Learning to Approximate: Auto Direction Vector Set Generation for Hypervolume Contribution Approximation

Ke Shang, Senior Member, IEEE, Tianye Shu, and Hisao Ishibuchi, Fellow, IEEE

Abstract—Hypervolume contribution is an important concept in evolutionary multiobjective optimization (EMO). It involves hypervolume-based EMO algorithms and hypervolume subset selection algorithms. Its main drawback is that it is computationally expensive in high-dimensional spaces, which limits its applicability to many-objective optimization. Recently, an R2 indicator variant (i.e., $R^2_{HVC}$ indicator) is proposed to approximate the hypervolume contribution. The $R^2_{HVC}$ indicator uses line segments along a number of direction vectors for hypervolume contribution approximation. It has been shown that different direction vector sets lead to different approximation qualities. In this article, we propose learning to approximate (LtA), a direction vector set generation method for the $R^2_{HVC}$ indicator. The direction vector set is automatically learned from training data. The learned direction vector set can then be used in the $R^2_{HVC}$ indicator to improve its approximation quality. The usefulness of the proposed LtA method is examined by comparing it with other commonly used direction vector set generation methods for the $R^2_{HVC}$ indicator. Experimental results suggest the superiority of LtA over the other methods for generating high-quality direction vector sets.

Index Terms—Approximation, evolutionary multiobjective optimization (EMO), hypervolume contribution, hypervolume indicator.

I. INTRODUCTION

In evolutionary multiobjective optimization (EMO), there are many performance indicators, such as generalization distance (GD) [1], inverted GD (IGD) [2], hypervolume [3], [4], and R2 [5]. Among these performance indicators, the hypervolume indicator is one of the most popular ones. It is able to evaluate the convergence and the diversity of a solution set simultaneously. Thus, it is widely used for the performance evaluation of EMO algorithms. On the other hand, it can also be used for EMO algorithm design. Representative hypervolume-based EMO algorithms include SMS-EMOA [6], [7], FV-MOECA [8], HypE [9], and R2HCA-EMOA [10].

In hypervolume-based EMO algorithms, a multiobjective optimization problem is transformed into a single-objective optimization problem where the single objective is to maximize the hypervolume of the population. In a standard $(\mu + 1)$ hypervolume-based EMO algorithm (e.g., SMS-EMOA), in each generation, one offspring is generated and added into the population, then one individual with the least hypervolume contribution is removed from the population. In this manner, the hypervolume of the population is monotonically nondecreasing as the number of generations increases.

The hypervolume contribution is the key concept in hypervolume-based EMO algorithms. It describes the change of the hypervolume when an individual is added to or removed from the population. Some methods have been proposed to exactly calculate the hypervolume contribution. In [11], a dimension-sweep algorithm is proposed for computing all hypervolume contributions in three dimensions. Their algorithm has a time complexity of $O(\mu \log n)$. In [12], the HVC4D algorithm is proposed to compute all hypervolume contributions in four dimensions. HVC4D has a time complexity of $O(n^2)$. For higher dimensional cases, there exist some exact methods, such as IHSO [13], IWFG [14], and exQHV [15]. However, the runtime of these methods increases exponentially as the dimensionality increases, which limits their applicability to many-objective optimization.

In order to improve the applicability of the hypervolume contribution in many-objective optimization, some hypervolume contribution approximation methods have been proposed [16], [17], [18], [19]. In [16], a point-based method (also known as Monte Carlo (MC) method) was proposed. In this method, a large number of points are sampled in the objective space and the hypervolume contribution is approximated based on the percentage of the points lying inside of the hypervolume contribution region. In [17], a randomized MC (RMC) method was proposed to identify the least hypervolume contributor. By specifying two parameters $\epsilon$ and $\delta$, the hypervolume contribution approximation error is controlled.
the proposed method identifies a solution with contribution at most \((1 + \varepsilon)\) times the minimal contribution with probability at least \((1 - \delta)\). In [18], a line-based method [also known as an R2 indicator variant (i.e., \(R^2_\text{HVC}\) indicator)] was proposed. In this method, a set of line segments with different directions is used to approximate the hypervolume contribution.

Fig. 1 shows the performance comparison results among the above-mentioned three approximation methods. These methods are tested on 1000 solution sets where each solution set has 100 solutions randomly sampled on the Pareto front of 5- and 10-objective DTLZ2. The reference point for hypervolume calculation is set as \(\{1.5, \ldots, 1.5\}\). The horizontal axis “Runtime” means the total time used to handle these 1000 solution sets (except that the runtime of RMC in Fig. 1(b) is the time used to handle a single solution set). The vertical axis “correct identification rate (CIR)” means the percentage of solution sets for which each approximation method correctly identifies the least contributor. The number of points/lines in the point-based/line-based method is set as 100, 200, \ldots, 900 (i.e., we examine nine different settings in each method). The two parameters \(\varepsilon\) and \(\delta\) in the RMC method are set as \(\varepsilon = 1\) and \(\delta = 1\) to minimize its runtime. That is, we cannot further decrease the runtime of the RMC method. We can see that the line-based method clearly dominates the point-based method in terms of the runtime and the CIR value. The RMC method achieves the best CIR value in Fig. 1(a). However, it needs much more runtime than the other two methods. In Fig. 1(b), the RMC method takes about 14 h to handle a single solution set and we cannot obtain its CIR value (since it will take more than one year to handle all solution sets). This means that the use of RMC is unrealistic in high dimensions. Therefore, the line-based method is more flexible to control the balance between the runtime and the CIR value. In this article, we focus on investigating the line-based method.

One important component in the line-based method is the direction vector set, which defines the directions of the line segments. In our previous study [20], we investigated the effect of different direction vector set generation methods on the approximation quality of the \(R^2_\text{HVC}\) indicator. Our results showed that different direction vector sets lead to different approximation qualities. A uniform direction vector set has a worse performance than a nonuniform direction vector set, which is counterintuitive. This motivates us to study what is the best direction vector set for the \(R^2_\text{HVC}\) indicator.

In this article, we focus on high dimensions where exact hypervolume contribution calculation methods become impractical to use. We investigate how to find the best direction vector set for the \(R^2_\text{HVC}\) indicator. Our proposed method, learning to approximate (LtA), is able to automatically train a direction vector set from training data. The trained direction vector set can then be used in the \(R^2_\text{HVC}\) indicator to improve its approximation quality. The main contributions of this article are summarized as follows.

1) We formulate the task of finding the best direction vector set as an optimization problem. The objective function is to maximize the Pearson correlation coefficient between the true and the approximated hypervolume contribution values for training data. The training data is a number of nondominated solution sets.

2) We propose the LtA method to solve the defined optimization problem. The LtA method is a population-based algorithm where each individual is a direction vector. The population evolves in a steady-state manner which guarantees that the objective is monotonically nondecreasing as the number of generations increases.

3) We verify the effectiveness of the LtA method by comparing it with other commonly used direction vector set generation methods based on three different experiments, which allow us to evaluate the performance of the proposed method from different perspectives.

The remainder of this article is organized as follows. Section II presents the preliminaries of the study. Section III introduces the proposed method, LtA, to automatically generate a direction vector set. Section IV conducts the experimental studies. Section V gives further investigations, and Section VI concludes this article.

II. PRELIMINARIES

A. Basic Definitions

In EMO, the hypervolume indicator is defined as follows. Given a solution set \(S\) in the objective space, the hypervolume of \(S\) is defined as

\[
HV(S, r) = \mathcal{L}\left(\bigcup_{s\in S}\{s'|s < s' < r\}\right)
\]

where \(\mathcal{L}(.)\) is the Lebesgue measure of a set, \(r\) is the reference point which is dominated by all solutions in \(S\), and \(s \prec s'\) denotes that \(s\) Pareto dominates \(s'\) (i.e., \(s_i \leq s'_i\) for all \(i = 1, \ldots, m\) and \(s_j < s'_j\) for at least one \(j = 1, \ldots, m\) in the minimization case, where \(m\) is the number of objectives).

Fig. 2(a) illustrates the hypervolume of a solution set \(S = \{a^1, a^2, a^3\}\) in a 2-D objective space, where each objective is to be minimized.

The hypervolume contribution is defined based on the hypervolume indicator. It describes the change of the hypervolume when a solution is added to or removed from the solution set. Formally, given a solution \(s\in S\), the hypervolume
The length $l(\lambda, s, S, r)$ can be calculated as follows:

$$l(\lambda, s, S, r) = \min_{\lambda \in \Lambda} \left\{ g^{2\text{ch}}(s' | \lambda, s) \right\}$$

where $s'$ is the vector obtained by shifting $s$ along the direction vector $\lambda$. The $g^{2\text{ch}}$ function in (5) is defined for minimization problems as

$$g^{2\text{ch}}(s' | \lambda, s) = \max_{j \in \{1, \ldots, m\}} \left\{ \frac{s_j' - s_j}{\lambda_j} \right\}.$$  (6)

For maximization problems, it is defined as

$$g^{2\text{ch}}(s' | \lambda, s) = \min_{j \in \{1, \ldots, m\}} \left\{ \frac{s_j - s_j'}{\lambda_j} \right\}.$$  (7)

The $g^{\text{match}}$ function in (5) is defined for both minimization and maximization problems as

$$g^{\text{match}}(r | \lambda, s) = \min_{j \in \{1, \ldots, m\}} \left\{ \frac{|s_j - r_j|}{\lambda_j} \right\}.$$  (8)

Based on the above formulations, it is easy to obtain that the time complexity of approximating the hypervolume contribution of a solution $s \not\in S$ to a solution set $S$ is $O(m|\Lambda||S|)$. This time complexity suggests the high applicability of the line-based method to many-objective problems since the number of lines can be arbitrarily specified. Our computational experiments show that the line-based method works well on many-objective problems with 5, 8, and 10 objectives without increasing the number of lines. For more detailed explanations of the $R^2_{\text{HVC}}$ indicator, please refer to [18].

C. Direction Vector Set Generation Methods

In the $R^2_{\text{HVC}}$ indicator in (4), a direction vector set $\Lambda$ is used to specify the directions of the line segments for the hypervolume contribution approximation. Intuitively, one may think that a uniformly distributed direction vector set is a good choice. However, this intuition is not correct. In our previous study in [20], we investigated the effects of five different direction vector set generation methods on the performance of the $R^2_{\text{HVC}}$ indicator. Our experimental results showed that a uniform direction vector set has a worse performance than a nonuniform direction vector set.

The five generation methods considered in [20] are briefly explained as follows.

1) DAS: The Das and Dennis’s (DAS) method [21] is widely used in decomposition-based EMO algorithms (e.g., MOEA/D [22] and NSGA-III [23]) to generate weight vectors. DAS generates all weight vectors $w = (w_1, w_2, \ldots, w_m)$ satisfying the following relations:

$$\sum_{i=1}^{m} w_i = 1 \quad \text{and} \quad w_i \in \left\{ 0, \frac{1}{H}, \frac{2}{H}, \ldots, 1 \right\} \quad \text{for} \quad i = 1, 2, \ldots, m$$

where $m$ is the number of objectives and $H$ is a positive integer. The total number of generated weight vectors is $\binom{H+m-1}{m-1}$.
Therefore, we cannot arbitrarily specify the number of weight vectors for the DAS method.

Based on the weight vector \( \mathbf{w} \), we obtain the direction vector \( \lambda = \mathbf{w} / \| \mathbf{w} \|_2 \).

Note that most generated weight vectors by DAS are boundary vectors in many-objective cases. In order to generate more inside vectors, we can use the two-layered DAS method [24]. For more information, please refer to [24].

2) UNV: The unit normal vector (UNV) method [19] is a sampling method which samples points uniformly on the unit hypersphere. The direction vectors are generated by uniformly sampling points on the unit hypersphere as follows. First, we randomly sample points \( \mathbf{x} \) (i.e., \( m \)-dimensional vectors) according to the \( m \)-dimensional normal distribution \( \mathcal{N}_m(0, I_m) \), then the corresponding direction vectors are obtained by \( \lambda = |\mathbf{x}| / \| \mathbf{x} \|_2 \).

3) JAS: The JAS method [25] is a probabilistic filling method for generating a set of weight vectors on the simplex. The basic idea of JAS is to sample each component of a weight vector uniformly one by one on the simplex. For a weight vector \( \mathbf{w} \), its first component \( w_1 \) is sampled based on \( w_1 = 1 - n \sqrt{u_1} \), where \( u_1 \) is uniformly sampled in \([0, 1]\). This ensures that \( w_1 \) is uniformly sampled on the simplex. Then, the next component is sampled based on \( w_k = (1 - \sum_{j=1}^{k-1} w_j) (1 - n \sqrt{u_k}) \) for \( k = 2, \ldots, m-1 \), where \( u_k \) is uniformly sampled in \([0, 1]\). Finally, the last component \( w_m \) is computed as \( w_m = 1 - \sum_{j=1}^{m-1} w_j \). The above procedure is iterated to generate a prespecified number of weight vectors. For more detailed explanations of the JAS method, please refer to [25].

Based on the weight vector \( \mathbf{w} \), we obtain the direction vector \( \lambda = \mathbf{w} / \| \mathbf{w} \|_2 \).

4) MSS-D: The maximally sparse selection (MSS) method [26] starts with a large point set \( S \) on the unit hypersphere. First, the direction vector set \( \Lambda \) is initialized as the \( m \) extreme points \((1, 0, \ldots, 0)\), \((0, 1, \ldots, 0)\), \((0, 0, \ldots, 1)\). Then, the point in \( S \) with the largest distance to \( \Lambda \) is selected and added to \( \Lambda \). This procedure is repeated until a prespecified number of direction vectors are included in \( \Lambda \). If \( S \) is generated by the DAS method, the MSS method is called MSS-D.

5) MSS-U: When \( S \) is generated by the UNV method, the MSS method is called MSS-U.

In addition to the above five methods, we also consider the following method in this article.

6) Kmeans-U: This method is described in [27]. First, a large point set \( S \) on the unit hypersphere is generated by the UNV method. Then, the \( k \)-means clustering [28] is applied to \( S \) to select a prespecified number of points as the direction vector set. In the \( k \)-means clustering, the number of clusters is the same as the number of direction vectors to be selected. The closest point to each cluster center is selected.

Fig. 4(a)–(f) shows 91 direction vectors generated by each method in the 3-D objective space.\(^2\) We can see that the

\(^2\)We use 3-D direction vector sets for visual illustration whereas the focus of this article is high dimensions.

UNV and JAS methods generate nonuniform direction vectors whereas uniform direction vectors are generated by the other methods. However, it was reported in [20] that the UNV and JAS methods have better approximation quality than DAS, MSS-D, and MSS-U. In this article, all the six generation methods of direction vectors in Fig. 4 are compared with the proposed method.

Fig. 4(f) shows 91 direction vectors generated by Kmeans-U. As shown in Fig. 4(f), the direction vectors generated by this method are uniformly distributed. However, different from the DAS, MSS-D, and MSS-U methods, no boundary direction vector is generated by Kmeans-U. In fact, the direction vector set distribution in Fig. 4(f) is closely related to IGD minimization when \( S \) is used as the reference point set [27].

III. LEARNING TO APPROXIMATE

A. Mathematical Modeling

Our task is to find a direction vector set which has high approximation quality for the \( R^2_{HVC} \) indicator. Mathematically, we can formulate this task as an optimization problem as follows:

\[
\text{Maximize : } Q(\Lambda) \\
\text{Subject to : } \Lambda = \{ \lambda^i | i = 1, \ldots, n \}, \quad \| \lambda^i \|_2 = 1 \\
\lambda^i_j \geq 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m \quad (10)
\]

where \( Q \) is a function which can evaluate the approximation quality of the direction vector set \( \Lambda \), \( n \) is the number of direction vectors in \( \Lambda \), and \( m \) is the number of objectives.

Now, we have two questions with respect to the optimization problem in (10).

1) \( Q_1 \): How to define the objective function \( Q \)?

2) \( Q_2 \): How to solve the optimization problem in (10)?

If the above two questions can be solved, we can generate a good direction vector set for the \( R^2_{HVC} \) indicator. In the next two sections, we present our solutions with respect to the above two questions, respectively.
B. How to Define the Objective Function \( Q \)?

The objective function \( Q \) is used to evaluate the approximation quality of the direction vector set \( \Lambda \). However, we cannot evaluate the quality of \( \Lambda \) without data (i.e., a given set of solutions with their true and approximated hypervolume contribution values). Thus, we need to prepare data in order to define the objective function \( Q \).

Suppose we have a nondominated solution set \( S = \{s_1, s_2, \ldots, s_N\} \), we can calculate the hypervolume contribution and the \( R_2^{\text{HVC}} \) value of each solution in \( S \) and obtain a set of approximation results as follows in an \( N \times 2 \) matrix form:

\[
D(S) = \begin{bmatrix}
\text{HVC}(s_1, S, r) & R_2^{\text{HVC}}(s_1, S, r) \\
\text{HVC}(s_2, S, r) & R_2^{\text{HVC}}(s_2, S, r) \\
\vdots & \vdots \\
\text{HVC}(s_N, S, r) & R_2^{\text{HVC}}(s_N, S, r)
\end{bmatrix}.
\]

If the \( R_2^{\text{HVC}} \) indicator is a perfect approximator, we can expect that the relation between HVC and \( R_2^{\text{HVC}} \) is strictly linear. However, due to the approximation error, this relation can hardly hold. Therefore, one intuitive way is to use the degree of linearity between HVC and \( R_2^{\text{HVC}} \) as the objective function \( Q \). The Pearson correlation coefficient \([29]\) can be used for this purpose. Given the data set \( D(S) \), we define the objective function \( Q \) as follows:

\[
Q(\Lambda) = \frac{\sum_{i=1}^{N} (D_{i,1} - \overline{D}_{1}) (D_{i,2} - \overline{D}_{2})}{\sqrt{\sum_{i=1}^{N} (D_{i,1} - \overline{D}_{1})^2} \sqrt{\sum_{i=1}^{N} (D_{i,2} - \overline{D}_{2})^2}}
\]

where \( \overline{D}_{1,1} \) and \( \overline{D}_{1,2} \) represent the average values of the first and second columns of \( D(S) \), respectively.

The value of \( Q \) lies in \([-1, 1]\) where a higher value indicates a better linear relation between HVC and \( R_2^{\text{HVC}} \). We can then optimize the direction vector set \( \Lambda \) in order to maximize \( Q \). Fig. 5 illustrates the \( Q \) values of different data sets. In Fig. 5(a), there is no clear linear relation between HVC and \( R_2^{\text{HVC}} \), and \( Q \) has a low value 0.2322. In Fig. 5(b), the linear relation between HVC and \( R_2^{\text{HVC}} \) becomes clearer, and \( Q \) is increased to 0.6603. In Fig. 5(c), we can observe a good linear relation between HVC and \( R_2^{\text{HVC}} \), and \( Q \) reaches to a high value 0.9915.

C. How to Solve the Optimization Problem in (10)?

We can see that (10) is a set-based optimization problem where the decision variables are a set of direction vectors and the objective function is a complex and nonlinear function. It is not easy to use a gradient-based optimizer like BFGS to solve (10) due to the high dimensionality of the optimization problem in (10) where the number of decision variables is \( nm \) (which is 525–1100 in our computational experiments). Furthermore, the partial derivatives \( \partial Q/\partial \lambda_i \) for \( i = 1, \ldots, n \) and \( j = 1, \ldots, m \) are not easy to calculate. A natural idea is to use a population-based method where each individual in the population is a direction vector. Motivated by SMS-EMOA [6, 7], a classical EMO algorithm which aims to maximize the hypervolume of the population, we develop a population-based algorithm for solving (10). The proposed method, named LtA, is shown in Algorithm 1.

In Algorithm 1, the hypervolume contribution of each solution in solution set \( S \) is calculated (line 1). Then, the direction vector set \( \Lambda \) is initialized using the UNV method (line 2). After that, the direction vector set \( \Lambda \) is optimized in a steady-state manner for a maximum iteration number (lines 4–14). In each iteration, one direction vector is generated using the UNV method and added into \( \Lambda \) (lines 5 and 6). Now, there are \( n+1 \) direction vectors in \( \Lambda \). We need to remove one direction vector from \( \Lambda \). The basic idea is to remove the direction vector so that the rest direction vectors in \( \Lambda \) can achieve the best objective function value (i.e., maximum \( Q \) value). In order to do this, for each direction vector in \( \Lambda \), the corresponding \( Q \) value after tentatively removing it is calculated (lines 7–19). Finally, the direction vector which leads to the best \( Q \) value by its removal is identified and removed from \( \Lambda \) (lines 11 and 12).

We name Algorithm 1 LtA since \( \Lambda \) is learned based on a solution set \( S \). From \( S \), we can obtain data set \( D(S) \), which is used to evaluate \( \Lambda \) by calculating \( Q(\Lambda) \). It is clear that a single solution set \( S \) is not enough to learn a good direction.
vector set for various multiobjective problems since it is likely that the learned direction vector set is overfitted to \( S \) (i.e., the direction vector set has a good \( Q \) value for solution set \( S \) but not for other solution sets). In order to solve this issue, we improve Algorithm 1 as follows.

1) Prepare a number of different solution sets \( \{S_1, S_2, \ldots, S_L\} \) instead of a single solution set \( S \).
2) For these \( L \) solution sets, we can get \( L \) data sets \( \{D(S_1), D(S_2), \ldots, D(S_L)\} \).
3) For these \( L \) data sets, we can calculate \( L \) objective function values \( \{Q_1(\Lambda), Q_2(\Lambda), \ldots, Q_L(\Lambda)\} \).
4) Calculate \( Q(\Lambda) \) as \( Q(\Lambda) = (1/L) \sum_{i=1}^{L} Q_i(\Lambda) \) in line 9.

Based on the above improvement, we can expect that the learned direction vector set is not overfitted to a specific solution set and has a robust performance across different solution sets.

D. Discussion

We have the following discussions with respect to the LtA method.

1) How to Prepare Training Solution Sets for LtA?: For the training solution sets \( \{S_1, S_2, \ldots, S_L\} \), we need to specify two parameters: \( L \) (the number of solution sets) and \( N \) (the number of solutions in each solution set). These two parameters determine the overall size of the training solution sets. In this article, we set \( L = 100 \) and \( N = 100 \).

In principle, any solution sets can be used for training. In order to make the training more efficient and robust, these \( L \) solution sets need to be as diverse as possible. We will describe how to generate diverse solution sets for training in Section IV-A.

2) Why Is Pearson Correlation Coefficient Used as the Objective Function \( Q \)?: There are many ways to define the approximation quality of the \( R^2_{HVC} \) indicator. We choose the Pearson correlation coefficient as the objective function \( Q \) since it intuitively defines the linearity between the true and approximated hypervolume contribution values. The other reason is that it is sensitive to the change of the direction vector set. As shown in Algorithm 1, in each iteration, the direction vector set is slightly changed (i.e., only one direction vector is changed and the other direction vectors remain unchanged). If the objective function \( Q \) is insensitive to the slight change of the direction vector set, the training will become difficult. Using the Pearson correlation coefficient as the objective function \( Q \), the training of the direction vector set is very efficient.

3) What Is the Time Complexity of LtA?: The time complexity of LtA is analyzed as follows. Let us consider a single training solution set \( S \) in LtA. The main time cost is the while loop in Algorithm 1. In the while loop, the most time-consuming part is the for loop (lines 7–10). In order to efficiently calculate the \( R^2_{HVC} \) value in line 8, we use the tensor structure introduced in [10]. The tensor structure stores the information for the calculation of \( R^2_{HVC} \). For the detailed explanation of the tensor structure, please refer to [10].

The size of the tensor structure is \( N \times (n+1) \times N \), where \( N \) is the size of the solution set \( S \) and \( n \) is the size of the direction vector set \( \Lambda \). The tensor structure is updated in \( O(N \times N \times m) \) once a new direction vector is generated (line 6). Before entering the for loop, the minimum value of each column of the tensor structure is calculated in \( O(N \times n \times N) \) for further use. In each iteration of the for loop, the \( R^2_{HVC} \) values can be obtained in \( O(n \times N) \) (line 8). Thus, the for loop is run in \( O(n \times n \times N) \). In summary, the time complexity of one iteration of LtA is \( O(N^2 m + N^2 n + n^2 N) \).

If we have \( L \) training solution sets, the time complexity of one iteration of LtA is \( O(L(N^2 m + N^2 n + n^2 N)) \).

IV. Experiments

A. Learning Process of LtA

First, we show the learning process of LtA. We consider 5-, 8-, and 10-objective cases.

To generate training solution sets \( \{S_1, S_2, \ldots, S_L\} \) for each case, we set \( L = 100 \) and \( N = 100 \). That is, we generate 100 solution sets with 100 solutions in each solution set. Solutions in half solution sets (i.e., \( \{S_1, S_2, \ldots, S_{50}\} \)) are randomly generated on the triangular Pareto front \( \sum_{i=1}^{m-p} f_i = 1 \), \( f_i \geq 0 \) for \( i = 1, \ldots, m \), where \( p \) is randomly sampled in \([0.5, 2] \) for each solution set. Solutions in the other half solution sets (i.e., \( \{S_{51}, S_{52}, \ldots, S_{100}\} \)) are randomly generated on the inverted triangular Pareto front \( \sum_{i=1}^{m-p} (1-f_i)^p = 1 \), \( 0 \leq f_i \leq 1 \) for \( i = 1, \ldots, m \), where \( p \) is randomly sampled in \([0.5, 2] \) for each solution set. In our experiments, \( p \) controls the curvature of the Pareto fronts. Thus, 100 solution sets are generated on 100 Pareto fronts with different curvatures (conca, linear, and convex) and shapes (triangular and inverted triangular), which guarantees the training solution sets diverse.

LtA is applied to the generated 100 solution sets in each case under the following settings. The number of direction vectors is set as 105, 120, and 110 in 5-, 8-, and 10-objective cases, respectively, since these settings are valid for the DAS method. Table I shows the settings in the DAS method. The maximum number of iterations of LtA is set as 10,000. In LtA, the reference point for hypervolume calculation is specified as \( r = (1.2, \ldots, 1.2) \) independent of the number of objectives. We perform 20 independent runs of LtA for each case (i.e., 20 direction vector sets are obtained by LtA for each case).

Fig. 6 shows the learning curve of LtA in 5-, 8-, and 10-objective cases. The blue line is the average \( Q \) value over 20 runs, and the gray area denotes the standard deviation. We can see that the \( Q \) value is monotonically increasing as the iteration

| \( m \) | \( H \) (boundary) | \( H \) (inner) | Total number |
|---|---|---|---|
| 5 | 4 | 3 | 105 |
| 8 | 3 | 0 | 120 |
| 10 | 2 | 2 | 110 |

3It is not suggested to use a large number of direction vectors in the \( R^2_{HVC} \) indicator. This is because a small number of direction vectors is enough to achieve a good approximation quality as shown in [18]. This is also because the \( R^2_{HVC} \) indicator with a small number of direction vectors can be efficiently calculated even for many-objective problems.
We can also observe that as the number of objectives increases, the $Q$ value at the end of the learning process decreases. This shows that the learning process becomes more difficult as the number of objectives increases. However, the $Q$ value of the initial direction vector set also decreases with the increase in the number of objectives, and $Q > 0.9$ is achieved in all cases. These observations show the success of the learning process of LtA.

The learned direction vector set in a single run of LtA for each case is shown in Fig. 7. It is not easy to examine the success of the learning process of LtA. (a) 5-objective case. (b) 8-objective case. (c) 10-objective case.

Table II shows the mean Spacing values of the direction vector sets generated by different methods. We can see that the direction vector sets learned by LtA for each objective case do not have the lowest mean Spacing value, which indicates that they are not the most uniform ones. MSS-U and Kmeans-U can generate more uniform direction vector sets. UNV and JAS generate less uniform direction vector sets. These observations are consistent with the visual examination results in Fig. 4.

In the next three sections, we compare the learned direction vector sets by LtA with the direction vector sets generated by the six methods described in Section II-C through three experiments.

**B. Experiment 1: Correct Identification Rate**

In the first experiment, we use the CIR [18] to evaluate different direction vector set generation methods. The CIR is a metric which can evaluate the ability of the $R_2^{HVC}$ indicator to identify the least hypervolume contributor in a solution set. To be more specific, suppose that we have $M$ solution sets $\{T_1, T_2, \ldots, T_M\}$. In each solution set, the true hypervolume contribution of each solution is calculated and the solution with the minimum hypervolume contribution is identified. Using a direction vector set, the approximated hypervolume contribution $R_2^{HVC}$ of each solution is calculated in each solution set. Then, the solution with the minimum $R_2^{HVC}$ value is identified. Suppose that the solution with the minimum hypervolume contribution is correctly identified using the approximate calculation for $M'$ out of the $M$ solution sets. Then, the CIR value is defined as $M'/M$. A higher value of CIR indicates a better approximation quality of the $R_2^{HVC}$ indicator.

1) **Experimental Settings:** We consider six types of Pareto fronts: linear, concave, and convex Pareto fronts with triangular and inverted triangular shapes. For the Pareto fronts with triangular shape, the mathematical expression is $\sum_{i=1}^{m} d_i = 1$ and $f_i \geq 0$ for $i = 1, \ldots, m$, where $p = 1, 2$, and 0.5 correspond to linear, concave, and convex Pareto fronts, respectively. For the Pareto fronts with inverted triangular shape, the mathematical expression is $\sum_{i=1}^{m} (1 - f_i)p = 1$ and $0 \leq f_i \leq 1$ for $i = 1, \ldots, m$, where $p = 1, 2$, and 0.5 correspond to linear, convex, and concave Pareto fronts, respectively. It should be noted that 50 training solution sets were generated from each of triangular and inverted triangular.

| Method       | 5-objective | 8-objective | 10-objective |
|--------------|-------------|-------------|--------------|
| DAS          | 0.1107      | 0.2364      | 0.0552       |
| UNV          | 0.1189      | 0.1532      | 0.1705       |
| JAS          | 0.1210      | 0.1698      | 0.1864       |
| MSS-D        | 0.0582      | 0.1137      | 0.1487       |
| MSS-U        | 0.0562      | 0.0913      | 0.1120       |
| Kmeans-U     | 0.0461      | 0.0890      | 0.1112       |
| LtA          | 0.1026      | 0.1383      | 0.1520       |

where $\bar{d} = (1/n) \sum_{i=1}^{n} d_i$ and $d_i = \min_{\lambda \in \Lambda_1} \| \lambda_i - \lambda \|_1$. The lower the Spacing value, the better the uniformity of a direction vector set.
Pareto fronts in the LtA method where the value of \( p \) was randomly sampled in \([0, 2]\) as described in Section IV-A.

For each type of Pareto front, 100 solutions are randomly sampled on the Pareto front to form a solution set. We generate 100 solution sets for each type of Pareto front. The reference point is set as 1.2 times the nadir point of the Pareto front for hypervolume calculation (i.e., the reference point \( r = (1.2, \ldots, 1.2) \) is used since the ideal and nadir points are \((0, \ldots, 0)\) and \((1, \ldots, 1)\) for all Pareto fronts used in this section).

Each direction vector set generation method is run 20 times on each solution set, and the average CIR is recorded.

2) Results: The experimental results are shown in Table III (detailed results are provided in Table I in the supplementary material). We can see that the LtA method achieves the best overall performance. The UNV and JAS methods outperform the DAS, MSS-U, MSS-D, and Kmeans-U methods, which is consistent with the results obtained in [20]. The counterintuitive observation is that a uniform direction vector set does not have a good CIR performance. In Table III, the overall top three methods are LtA, UNV, and JAS. As shown in Table II, these three methods generate nonuniform direction vectors (whereas all the other methods generate uniform direction vectors). A possible reason for this counterintuitive observation is as follows: the hypervolume contribution of a solution has a complex and irregular shape in three and higher dimensional spaces [32], which makes the line-based approximation difficult. A uniform direction vector set cannot approximate the hypervolume contribution as expected in such a complex and irregular space.

C. Experiment 2: GAHSS

In the second experiment, we use GAHSS\(^4\) [33] to compare different direction vector set generation methods. GAHSS is a greedy approximated hypervolume subset selection algorithm for many-objective optimization. GAHSS follows the framework of the standard greedy hypervolume subset selection (GHSS) [34], [35]. In GHSS, a subset is selected from a candidate solution set in a greedy manner. Initially, the subset is empty. In each step, the solution in the candidate solution set with the largest hypervolume contribution to the subset is added into the subset. This procedure is repeated until the subset has the prespecified number of solutions. In GAHSS, the \( R_{HVC}^2 \) indicator is used to approximate the hypervolume contribution.

1) Experimental Settings: We consider six types of Pareto fronts which are the same as described in Section IV-B1. From each of the six Pareto fronts, we randomly sample 10,000 solutions to form a candidate solution set. The number of objectives is set as 5, 8, and 10. As a result, we have 6 (Pareto fronts) \( \times \) 3 (specifications of the number of objectives) = 18 candidate solution sets.

We select 100 solutions from each candidate solution set using GAHSS. In GAHSS, the reference point is set as 1.2 times the nadir point of the candidate solution set. For the hypervolume performance evaluation of the subsets, we use the same reference point specification.

GAHSS equipped with a direction vector set generated by each method is performed 20 times on each candidate solution set. Furthermore, the results are analyzed by the Wilcoxon rank-sum test with a significance level of 0.05 to determine whether one method shows a statistically significant difference with the other, where “\(+\), “\(-\),” and “\(\approx\)” indicate that the proposed LtA method is “significantly better than,” “significantly worse than,” and “statistically similar to” the compared method, respectively.

2) Results: The experimental results are shown in Table IV (detailed results are provided in Table II in the supplementary material). We can see that the LtA method achieves the best overall performance. The UNV and Kmeans-U methods are worse than the LtA method, but better than the JAS, DAS, MSS-U, and MSS-D methods. One observation is that the Kmeans-U method has a poor CIR performance in Table III but a good hypervolume performance in Table IV. This shows that a single experiment cannot fully reflect the performance of a direction vector set generation method. We need to examine the performance of a method from different perspectives. Our proposed LtA method always performs the best in the above two different experiments, which shows its robustness across different tasks.

D. Experiment 3: R2HCA-EMOA

In the third experiment, we use R2HCA-EMOA\(^5\) [10] to compare different direction vector set generation methods.

\(^4\)The source code of GAHSS is available at: https://github.com/HisaoLabSUSTC/GAHSS.

\(^5\)The source code of R2HCA-EMOA is available at: https://github.com/HisaoLabSUSTC/R2HCA-EMOA.
R2HCA-EMOA is a new hypervolume-based EMO algorithm for many-objective optimization. It is similar to SMS-EMOA. The difference is that the $R^2_{\text{HVC}}$ indicator is used in R2HCA-EMOA to approximate the hypervolume contribution. In each generation of R2HCA-EMOA, one offspring is generated and added into the population, and one individual with the least $R^2_{\text{HVC}}$ value is removed from the population. In the original R2HCA-EMOA paper, the UNV method is used for direction vector set generation. As reported in [10], R2HCA-EMOA achieves better hypervolume performance compared with several state-of-the-art EMO algorithms. In particular, R2HCA-EMOA is better than HypE in terms of both runtime and hypervolume performance.

1) **Experimental Settings:** We apply R2HCA-EMOA equipped with different direction vector sets to multi/many-objective optimization problems. We choose DTLZ1–4 [36], WFG1–9 [37], and their minus versions (i.e., MinusDTLZ1–4 and MinusWFG1–9) [38] as test problems, which are the same as in [10]. We follow the experimental settings in [10], which are listed as follows.

1. The population size is set as 100 for 5-, 8-, and 10-objective cases.
2. The maximum number of solution evaluations is set to 100 000 for DTLZ1, DTLZ3, WFG1, and their minus versions; these problems are difficult for an EMO algorithm to converge within a small number of solution evaluations. It is set to 30 000 for the other test problems.
3. For the hypervolume calculation in the performance evaluation, we first normalize the obtained solutions by R2HCA-EMOA based on the true Pareto front. Then, the reference point is set as $r = (1.2, \ldots, 1.2)$.
4. Each method is run 20 times on each test problem. The results are analyzed by the Wilcoxon rank-sum test with a significance level of 0.05 to determine whether one method shows a statistically significant difference with the other, where $+/-\approx$ indicate that the proposed LtA method is significantly better than, significantly worse than, and statistically similar to the compared method, respectively.

2) **Results:** The experimental results are shown in Tables V and VI (detailed results are provided in Tables III and IV in the supplementary material). Table V shows the results on DTLZ and WFG test problems, and Table VI shows the results on MinusDTLZ and MinusWFG test problems.

### Table V

| Method     | LtA | DAS | UNV | JAS | MSS-D | MSS-U | Kmeans-U |
|------------|-----|-----|-----|-----|-------|-------|----------|
| Average Rank | 1.85 | 6.13 | 2.38 | 2.90 | 6.18  | 5.23  | 3.33     |
| $+/-\approx$ | -    | 37/2/0 | 19/2/18 | 24/5/10 | 37/2/0 | 36/2/1 | 31/4/4   |

### Table VI

| Method     | LtA | DAS | UNV | JAS | MSS-D | MSS-U | Kmeans-U |
|------------|-----|-----|-----|-----|-------|-------|----------|
| Average Rank | 1.36 | 6.77 | 2.44 | 3.03 | 6.15  | 4.41  | 3.85     |
| $+/-\approx$ | -    | 39/0/0 | 34/2/3 | 33/4/2 | 39/0/0 | 34/4/1 | 37/2/0   |

From both tables, we can see that the LtA method achieves the best overall performance. The UNV method performs the second best. On the contrary, the DAS, MSS-U, and MSS-D are the worst three methods. These three methods also perform badly in the previous two experiments.

From the above three different experiments, we can see that the LtA method can always achieve the best overall performance, which shows the superiority of the learned direction vector sets over the others for hypervolume contribution approximation.

## V. Further Investigations

### A. Effect of the Training Solution Sets

In Section III-C, we claimed that a single training solution set is not enough for LtA to train a good direction vector set. In this section, we perform a simple experiment to verify our claim. We follow the experimental settings in Section IV-A. However, instead of using 100 solution sets $\{S_1, S_2, \ldots, S_{100}\}$ for training, we use only a single solution set $S_i$, where $i$ is randomly chosen from 1 to 100 (i.e., $S_i$ is randomly selected from $\{S_1, S_2, \ldots, S_{100}\}$). The chosen $S_i$ used in our experiments is on a concave inverted triangular Pareto front. The LtA method using a single training solution set is denoted as LtA-S. The 5-objective case is considered for illustration.

Fig. 8(a) shows the learning curve of LtA-S over 20 independent runs and Fig. 8(b) shows the learned direction vector set in a single run of LtA-S. We compute the Spacing indicator of the direction vector sets learned by LtA-S. The value is 0.1288. Compared with 0.1026 achieved by LtA as shown in Table II, we can see that the direction vector sets learned by LtA-S become less uniform.

We perform Experiment 1 (i.e., CIR in Section IV-B) to evaluate the performance of the learned direction vector sets by LtA-S. Fig. 9 shows the comparison results among different methods. We can observe that compared with LtA, the performance of LtA-S is degraded. This shows that a single training solution set is not as good as a set of diverse training solution sets. However, we can also observe that LtA-S outperforms all the other methods in most cases.
Fig. 8. Learning curve of LtA-S over 20 independent runs and the learned direction vector set in a single run of LtA-S in 5-objective case. (a) Learning curve. (b) Learned direction vector set.

Fig. 9. CIR of the $R^2_HVC$ indicator with different direction vector sets on different solution sets in 5-objective case. All the results are based on 20 runs. a single training solution set is used in LtA, better direction vector sets are obtained than the other methods in most cases. This reflects the usefulness of LtA for auto-direction vector set generation.

B. Nonuniformity of the Learned Direction Vector Sets

We have shown that the uniform direction vector sets generated by DAS, MSS-D, MSS-U, and Kmeans-U perform worse than the nonuniform direction vector sets generated by UNV, JAS, and LtA (except that Kmeans-U performs better than UNV and JAS in Experiment 2). It seems that a uniform direction vector set is not a good choice for the $R^2_HVC$ indicator. In order to further verify this counterintuitive conclusion, we perform additional experiments to further examine whether the uniformity improvement of the learned direction vector sets by LtA can improve their approximation quality. More specifically, we perform the following experiment: we modify the learned direction vector sets by LtA to make them more uniform and test their approximation quality.

Given a learned direction vector set $\Lambda$ by LtA, the modification is performed as follows. Step 1: Randomly generate 10,000 direction vectors by the UNV method as a candidate direction vector set $C$. Step 2: Identify the closest two direction vectors in $\Lambda$ and randomly remove one of them. Step 3: Find the direction vector in $C$ with the largest distance to $\Lambda$ and add this direction vector to $\Lambda$. Step 4: Repeat steps 2 and 3 ten times. From the above procedure, we know that ten direction vectors are modified. The modified direction vector set will become more uniform since very close direction vectors are removed.

Fig. 10(a) shows the direction vector set learned by LtA [the same as in Fig. 7(a)]. Fig. 10(b) shows the modified direction vector set where the modified direction vectors are highlighted in blue. We compute the Spacing indicator of the modified direction vector sets. The value is 0.0809. Compared with 0.1026 achieved by LtA as shown in Table II, we can see that the modified direction vector sets become more uniform. We perform Experiment 1 (i.e., CIR) to evaluate the performance of the modified direction vector sets (denoted as LtA-M). Fig. 11 shows the comparison results among different methods. We can observe that the performance of LtA-M is degraded in most cases compared with LtA. That is, although the uniformity of the direction vector sets is improved, their performance becomes worse in most cases. However, we can see that LtA-M outperforms the other methods in most cases. This shows that the slightly modified direction vector sets from the learned ones still have good performance.

These results support our conclusion: the learned direction vector sets by the LtA method are nonuniform and have better approximation quality than uniformly generated direction vector sets.

C. Runtime of LtA

We divide the runtime of LtA into two parts. Part I is the calculation of the hypervolume contribution for the training solution sets (line 1 in Algorithm 1). Part II is the loop to iteratively learn the direction vector set (lines 4–14 in Algorithm 1). We investigate the runtime of LtA by following the experimental settings in Section IV-A. For the hypervolume contribution calculation, we use the WFG algorithm [39]. Our hardware
platform is a virtual machine equipped with Intel Core i7-8700K CPU @ 3.70 GHz. Fig. 12 shows the runtime of a single run of LtA in each objective case.

Fig. 12(a) shows the runtime of Part I (i.e., hypervolume contribution calculation). We can see that the runtime increases exponentially as the number of objectives increases. This is due to the #P-hardness of exact hypervolume contribution calculation [17]. However, we do not need to perform Part I for each run if the same training solution sets are used among multiple runs. That is, we can save the hypervolume contribution information for further reuse.

Fig. 12(b) shows the runtime of Part II (i.e., the learning process of LtA). We can see that it needs a little bit long runtime (more than 4 h) to learn a direction vector set. As analyzed in Section III-D3, the time complexity of LtA is proportional to the training solution set size, the direction vector set size, and the maximum iteration number. We can control the runtime of LtA by modifying these parameters.

Although the learning process of LtA needs a little bit long runtime, the learned direction vector sets can be saved and are ready to use at any time. That is, once we obtain a good direction vector set by LtA, we can save it and use it directly in the future without spending a lot of time to regenerate it.

VI. CONCLUSION

The direction vector set plays an important role in the \( R_{2}^{HVC} \) indicator for hypervolume contribution approximation. We investigated how to generate a good direction vector set for the \( R_{2}^{HVC} \) indicator and proposed a learning method of direction vectors for improving the approximation quality of \( R_{2}^{HVC} \). The proposed method, LtA, is able to automatically learn a direction vector set based on given training solution sets. We verified the effectiveness of LtA based on three different experiments. All the results showed that LtA outperforms the other six direction vector set generation methods, which suggests the usefulness of LtA for auto direction vector set generation.

In the future, we will consider to further improve the LtA method. One direction is to improve the initialization of the direction vector set. Another direction is to improve the generation of a new direction vector in each iteration. We will also consider theoretical investigation about the reason behind the nonuniformity of the learned direction vector sets by LtA. A more fundamental theoretical challenge is to analyze the relation between the approximation quality of the \( R_{2}^{HVC} \) indicator and the number of direction vectors.

The proposed method can potentially improve the performance of the hypervolume-based EMO algorithms for many-objective problems where the exact hypervolume calculation is unrealistic. However, one issue in the hypervolume-based EMO algorithms, which has not been well recognized, is that the obtained solutions are often biased toward the lower order boundary of the Pareto front [40], especially in the case of many-objective optimization. That is, only a small number of inside solutions together with a large number of boundary solutions are often obtained. Moreover, such a bias is not easily visible in a high-dimensional objective space [41]. Further analysis of such a biased distribution of the obtained solutions for many-objective problems is also an interesting future research topic.

All the data sets and results involved in our experiments are available at https://github.com/HisaoLabSUSTC/LtA.

REFERENCES

[1] D. A. Van Veldhuizen, “Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations,” Ph.D. dissertation, Graduate School Eng., Air Force Inst. Technol., Dayton, OH, USA, 1999.
[2] C. A. C. Coello and M. R. Sierra, “A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm,” in Proc. Mex. Int. Conf. Artif. Intell., 2004, pp. 688–697.
[3] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. D. Fonseca, “Performance assessment of multiobjective optimizers: An analysis and review,” IEEE Trans. Evol. Comput., vol. 7, no. 2, pp. 117–132, Apr. 2003.
[4] K. Shang, H. Ishibuchi, L. He, and L. M. Pang, “A survey on the hypervolume indicator in evolutionary multiobjective optimization,” IEEE Trans. Evol. Comput., vol. 25, no. 1, pp. 1–20, Feb. 2021.
[5] M. P. Hansen and A. Jaskiewicz, “The Quality of Approximations to the Non-Dominated Set,” Dept. Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, 1998.
[6] M. Emmerich, N. Beume, and B. Naujoks, “An EMO algorithm using the hypervolume measure as selection criterion,” in Proc. Int. Conf. Evol. Multi-Criterion Optim., 2005, pp. 62–76.
[7] N. Beume, 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.
[8] 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.
[9] 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.
[10] K. Shang and H. Ishibuchi, “A new hypervolume-based evolutionary algorithm for many-objective optimization,” IEEE Trans. Evol. Comput., vol. 24, no. 5, pp. 839–852, Oct. 2020.
[11] M. T. M. Emmerich and C. M. Fonseca, “Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results,” in Proc. Int. Conf. Evol. Multi-Criterion Optim., 2011, pp. 121–135.
[12] A. P. Guerreiro and C. M. Fonseca, “Computing hypervolume contributions in up to four dimensions,” IEEE Trans. Evol. Comput., vol. 22, no. 3, pp. 449–463, Jun. 2018.
[13] L. While and L. Bradstreet, “A fast incremental hypervolume algorithm,” IEEE Trans. Evol. Comput., vol. 12, no. 6, pp. 714–723, Dec. 2008.
[14] L. While and L. Bradstreet, “Applying the WFG algorithm to calculate incremental hypervolumes,” in Proc. IEEE Congr. Evol. Comput., 2012, pp. 1–8.
[15] L. M. S. Russo and A. P. Francisco, “Extending quick hypervolume,” J. Heuristics, vol. 22, no. 3, pp. 245–271, 2016.
K. Shang, H. Ishibuchi, and W. Chen, “Greedy approximated hypervolume subset selection for many-objective optimization,” in IEEE Trans. Evol. Comput., vol. 23, no. 5, pp. 631–657, 2019.

A. Nan, K. Shang, and H. Ishibuchi, “What is a good direction vector for generating uniformly distributed points on a unit simplex for evolutionary many-objective optimization,” in Proc. IEEE Congr. Evol. Multi-Criterion Optim., 2019, pp. 179–190.

H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, “Reference point specification in inverted generational distance for triangular linear Pareto front,” IEEE Trans. Evol. Comput., vol. 16, no. 1, pp. 96–104, Feb. 2012.

H. Ishibuchi, T. Matsumoto, N. Masuyama, and Y. Nojima, “Optimal distributions of solutions for hypervolume maximization on triangular and inverted triangular Pareto fronts of four-objective problems,” in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), 2019, pp. 1857–1864.

Ke Shang (Senior Member, IEEE) received the B.S. and Ph.D. degrees from Xi’an Jiaotong University, Xi’an, China, in 2009 and 2016, respectively. He is currently a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His current research interests include evolutionary multiobjective optimization and artificial intelligence.

Tianye Shu received the B.S. degree from the Southern University of Science and Technology, Shenzhen, China, in 2021, where he is currently pursuing the master’s degree with the Department of Computer Science and Engineering. His current research interests include evolutionary multiobjective optimization and its applications.

Hisao Ishibuchi (Fellow, IEEE) received the B.S. and M.S. degrees from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree from Osaka Prefecture University, Sakai, Japan, in 1992. He is a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His research interest is evolutionary multiobjective optimization.

Dr. Ishibuchi is currently an IEEE CIS AdCom Member from 2021 to 2023, an IEEE CIS Distinguished Lecturer from 2021 to 2023, and the General Chair of IEEE WCCI 2024 in Yokohama, Japan.