THE PROPOSED REMOA ACCORDING TO WILCOXON'S RANK SUM TEST

| Instances | REMOA | SMS-EMOA | SMS-EMOAc | NSGA-II | IBEA | SRA2 | MOEA/D | M2M | RVEA | Two_Arch2 | MaOEA/C | MOEA/D-CMA | MSEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ZDT1 | **0.7002** | 0.6822 | 0.0000† | 0.2035† | 0.5361† | 0.0000† | 0.6038† | 0.0000† | 0.0079† | 0.0000† | 0.5068† | 0.0049† | 0.0023† |
| ZDT2 | **0.4253** | 0.1048† | 0.0000† | 0.0000† | 0.0079† | 0.0000† | 0.1754† | 0.0000† | 0.0000† | 0.0000† | 0.0221† | 0.0000† | 0.0000† |
| ZDT3 | **1.0931** | 0.7418† | 0.0072† | 0.3951† | 0.5006† | 0.0000† | 0.5667† | 0.0000† | 0.0116† | 0.0000† | 0.5762† | 0.0189† | 0.0305† |
| ZDT4 | **0.6502** | 0.3173† | 0.0006† | 0.1056† | 0.3960† | 0.0000† | 0.2314† | 0.0016† | 0.0027† | 0.0417† | 0.3214† | 0.0000† | 0.0056† |
| ZDT6 | **0.4033** | 0.3029† | 0.0033† | 0.0475† | 0.3279† | 0.0000† | 0.3047† | 0.0611† | 0.0188† | 0.0000† | 0.1424† | 0.3676† | 0.0370† |
| MF1 | **0.9100** | 0.4258† | 0.6396† | 0.7881† | 0.2445† | 0.1866† | 0.6812† | 0.8260† | 0.6908† | 0.5993† | 0.6584† | 0.4596† | 0.6819† |
| MF2 | **0.5187** | 0.1750† | 0.2902† | 0.2929† | 0.2088† | 0.2058† | 0.1129† | 0.3280† | 0.1731† | 0.3642† | 0.3285† | 0.1712† | 0.2254† |
| MF3 | **0.6673** | 0.4569† | 0.4621† | 0.5410† | 0.3833† | 0.5065† | 0.5066† | 0.6062† | 0.5336† | 0.5582† | 0.4538† | 0.4724† | 0.6020† |
| MF4 | **0.8536** | 0.2762† | 0.5929† | 0.6207† | 0.2691† | 0.1757† | 0.4940† | 0.8059† | 0.5940† | 0.4560† | 0.5820† | 0.4112† | 0.6563† |
| MF5 | **0.5395** | 0.1808† | 0.1827† | 0.3311† | 0.2094† | 0.2086† | 0.1097† | 0.3198† | 0.2051† | 0.3015† | 0.2001† | 0.1828† | 0.1551† |
| MF6 | **0.6861** | 0.3969† | 0.4734† | 0.6762 | 0.4455† | 0.5865† | 0.4879† | 0.6725 | 0.5751† | 0.6194† | 0.5043† | 0.4983† | 0.5605† |
| MF7 | **0.8727** | 0.2850† | 0.6568† | 0.7212† | 0.2053† | 0.1794† | 0.6550† | 0.7689† | 0.6408† | 0.3750† | 0.6769† | 0.3752† | 0.6486 |
| MF8 | **1.0308** | 0.5726† | 0.3188† | 0.0812† | 0.3756† | 0.3061† | 0.4245† | 0.7435† | 0.5185† | 0.1968† | 0.6495† | 0.0145† | 0.6001† |
| MF9 | **0.5860** | 0.2786† | 0.0000† | 0.0000† | 0.1538† | 0.0530† | 0.1974† | 0.4803† | 0.2496† | 0.0691† | 0.0056† | 0.0000† | 0.0000† |

The results in Tables I and II indicate that the performance of the proposed REMOA is much better than that of the compared algorithms in ZDT test instances in terms of IGD and HV metric. As shown in Fig. 4, the approximate PFs obtained by the proposed REMOA almost converge to the true PF of the ZDT test instances except for ZDT4. The approximate PF obtained by the proposed REMOA still shows promising convergence and good distribution in ZDT4. In contrast, the other compared algorithms cannot converge to the true PF with a smaller number of generations. This means that the proposed algorithm has a faster convergence speed. Once the population with few solutions approaches the PS, it is easy to extend the population and cover well the PF because of the simple PS of ZDT.

The results in Tables I and II also indicate that REMOA achieves the best performance on MF1-MF7 with complex PS except for MF6 in terms of the IGD value. M2M and NSGA-II are slightly outperformed by REMOA on MF6 in terms of IGD. Fig. 5 plots the NSs with the median IGD value obtained by REMOA, NSGA-II, MOEA/D, and SMS-EMOA on MF1-MF3, MF8, and MF9. The results of the other test instances are plotted in the supplementary material. As can be observed from Tables I and II and Fig. 5, the proposed REMOA exhibits the most competitive performance on the

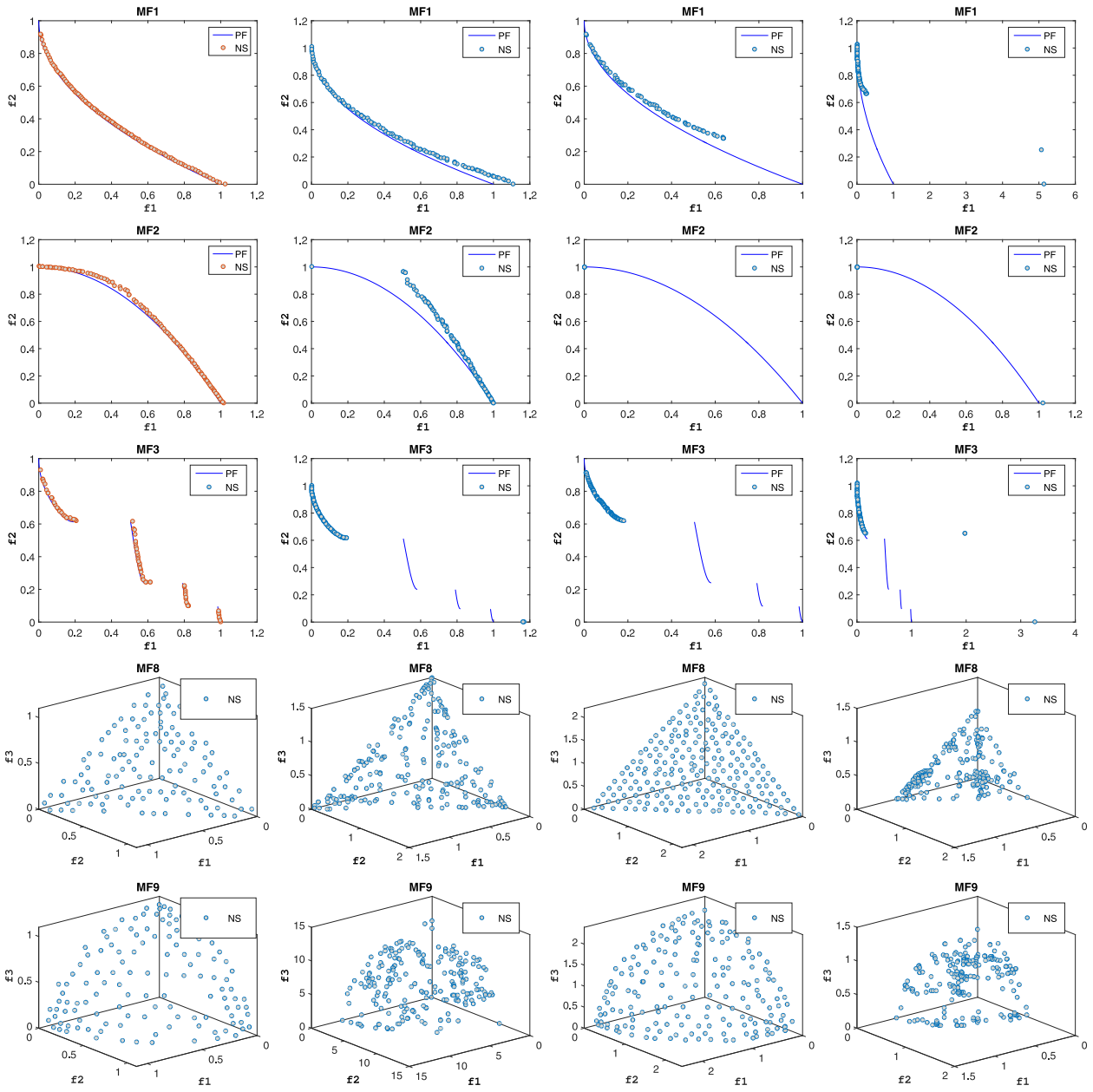


Fig. 5. Final NSs with the median of the IGD value found by REMOA (the first column), NSGA-II (the second column), MOEA/D (the third column), and SMS-EMOA (the fourth column) for MF1–MF3, MF8, and MF9.

test instance with complex PS. Nonetheless, it is not as superior as on ZDT test instances with simple PSs. The reason is that it is more difficult to extend the population to cover well the PF for the problems with complex PS shapes than for the problems with simple PS shapes. Namely, the computational resources for extending the population to cover well the PF are greater. The proposed REMOA is also superior to SMS-EMOAc with increasing population size. SMS-EMOAc is even worse than SMS-EMOA with constant population size on several test problems. The reason is that SMS-EMOAc with a smaller population size cannot maintain the diversity and representativeness of the population. MF8 and MF9 are highly multimodal problems. It is also observed that the approximate PF obtained by the proposed REMOA converges to the true PF, while the other compared algorithms fail to reach the true PFs of these problems.

### F. Sensitivity to the Parameter $\Delta$

To study how the parameter $\Delta$ influences the performance of the proposed algorithm, we have considered three values for $\Delta = 0.05, 0.1, 0.2$, respectively. In our experiments, all parameters are kept the same as the aforementioned experiments except the setting of $\Delta$. Considering the page limit, we only present the plot of the average values of the IGD metric found in 30 independent runs for each $\Delta$ on ZDT1, MF1, and MF8 in Fig. 6. It can be seen that the best parameter value of $\Delta$ is associated with the number of objectives. Nevertheless, the performance of REMOA is relatively stable on parameter

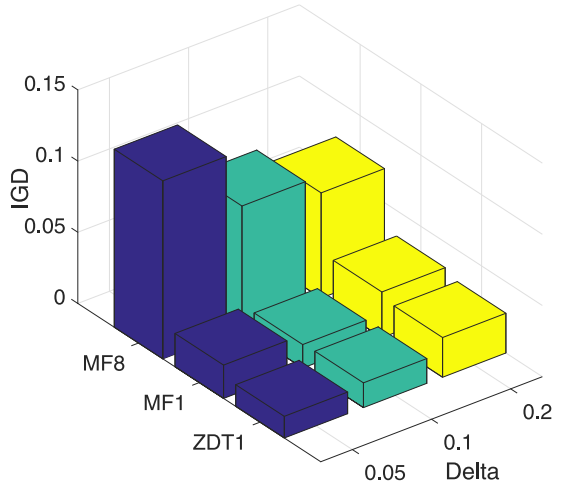


Fig. 6. Parameter sensitivity studies of $\Delta$ on ZDT1, MF1, and MF8.

$\Delta$. For simplicity, it is set to the same value $\Delta = 0.1$ for all test problems in this article.

### *G. Further Investigations of the Proposed Algorithm on Many-Objective Optimization Problems*

We compared the proposed algorithm with nine EMO algorithms on well-known benchmark problem sets, the DTLZ and WFG test suites, and the inverted DTLZ1 and DTLZ2 [65], denoted IDTLZ1 and IDTLZ2. The problems DTLZ1–DTLZ4, IDTLZ1, IDTLZ2, and WFG1–WFG9 with 5, 10, and 15 objectives were used for empirical studies. Due to space limitations, we provide the results in the supplementary file. These results indicate that the proposed algorithm can obtain a competitive result for solving many-objective optimization problems. The performance of REMOA on these test problems is not as outstanding as that for ZDT and MF problems because the PSs of the former problems are not within a small area in the decision space. Another possible cause might be the length of the evolutionary path from a current Pareto-optimal $B$ we have got to the expected solution $A$ is generally large due to the curse of dimensionality. This does not highlight the advantages and necessities of approaching the PF with a smaller population size. The performance of REMOA is not the best one of the problems with irregular PF, for example, IDTLZ1 and IDTLZ2, because the weight vectors used in the proposed algorithm are in a triangular shape.

### *H. Further Investigations of the Proposed Algorithm on Practical Problem: Crash Worthiness Problem*

To investigate the performance of the proposed algorithm in a practical problem, crash worthiness problem [66] is considered in this article. This problem aims to minimize three objectives: 1) the mass of the vehicle; 2) the deceleration during the full-frontal crash; and 3) the toe board intrusion in the offset-frontal crash. The true PF of this practical problem is unknown. As suggested in [17], we obtained an NS set by using NSGA-III with a population size of 7381. It is run for 1000 generations. We choose 1000 points from these NSs, which have the greatest distance from each other to approximate the true PF. The parameters of the proposed algorithm for this problem are the same as used in problems MF8 and MF9. The average IGD value obtained by the proposed algorithm on this problem is 0.1203. This result demonstrates the proposed algorithm can find a representative solution set in the considered practical problem.

## V. Conclusion

In this article, we have presented a rough-to-fine EMO algorithm. A relatively balanced tree has been constructed by using a modified $k$-means algorithm on a set of uniform weight vectors. The search domain is decomposed into some subdomains with the help of the weight vectors, and the subdomains related to descendants are the further decompositions of the subdomains related to ancestors. The proposed REMOA approaches the PF by a population with a smaller size and refines the PF by a population with a larger size. Moreover, the proposed REMOA has a low complexity because we only need to compare each candidate solution with the solutions along the path from the root to a leaf. We have compared the proposed algorithm with 12 state-of-the-art EMO algorithms on ZDT test instances and nine test problems constructed in this article. The experimental results demonstrated the efficacy of the proposed approach.

## References

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems*, 2001, pp. 95–100.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[3] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[4] B. Nicola, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, 2007.

[5] F. Gu and Y.-M. Cheung, "Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 211–225, Apr. 2018.

[6] L. Hai-Lin, F.-Q. Gu, and Y.-M. Cheung, "T-MOEA/D: MOEA/D with objective transform in multi-objective problems," in *Proc. Int. Conf. Inf. Sci. Manage. Eng.*, vol. 2. Shaanxi, China, 2010, pp. 282–285.

[7] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

[8] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 6, pp. 565–588, Dec. 2001.

[9] D. Hadka and P. Reed, "Borg: An auto-adaptive many-objective evolutionary computing framework," *Evol. Comput.*, vol. 21, no. 2, pp. 231–259, May 2013.

[10] T. Glasmachers, B. Naujoks, and G. Rudolph, "Start small, grow big? saving multi-objective function evaluations," in *Proc. 13th Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2014, pp. 579–588.

[11] K. Nag, T. Pal, and N. R. Pal, "ASMiGA: An archive-based steady-state micro genetic algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 40–52, Jan. 2015.

[12] T. Glasmachers, "A fast incremental BSP tree archive for non-dominated points," in *Proc. 9th Int. Conf. Evol. Multi Criterion Optim.*, 2017, pp. 252–266.

[13] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

[14] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surveys*, vol. 48, no. 1, p. 13, 2015.

[15] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 440–462, Jun. 2017.

[16] F. Gu, H.-L. Liu, and K. C. Tan, "A hybrid evolutionary multiobjective optimization algorithm with adaptive multi-fitness assignment," *Soft Comput.*, vol. 19, pp. 3249–3259, Nov. 2015.

[17] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[18] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 264–283, Apr. 2015.

[19] Y.-M. Cheung, F. Gu, and H.-L. Liu, "Objective extraction for many-objective optimization problems: Algorithm and test problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 755–772, Oct. 2016.

[20] L. Chen, H.-L. Liu, K. C. Tan, Y.-M. Cheung, and Y. Wang, "Evolutionary many-objective algorithm using decomposition-based dominance relationship," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 4129–4139, Dec. 2019.

[21] H. Wang and X. Yao, "Corner sort for Pareto-based many-objective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 92–102, Jan. 2014.

[22] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

[23] Y. Zhou, Z. Chen, and J. Zhang, "Ranking vectors by means of the dominance degree matrix," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 34–51, Feb. 2017.

[24] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 445–460, Jun. 2015.

[25] M. Elarbi, S. Bechikh, A. Gupta, L. B. Said, and Y.-S. Ong, "A new decomposition-based NSGA-II for many-objective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 7, pp. 1191–1210, Jul. 2018.

[26] H.-L. Liu, Y. Wang, and Y.-M. Cheung, "A multiobjective evolutionary algorithm using min–max strategy and sphere coordinate transformation," *Intell. Autom. Soft Comput.*, vol. 15, no. 3, pp. 361–384, 2009.

[27] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 450–455, Jun. 2014.

[28] Y. Zhang, R. Yang, J. Zuo, and X. Jing, "Enhancing MOEA/D with uniform population initialization, weight vector design and adjustment using uniform design," *J. Syst. Eng. Electron.*, vol. 26, no. 5, pp. 1010–1022, 2015.

[29] X. He, Y. Zhou, Z. Chen, and Q. Zhang, "Evolutionary many-objective optimization based on dynamical decomposition," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 361–375, Jun. 2019.

[30] Z. He and G. G. Yen, "Many-objective evolutionary algorithms based on coordinated selection strategy," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 220–233, Apr. 2017.

[31] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 3–18, Feb. 2018.

[32] H. Zhang and A. Zhou, "Tree-structured decomposition and adaptation in MOEA/D," in *Parallel Problem Solving From Nature—PPSN XV*. Coimbra, Portugal: Springer, 2018, pp. 359–371.

[33] H. Xu, W. Zeng, D. Zhang, and X. Zeng, "MOEA/HD: A multiobjective evolutionary algorithm based on hierarchical decomposition," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 517–526, Feb. 2019.

[34] Y. Hua, Y. Jin, and K. Hao, "A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular Pareto fronts," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2758–2770, Jul. 2019.

[35] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.

[36] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 16–37, Feb. 2016.

[37] J. Cheng, G. G. Yen, and G. Zhang, "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 592–605, Aug. 2015.

[38] Y. Tian, C. He, R. Cheng, and X. Zhang, "A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Dec. 20, 2020, doi: 10.1109/TSMC.2019.2956288.

[39] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.

[40] S. Jiang, J. Zhang, Y.-S. Ong, A. N. Zhang, and P. S. Tan, "A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2202–2213, Oct. 2015.

[41] L. M. S. Russo and A. P. Francisco, "Quick hypervolume," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 481–502, Aug. 2014.

[42] K. Bringmann and T. Friedrich, "Approximation quality of the hypervolume indicator," *Artif. Intell.*, vol. 195, pp. 265–290, Feb. 2013.

[43] H. Eskandari, C. D. Geiger, and G. B. Lamont, "FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems," in *Proc. 4th Int. Conf. Evol. Multi Criterion Optim.*, 2007, pp. 141–155.

[44] T. Weise, Y. Wu, R. Chiong, K. Tang, and J. Lässig, "Global versus local search: The impact of population sizes on evolutionary algorithm performance," *J. Global Optim.*, vol. 66, no. 3, pp. 511–534, 2016.

[45] F. G. Lobo and C. F. Lima, "Adaptive population sizing schemes in genetic algorithms," in *Parameter Setting in Evolutionary Algorithms*, vol. 54. Heidelberg, Germany: Springer, 2007, pp. 185–204.

[46] P. García-Sánchez *et al.*, "Studying the effect of population size in distributed evolutionary algorithms on heterogeneous clusters," *Appl. Soft Comput.*, vol. 38, pp. 530–547, Jan. 2016.

[47] T. Storch, "On the choice of the parent population size," *Evol. Comput.*, vol. 16, no. 4, pp. 557–578, 2008.

[48] H. Wang, S. Rahnamayan, and Z. Wu, "Adaptive differential evolution with variable population size for solving high-dimensional problems," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 2626–2632.

[49] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.

[50] J. W. Hallam, O. Akman, and F. Akman, "Genetic algorithms with shrinking population size," *Comput. Stat.*, vol. 25, no. 4, pp. 691–705, 2010.

[51] T. Hu, S. Harding, and W. Banzhaf, "Variable population size and evolution acceleration: A case study with a parallel evolutionary algorithm," *Genet. Program. Evol. Mach.*, vol. 11, no. 2, pp. 205–225, 2010.

[52] O. Krause, T. Glasmachers, N. Hansen, and C. Igel, "Unbounded population MO-CMA-ES for the bi-objective BBOB test suite," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2016, pp. 1177–1184.

[53] H. Chen, G. Wu, W. Pedrycz, P. N. Suganthan, L. Xing, and X. Zhu, "An adaptive resource allocation strategy for objective space partition-based multiobjective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 3, pp. 1507–1522, Mar. 2021.

[54] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evol. Comput.*, vol. 22, no. 2, pp. 231–264, Jun. 2014.

[55] F. Gu, H.-L. Liu, and K. C. Tan, "A multiobjective evolutionary algorithm using dynamic weight design method," *Int. J. Innovat. Comput. Inf. Control*, vol. 8, no. 5B, pp. 3677–3688, 2012.

[56] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[57] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Parallel Problem Solving Nat. (PPSN)*, 2004, pp. 832–842.

[58] B. Li, K. Tang, J. Li, and X. Yao, "Stochastic ranking algorithm for many-objective optimization based on multiple indicators," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 924–938, Dec. 2016.

[59] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.

[60] H. Li, Q. Zhang, and J. Deng, "Biased multiobjective optimization and decomposition algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 52–66, Jan. 2017.

[61] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2015.

[62] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.

[63] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evol. Multi Criterion Optim.*, 2005, pp. 62–76.
[64] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, Nov. 2017.
[65] H. Jain and K. Deb, "An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization," in *Proc. Int. Conf. Evol. Multi Criterion Optim.*, 2013, pp. 307–321.
[66] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Struct. Multidiscipl. Optim.*, vol. 35, no. 6, pp. 561–569, 2008.

**Fangqing Gu** received the B.S. degree from Changchun University, Jilin, China, in 2007, the M.S. degree from the Guangdong University of Technology, Guangzhou, China, in 2011, and the Ph.D. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2016.

He joined the School of Applied Mathematics, Guangdong University of Technology, as a Lecturer. His research interests include data mining, machine learning, and evolutionary computation.

**Hai-Lin Liu** (Senior Member, IEEE) received the B.S. degree in mathematics from Henan Normal University, Xinxiang, China, in 1984, the M.S. degree in applied mathematics from Xidian University, Xi'an, China, in 1989, and the Ph.D. degree in control theory and engineering from the South China University of Technology, Guangzhou, China, in 2002.

He is a Full Professor with the School of Applied Mathematics, Guangdong University of Technology, Guangzhou. He is a Postdoctoral Researcher with the Institute of Electronic and Information, South China University of Technology. He has published over 100 research papers in journals and conferences. His research interests include evolutionary computation and optimization, wireless network planning and optimization, and their applications.

Prof. Liu currently serves as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

**Yiu-Ming Cheung** (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2000.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His research interests include machine learning, pattern recognition, visual computing, and optimization.

Prof. Cheung is an FIET and FBCS, and serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, and *Pattern Recognition*, to name a few.

**Minyi Zheng** received the B.S. degree from the South China Normal University, Guangzhou, China, in 2018. She is currently pursuing the master's degree with the School of Applied Mathematics, Guangdong University of Technology, Guangzhou.

Her research interests include data mining and evolutionary computation.