Set-Based Discrete Particle Swarm Optimization Based on Decomposition for Permutation-Based Multiobjective Combinatorial Optimization Problems

Xue Yu, *Student Member, IEEE*, Wei-Neng Chen, *Member, IEEE*, Tianlong Gu, Huaxiang Zhang, Huaqiang Yuan, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—This paper studies a specific class of multiobjective combinatorial optimization problems (MOCOPs), namely the permutation-based MOCOPs. Many commonly seen MOCOPs, e.g., multiobjective traveling salesman problem (MOTSP), multiobjective project scheduling problem (MOPSP), belong to this problem class and they can be very different. However, as the permutation-based MOCOPs share the inherent similarity that the structure of their search space is usually in the shape of a permutation tree, this paper proposes a generic multiobjective set-based particle swarm optimization methodology based on decomposition, termed MS-PSO/D. In order to coordinate with the property of permutation-based MOCOPs, MS-PSO/D utilizes an element-based representation and a constructive approach. Through this, feasible solutions under constraints can be generated step by step following the permutation-tree-shaped structure. And problem-related heuristic information is introduced in the constructive approach for efficiency. In order to address the multiobjective optimization issues, the decomposition strategy is employed, in which the problem is converted into multiple single-objective subproblems according to a set of weight vectors. Besides, a flexible mechanism for diversity control is provided in MS-PSO/D. Extensive experiments have been conducted to study MS-PSO/D on two permutation-based MOCOPs, namely the MOTSP and the MOPSP. Experimental results validate that the proposed methodology is promising.**

***Index Terms*—Combinatorial optimization, decomposition, multiobjective optimization, permutation-based, particle swarm optimization (PSO), set-based.**

Manuscript received October 14, 2016; revised May 16, 2017; accepted July 8, 2017. Date of publication August 7, 2017; date of current version June 14, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, in part by the Natural Science Foundation of Guangdong under Grant 2015A030306024, and in part by the "Guangdong Special Support Program" under Grant 2014TQ01X550. This paper was recommended by Associate Editor S. Yang. *(Corresponding authors: Wei-Neng Chen; Jun Zhang.)*

X. Yu is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

W.-N. Chen and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

T. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

H. Zhang is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China.

H. Yuan is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The file contains additional figures and tables. The material is 767 kB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2728120

I. Introduction

COMBINATORIAL optimization [1] has been a prominent research topic in computer science and applications. There are a considerable number of combinatorial optimization problems (COPs) that are ubiquitous in reality but lack a polynomial-time solution. In practice, several conflicting objectives are usually expected to be optimized simultaneously, thus resulting in the development of multiobjective COPs (MOCOPs). Since many MOCOPs tend to be nondeterministic polynomial complete, even nondeterministic polynomial hard [2], it is often difficult to design exact algorithms to find all optimal solutions for an MOCOP. Instead, most of recent studies have focused on introducing stochastic methods to find a set of near-optimal solutions. Among these methods, multiobjective evolutionary algorithms (MOEAs) have attracted increasing attention [3]–[9] because of their effectiveness in solving continuous multiobjective optimization problems (MOPs) [10].

In this paper, we focus on a special class of MOCOPs, i.e., the permutation-based MOCOPs. Specifically, a permutation-based MOCOP is a problem with a finite set of elements in its problem model, and a solution to this problem always includes a permutation of these elements that satisfies certain constraints. This kind of problems is notable mainly due to two reasons. First, a number of widely researched MOCOPs are essentially permutation-based MOCOPs, e.g., the multiobjective traveling salesman problem (MOTSP) [11]–[14], the multiobjective project scheduling problem (MOPSP) [15]–[18], the multiobjective vehicle routing problem (MOVRP) [19]–[25], and the multiobjective flow-shop scheduling problem (MOFSSP) [26]–[30]. Second, although MOCOPs can be very different in constraints and objectives, permutation-based MOCOPs actually have the similar search space of permutation tree structure as shown in Fig. S1 in supplementary material available online at http://ieeexplore.ieee.org and the same requirement for permutation construction during the solution generation.

Therefore, although many algorithms have been developed specifically for each permutation-based MOCOP, it should be rational to develop a general method owing to their inherent similarities. To develop an MOEA for permutation-based MOCOPs, there are two main issues to be addressed: 1) *encoding and evolution operator* and 2) *multiobjective techniques*.

From the aspect of encoding and evolution operator, various evolutionary algorithms (EAs) with different encodings and evolution operators have been employed in MOEAs for different permutation-based MOCOPs. Particularly, genetic algorithms (GAs) and ant colony optimization (ACO) are quite popular. GAs, with problem-related encodings and different evolution operators, have been adopted in MOEAs for MOTSP [12], MOPSP [16], [18], MOVRP [19], [22], and MOFSSP [27]. ACO, featured with a constructive approach for solution generation with embedded heuristic information, has recently seen an increasing popularity in solving MOTSP [13], MOPSP [17], MOVRP [23], and MOFSSP [28]. Besides, multiobjective particle swarm optimizations (PSOs) have been proposed for problems such as MOVRP [25] and MOFSSP [29]. Furthermore, there are MOEAs based on estimated distribution algorithm (EDA) [11], memetic algorithm [21], differential evolution (DE) [30], and quantum EA [24].

From the aspect of multiobjective techniques, two types of techniques are often applied in MOEAs for permutation-based MOCOPs, i.e., the *Pareto-based* and the *decomposition-based*. The Pareto-based MOEAs such as NSGA-II [31], tend to solve a multiobjective problem as a whole mainly according to Pareto dominance and Pareto optimality. Decomposition-based MOEAs such as MOEA/D [32], usually decompose an MOP into a set of single-objective subproblems. And we found that compared with the Pareto-based MOEAs [14], [18], [19], [22], [23], fewer decomposition-based MOEAs have been developed in the field of permutation-based MOCOPs [13], [20].

Based on the above discussion, this paper intends to develop a general method termed multiobjective set-based PSO methodology based on decomposition (MS-PSO/D) for permutation-based MOCOPs, by combining concepts in both the set-based PSO (S-PSO) method [33] and MOEA/D [32].

First, S-PSO is a method for transforming PSO algorithms from continuous space to discrete space so as to solve COPs [33]. It is adopted to provide encoding and evolution operators in MS-PSO/D due to the following reasons.

1) PSO algorithms are pretty popular and ease-of-use [34]–[38]. However, since PSO is originally proposed for continuous problems, special methods are necessary to apply it to solve COPs, let alone MOCOPs. To integrate S-PSO into MOEA/D, we are actually to propose a method that can extend PSO algorithms into multiobjective discrete space so as to solve MOCOPs.
2) A step-by-step constructive approach can be achieved in S-PSO, which is quite intuitive to generate feasible solutions following the search space structured in a permutation tree, as shown in Fig. S2 in supplementary material. In addition, problem-based heuristic information can be easily brought into the approach to improve its efficiency.
3) Compared with other EAs having a constructive approach such as ACO, S-PSO has been proven promising for solving COPs such as traveling salesman problem (TSP) [33]. Thus, the proposed MS-PSO/D is also expected to be effective for solving MOCOPs.

Second, MOEA/D is chosen to tackle with multiobjective issues for reasons that: 1) decomposition-based multiobjective techniques have shown effectiveness when used in MOEAs to solve continuous MOPs [39]. However, in the field of permutation-based MOCOPs, they are much less researched compared with other techniques and 2) scalar heuristic information and scalar local search can be used in each subproblem in a natural way in MOEA/D [39], which can simplify the design of heuristic information in MS-PSO/D.

Furthermore, there are several noteworthy issues to tailor the proposed MS-PSO/D for solving permutation-based problems efficiently. First, the set-based representation in S-PSO should be further refined for permutation-based MOCOPs. Second, based on the specific set-based representation, a constructive approach is expected to generate feasible solutions exploring the search space in the shape of a permutation tree. Third, since different problems can have significantly different objectives, the introduction of problem-related heuristic information needs to be formalized. Besides, a flexible mechanism for diversity control is also required to preserve good performance under given computing resources.

Finally, the effectiveness and efficiency of MS-PSO/D will be studied on two typical permutation-based MOCOPs, i.e., MOTSP and MOPSP. Particularly, we consider a general extension of MOPSP, i.e., multiobjective multimode resource constrained project scheduling problem (MMRCPSP).

The rest of this paper is organized as follows. Section II introduces relevant background, including permutation-based MOCOPs, the set-based PSO and MOEA/D. In Section III, the proposed MS-PSO/D is presented in details. Sections IV and V discuss the behaviors of MS-PSO/D on MOTSP and MMRCPSP, respectively. Section VI draws a conclusion.

II. Background

A. Multiobjective Combinatorial Optimization

Under the assumption of minimizing all objectives, an MOCOP can be mathematically expressed as

$$\begin{aligned} &\text{minimize} \;\; f_1(X), f_2(X), \ldots, f_m(X) \\ &\text{subject to} \;\; X \in \Omega \end{aligned} \tag{1}$$

where Ω is the *discrete decision space*, X is a *feasible* solution in Ω, m is the number of objectives, and $f_i(X)$ is the

ith objective function. $F(X) = (f_1(X), f_2(X), \ldots, f_m(X))$ is the objective vector corresponding to solution X.

To deal with independent and conflicted objective functions, the *Pareto dominance* among solutions is defined [39].

Definition 1 (Pareto Dominance): Consider vectors $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{R^m}$. $\boldsymbol{u}$ is said to dominate $\boldsymbol{v}$ if and only if $u_j \leq v_j$ holds for every $j = 1, \ldots, m$, and $u_j < v_j$ is true for at least one j. A solution $X^* \in \Omega$ is said to be *Pareto optimal* if there is no other feasible solution $X \in \Omega$ such that $F(X)$ dominates $F(X^*)$. $F(X^*)$ is hence termed a *Pareto optimal objective vector*. The set of all the Pareto optimal solutions is called the Pareto set, and the corresponding set of Pareto optimal objective vectors is called the Pareto front (PF).

Based on this definition, an algorithm for MOPs is expected to find a set of solutions that can make a good approximation of the PF.

B. Permutation-Based MOCOPs

Many typical and widely researched MOCOPs are found to be permutation-based, despite differences in their objectives and constraints. Therefore, it is possible to develop a general algorithm for such a class of MOCOPs.

Definition 2 (Permutation-Based MOCOPs): Given n distinct elements with the universal element set $U = \{0, \ldots, n-1\}$, a permutation of the n elements is defined as

$$P = (x_0, \ldots, x_{n-1}), \text{ subject to}$$
$$\forall p, q \in \{0, \ldots, n-1\}, x_p, x_q \in U, x_p \neq x_q \text{ if } p \neq q. \quad (2)$$

The permutation space $\mathbf{P}$ consists of all possible permutations of the n elements, i.e., $|\mathbf{P}| = n!$. A COP or MOCOP is said to be permutation-based if any candidate solution to the problem must include a permutation of some certain elements.

To further illustrate this concept, MOTSP and MMRCPSP are introduced as examples in the following.

1) MOTSP: Traveling salesman problem is a typical permutation-based COP that aims to find a Hamiltonian circuit with minimum cost in a graph. More specifically, given a set of cities (elements) $U = \{0, \ldots, n-1\}$ in a graph, the goal of a TSP is to find a permutation P of the n cities in the form of (2), such that the salesman visits each city once and only once following the permutation and returns to the initial city, and finally the travel cost is minimized.

The cost function is usually defined based on the weights of arcs in the graph. In the single-objective TSP, each pair of arcs that connects two nodes j and k is associated with costs $c_{j,k}$ and $c_{k,j}$. Then, the cost of a city permutation P is $f(P) = \sum_{d=0}^{n-1} c_{x_d, x_{(d+1)\%n}}$. Accordingly, an MOTSP with m objectives can be defined as

$$\text{minimize } f_i(P) = \sum_{d=0}^{n-1} c^i_{x_d, x_{(d+1)\%n}}, i = 1, \ldots, m \quad (3)$$

where $f_i(P)$ is the ith objective under the ith cost matrix, i.e., $C(i) = (c^i_{j,k})_{n \times n}$. $c^i_{j,k}$ and $c^i_{k,j}$ are the ith kind of costs associated with the arcs that connect nodes j and k.

2) MMRCPSP: The traditional resource constrained project scheduling problem (RCPSP) is a problem to schedule the tasks of a project so as to minimize project makespan considering task dependency and resource constraints. To solve RCPSP, a common approach is to schedule the tasks based on a task permutation [40]. That is, given a project with a set of tasks (elements), an algorithm first finds a task permutation in the form of (2). Then, a serial schedule generation scheme (SSGS) can be applied to generate a project schedule [40]. The SSGS works by repeatedly picking the first unassigned task from the task permutation, putting it to the earliest possible start time that satisfies the constraints.

MRCPSP is a general extension of RCPSP, where each task can be implemented in different alternative modes and a mode usually has a shorter duration if it consumes more resources. Also, researchers have recently considered that other objectives in addition to project makespan, such as minimization of task delay, should also be included in project scheduling problems [40], [41]. Therefore, this paper considers the MMRCPSP with three objectives following [41]: project makespan, waiting time, and resource consumption, defined as:

$$\text{minimize Makespan} = \max\{\text{FT}(j) | j \in U\}$$
$$\text{WT} = \sum_{j=0}^{n-1} \text{ST}(j)$$
$$\text{RC} = \sum_{j=0}^{n-1} \sum_{k=1}^{R} cr^k \times rr^k_{j,mj}. \quad (4)$$

Project Makespan is defined by the largest finish time FT(j) of all tasks j ($j \in U$), i.e., the time when the execution of all tasks has finished. Waiting time WT is defined by the summation of the start time ST(j) of all tasks j. Denote R as the number of renewable resources considered, cr^k as the cost per unit of the kth resource ($k = 1, 2, \ldots, R$), mj as the selected mode for the jth task, and $rr^k_{j,mj}$ as the number of the kth resource required by the mjth mode of the jth task. Then, the resource consumption RC is given by the sum of the resource costs of each task.

As described above, MOTSP and MMRCPSP are both permutation-based MOCOPs. However, these two problems are actually with quite different objective functions and constraints. Moreover, they have different characteristics in permutation positioning and decision space. First, relative permutation positioning is concerned in MOTSP since permutations $(x_0, \ldots, x_{n-1})$ and $(x_1, \ldots, x_{n-1}, x_0)$ are indeed identical. In contrast, absolute positioning is often concerned in MMRCPSP, since a task appears earlier in the task permutation can have a higher priority to consume resources. Second, the decision space of MOTSP is simply the permutation space since a permutation is already a solution. But MMRCPSP has a more complex decision space and extra decisions for mode selection are required. Since MOTSP and MMRCPSP are two representative permutation-based MOCOPs, we validate the proposed method on these two problems in this paper.

C. Population-Based EAs

To deal with complex or difficult problems efficiently, many EAs have been proposed to obtain optimal or suboptimal solution(s) in a reasonable time. In general, an EA evolves a population iteratively to find the optima of problems to be settled, where each individual in the population usually represents a candidate solution.

The development of EAs dates back to 1950s [42] with the emergence of algorithms such as evolutionary programming and evolution strategy. Since then, more and more EAs have been proposed and applied to solve theoretic and realistic problems, such as GAs [43], [44], DEs [45]–[47], PSO algorithms [48]–[52], ACO algorithms [53]–[56], and hybrid or memetic algorithms [57]–[58].

Most of the above mentioned EAs are originally developed for single-objective and continuous problems. However, in reality, multiobjective and discrete problems are quite common. To fill the gap, multiobjective techniques such as MOEA/D [32] and discretization methods such as S-PSO [33], have been proposed to extend the original EAs to their multiobjective and discrete versions, respectively.

D. Set-Based Particle Swarm Optimization

S-PSO is a method proposed in [33], in which continuous PSO algorithms can be implemented so as to solve COPs. S-PSO is characterized by its set-based representation and set-based way to update velocities and positions.

1) Representation: Many COPs can be formulated as "find from a set U a subset X that satisfies some constraints and optimizes the objective function" [33]. Thus in S-PSO, particle i's position X_i is represented by a set of elements and its velocity V_i is represented by a set with possibilities to indicate the possibility of selecting any element into its position. The concrete representation is problem-dependent. For instance, particles are represented based on arcs between cities for TSP, but based on items and 0-1 values for the 0-1 knapsack problem (0-1 KP) [33]. However, by concentrating exclusively on permutation-based problems, it is possible to define a general set-based representation.

2) Velocity Update: In S-PSO, velocities are updated according to the velocity update formula of a specific continuous PSO. However, mathematic operations in the original formula need to be redefined on sets. For example, supposing the global PSO (GPSO) is implemented in S-PSO, i.e., S-GPSO, the velocity update formula is

$$V_i = \omega V_i + c_1 r_1 (p\text{Best}_i - X_i) + c_2 r_2 (g\text{Best}_i - X_i) \quad (5)$$

where ω, $c_1 r_1$, and $c_2 r_2$ are real coefficients, pBest and gBest are best-so-far positions found by a single particle and the whole swarm respectively. Based on the above set-based representation, X_i, $p\text{Best}_i$, $g\text{Best}_i$ are crisp sets, V_i is a set with possibilities, and $(p\text{Best}_i - X_i)$ and $(g\text{Best}_i - X_i)$ are also crisp sets by set subtraction. To obtain the new V_i, operations such as multiplication between a coefficient and a set with possibilities, multiplication between a coefficient and a crisp set, and addition of sets of possibilities, are defined in [33].

3) Position Update (Solution Construction): The elements in the best-so-far positions are more promising and they tend to be assigned larger possibility values in velocity update. Thus, in position update, a particle preferentially learns from its velocity by selecting elements whose possibilities values excess some certain threshold, such that a more potential position can be constructed. Moreover, heuristic information can be embedded to guide the element selection. The concrete process of position update is also problem-dependent. For example, in [33], the position is viewed as a whole and updated in a single step when solving 0-1 KP, but for solving TSP, a step-by-step constructive approach is applied and only available elements are considered in each step. For permutation-based problems such as TSP and MOTSP, the constructive approach with embedded heuristic information is quite intuitive, suitable, and practicable.

E. Decomposition and MOEA/D

To our best knowledge, fewer decomposition-based MOEAs than Pareto-based MOEAs have been proposed for solving permutation-based MOCOPs. However, decomposition-based MOEAs should be quite potential for solving these problems as scalar heuristic information can be easily used in them.

Decomposition strategy has been widely adopted to develop MOEAs [59]–[63]. Generally, it is a fitness assignment scheme to convert a vector into a scalar value, and the value can be used for selection, diversity maintenance, and mating restriction. A decomposition-based MOEA decomposes an MOP into a series of single-objective subproblems using a set of weight vectors according to some certain aggregation approaches, among which the weighted-sum approach and the Tchebycheff approach are simple and popular [64]. Other approaches such as normal boundary intersection [65], have also been adopted in [66] and [67].

MOEA/D is a popular decomposition-based multiobjective evolutionary framework [32], characterized by *subproblems* and *neighborhood*. It assigns each individual in the population a specific weight vector, which corresponds to a subproblem. A good approximation of the PF might depend on a good coverage of the weight vectors. In addition to the widely used evenly distributed weight vectors, some more sophisticated weight vectors are also proposed [59], [68]. Furthermore, the neighborhood $B(i)$ of subproblem i consists of the closest T subproblems in terms of the distances between weight vectors. All subproblems in a neighborhood cooperate to evolve.

III. Set-Based Discrete Particle Swarm Optimization Based on Decomposition

In order to develop a general method for permutation-based MOCOPs, a set-based discrete particle swarm optimization based on decomposition is developed in this section. It is essentially formed by the concepts of S-PSO and MOEA/D, and is hence denoted as MS-PSO/D. This is rather a method than an algorithm, in which PSOs with different learning strategies can be implemented in order to solve different permutation-based MOCOPs.

Fig. 1 illustrates the general framework of MS-PSO/D. The "problem analysis" is necessary to identify the problem characteristics, such as constraints and basic elements. Learners, i.e., best-so-far positions, are updated in MOEA/D way and archive is maintained according to Pareto dominance. In the

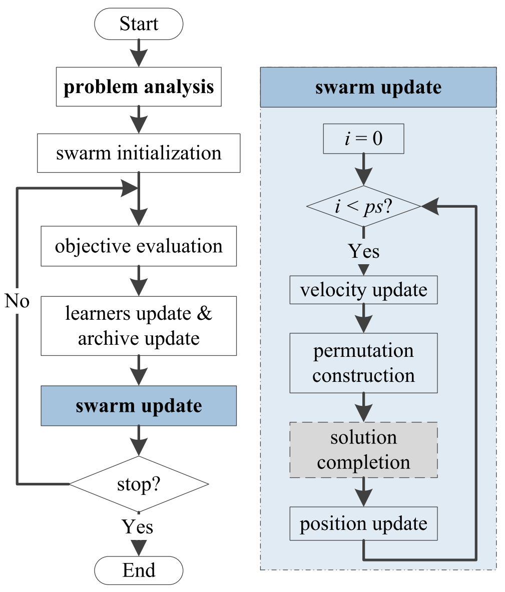

Fig. 1. Overall framework of MS-PSO/D.

main "swarm update," velocity update, solution construction, and position update are performed successively by each particle. Velocity update and position update depend on the update formulas and the representation of particles. In solution construction, permutation construction is always essential and necessary, and a solution completion process is performed only when extra decisions are required.

In the following, we will demonstrate MS-PSO/D from four respects: 1) representation scheme for particles and the related operations; 2) solution construction scheme; 3) heuristic information; and 4) MOEA/D issues.

A. Element-Based Representation Scheme and Operations

Although the concrete representation in S-PSO usually needs to be specifically defined for different COPs, MS-PSO/D manages to develop a general set-based representation for permutation encoding and permutation construction of all permutation-based MOCOPs. For permutation encoding, a permutation can be obtained from the adjacent relationship of elements within it. For permutation construction, the main concern is indeed to find the potential adjacent elements to a certain element. Therefore, contrary to the representation in S-PSO which views all elements as a whole, MS-PSO/D develops an element-based representation by defining a unique set-based component for each element in position or velocity.

Given a problem with element set $U = \{0, \ldots, n-1\}$, to represent permutation $P = (x_0, \ldots, x_{n-1})$, the position of particle i is defined as an array of n crisp sets, i.e., $X_i = (X_i^0, \ldots, X_i^{n-1})$. Each component $X_i^d (d \in U)$ is defined to include adjacent elements to element d as

$$X_i^d = \left\{x_{(p-1+n)\%n}, x_{(p+1)\%n}\right\}, d = x_p. \tag{6}$$

The permutation is thought to be circular and each element hence has two adjacent elements. Element $x_{(p-1+n)\%n}$ is prior to x_p in permutation P, and it will be x_{n-1} for x_0. Element $x_{(p+1)\%n}$ is next to x_p, and it will be x_0 for x_{n-1}.

Accordingly, the velocity of a particle is defined to indicate the possibility that any element is adjacent to a certain element in a permutation. That is, the velocity of particle i is represented as $V_i = (V_i^0, \ldots, V_i^{n-1})$. Each component $V_i^d (d \in U)$ is related to element d and it is a set with possibilities defined as

$$V_i^d = \left\{(e, p_i^d(e)) | e \in U, p_i^d(e) \in [0, 1]\right\}. \tag{7}$$

Each integer-real pair $(e, p_i^d(e))$ means that element e is adjacent to element d in a permutation with possibility $p_i^d(e)$. Since the velocity is used to guide the step-by-step permutation construction, $p_i^d(e)$ actually reflects the possibility of selecting element e next to element d when constructing a permutation of elements. Thus, $p_i^d(e)$ and $p_i^e(d)$ are recommended to always keep equal in some problems, e.g., in MOTSP, where relative permutation positioning is concerned.

The above representation is merely defined for permutation encoding and construction. As mentioned in Section II, in some problems such as MMRCPSP, the decision space is more complex and a solution should include not only a permutation of elements but also some problem-related extra decisions. For these problems, the particles must be further represented for the extra decisions. In the following, taking MMRCPSP for an example, we demonstrate that an independent element-based representation can also be realized for extra decisions.

Given an MMRCPSP where each task d has md modes with mode set $M^d = \{0, \ldots, md-1\}$, the position and velocity of particle i are further defined as $S_i = \{X_i, \bar{X}_i\}$ and $W_i = \{V_i, \bar{V}_i\}$, respectively. For extra decisions required by multimode case, the extra position $\bar{X}_i = (\bar{X}_i^0, \ldots, \bar{X}_i^{n-1})$ is defined to represent results of mode selection, where each component $\bar{X}_i^d$ is a crisp set filled by the index of the mode selected for task d; the extra velocity $\bar{V}_i = (\bar{V}_i^0, \ldots, \bar{V}_i^{n-1})$ is an array of sets with possibilities to guide mode selection, with each component defined as

$$\bar{V}_i^d = \left\{(m, \bar{p}_i^d(m)) | m \in M^d, \bar{p}_i^d(m) \in [0, 1]\right\}. \tag{8}$$

Each pair $(m, \bar{p}_i^d(m))$ indicates the possibility of selecting mode m for task d. Thus, $S_i^d = \{X_i^d, \bar{X}_i^d\}$ and $W_i^d = \{V_i^d, \bar{V}_i^d\}$ form the components for task (element) d in particle i's position and velocity, respectively.

Based on the aforementioned representation, element-based arithmetic operations are defined such that velocity can be updated component by component. Recall that MS-PSO/D is a methodology and the velocity update is related to the specific PSO implemented within it. Different PSOs have different learning strategies and velocity update formulas. Fortunately, the various formulas share similar arithmetic operations. Specifically, the multiplication between a coefficient and a velocity component (e.g., $c \times V_i^d$) is defined as (9) to change the possibility values using the coefficient; the addition between velocity components (e.g.,$V_i^d + V_j^d$) is defined as (10) to merge pairs within them and obtain a set with possibilities; the subtraction between two position components performs a normal set subtraction as (11); and the multiplication between a coefficient and a position component is defined to convert a set into a set with possibilities using the coefficient as (12). Besides, to convert a velocity component into

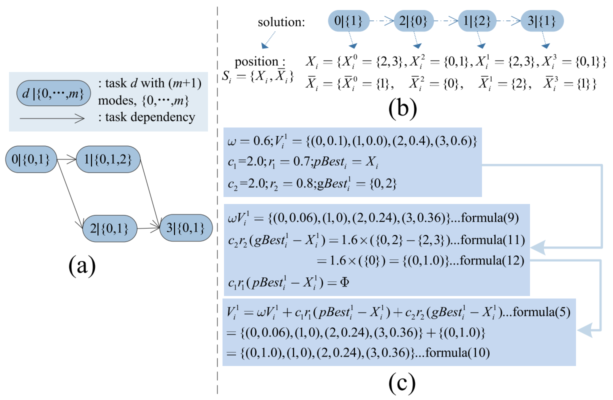
$d\,|\{0,\cdots,m\}$: task d with (m+1) modes, $\{0,\cdots,m\}$
: task dependency
0|{0,1}
1|{0,1,2}
2|{0,1}
3|{0,1}
(a)
solution:
0|{1}
2|{0}
1|{2}
3|{1}
position : $S_i = \{X_i, \bar{X}_i\}$
$X_i = \{X_i^0 = \{2,3\}, X_i^2 = \{0,1\}, X_i^1 = \{2,3\}, X_i^3 = \{0,1\}\}$
$\bar{X}_i = \{\bar{X}_i^0 = \{1\}, \bar{X}_i^2 = \{0\}, \bar{X}_i^1 = \{2\}, \bar{X}_i^3 = \{1\}\}$
(b)
$\omega = 0.6; V_i^1 = \{(0,0.1),(1,0.0),(2,0.4),(3,0.6)\}$
$c_1 = 2.0; r_1 = 0.7; pBest_i = X_i$
$c_2 = 2.0; r_2 = 0.8; gBest_i^1 = \{0,2\}$
$\omega V_i^1 = \{(0,0.06),(1,0),(2,0.24),(3,0.36)\}$...formula(9)
$c_2 r_2 (gBest_i^1 - X_i^1) = 1.6\times(\{0,2\}-\{2,3\})$...formula(11)
$= 1.6\times(\{0\}) = \{(0,1.0)\}$...formula(12)
$c_1 r_1 (pBest_i^1 - X_i^1) = \Phi$
$V_i^1 = \omega V_i^1 + c_1 r_1 (pBest_i^1 - X_i^1) + c_2 r_2 (gBest_i^1 - X_i^1)$...formula(5)
$= \{(0,0.06),(1,0),(2,0.24),(3,0.36)\} + \{(0,1.0)\}$
$= \{(0,1.0),(1,0),(2,0.24),(3,0.36)\}$...formula(10)
(c)

Fig. 2. Sample representation and operations for MMRCPSP. (a) MMRCPSP instance. (b) Example of solution representation. (c) Example of velocity update.

Algorithm 1 Permutation Construction

Input: position and velocity of particle i (X_i, V_i)
Output: a permutation $P = (x_0, \ldots, x_{n-1})$

```
1.  $UE = \{0, 1, \ldots, n-1\}$ //set of unchosen elements
2.  for $p = 0, 1, \ldots, n-1$
3.      $E = CheckConstraints(UE)$; // available elements
4.      clear set $V$ and set $X$; // $V$ and $X$ should be empty
5.      if $p > 0$ //obtain available elements in velocity and position
6.          $V = \text{cut}_\alpha(V_i^{x_{p-1}}) \cap E$; $X = X_i^{x_{p-1}} \cap E$;
7.      end if
8.      if $V$ is not empty //learn from velocity in the highest priority
9.          $x_p = ChooseOneFrom(V)$; //choose one element from set $V$
10.     else if $rand(0, 1) < plearnX$ and $X$ is not empty
11.         $x_p = ChooseOneFrom(X)$;
12.     else
13.         $x_p = ChooseOneFrom(E)$;
14.     end if
15.     delete $x_p$ from set $UE$; //delete the chosen element from $UE$
16. end for
```

a crisp set of elements, operation $\alpha-$cut is defined as (13) according to a random real number α generated in [0, 1]

$$c \times V_i^d = \left\{\left(e, p'(e)\right) \middle| \left(e, p_i^d(e)\right) \in V_i^d\right\}$$

$$p'(e) = \min\left\{c \times p_i^d(e), 1\right\} \tag{9}$$

$$V_i^d + V_j^d = \left\{\left(e, p'(e)\right) \middle| \left(e, p_i^d(e)\right) \in V_i^d \text{ or } \left(e, p_j^d(e)\right) \in V_j^d\right\}$$

$$p'(e) = \begin{cases} \max\left(p_i^d(e), p_j^d(e)\right), \left(e, p_i^d(e)\right) \in V_i^d, \left(e, p_j^d(e)\right) \in V_j^d \\ p_i^d(e), \left(e, p_i^d(e)\right) \in V_i^d \\ p_j^d(e), \left(e, p_j^d(e)\right) \in V_j^d \end{cases} \tag{10}$$

$$X_i^d - X_j^d = X_{i,j}^d = \left\{e \middle| e \in X_i^d, e \notin X_j^d\right\} \tag{11}$$

$$c \times X_{i,j}^d = \left\{\left(e, p'(e)\right) \middle| x \in X_{i,j}^d\right\}, p'(e) = \min\{c, 1\} \tag{12}$$

$$\text{cut}_\alpha\left(V_i^d\right) = \left\{e \middle| \left(e, p_i^d(e)\right) \in V_i^d, p_i^d(e) > \alpha\right\}. \tag{13}$$

In Fig. 2, we exemplify the element-based representation and operations in an MMRCPSP instance. Fig. 2(a) illustrates the instance of four tasks, where task 0 must be finished before task 1 and task 2 start, task 1 and task 2 must be finished before task 3 starts. Except task 1 has three modes, each task has two modes. Fig. 2(b) shows a possible solution, where tasks are scheduled in the permutation $P = (0, 2, 1, 3)$ and mode 1, mode 0, mode 2, and mode 1 are chosen for task 0, task 2, task 1, and task 3, respectively. And a set-based position S can be generated to represent the solution, where $X_i^1 = \{2, 3\}$ and $\bar{X}_i^1 = \{1\}$ since tasks 2 and 3 are adjacent to task 1 in P, and mode 1 is chosen for task 1. And Fig. 2(c) exemplifies how a velocity component can be updated according to velocity update formula (5).

B. Solution Construction Scheme

For various permutation-based MOCOPs, a solution must include a permutation of some certain elements and the goal of solution construction in MS-PSO/D is to construct a feasible and potential permutation-based solution.

1) The feasibility is promised by the step-by-step constructive approach such that constraints can be easily satisfied in each step. Owing to the element-based representation, a constructive approach can be well-defined for solution construction.
2) The potential is promised by providing a flexible balance of convergence and diversity, which is achieved by introducing a few parameters for diversity control in MS-PSO/D.

For the same requirement for permutation construction of various permutation-based MOCOPs, a general constructive approach is formalized, where elements are chosen one by one and each element is chosen by learning from the velocity component and position component corresponding to the last chosen element. Pseudocode of the general permutation construction is detailed in Algorithm 1, where only function "*CheckConstraints*" needs modifications when dealing with different problems. Set UE records all unchosen elements. Permutation is constructed in a loop with n iterations and an element is chosen to be x_p in the pth iteration (lines 2–16). In the loop, all unchosen elements are checked first to obtain a set E of available elements that satisfy the constraints (line 3). If $p > 0$, sets V and X are formed by available elements from the x_{p-1}th components of velocity and position of particle i respectively (lines 5–7). Then, the particle chooses an element as x_p from sets V, X and E with decreasing priority (lines 8–14). Elements in V, i.e., velocity, are chosen in the highest priority because that the velocity of a particle is mainly maintained according to good experiences of particles in the swarm. If there is no available element in V, the particle tends to check elements in X with probability *plearnX* (line 11) so as to exploit the decision space near its old position. If nothing found in X, the particle treats all available elements in E impartially to explore new decision space (line 13).

In the above process, element selection is performed in S-PSO way. But differing from the original S-PSO, MS-PSO/D utilizes only the components rather than the whole of velocity and position owing to the element-based representation, which can accelerate the computation of sets V and X in each iteration. Moreover, in S-PSO, different levels of diversity can only be achieved by implementing different PSOs. To overcome the lack of mechanism for diversity control, parameters *plearnX* and *p*Max are introduced in element selection.

Algorithm 2 Solution Completion Process for MMRCPSP

Input: task permutation P, position and velocity of particle i
Output: array $MT[]$ to record results of mode selection

```
1.  for $p = 0, 1, \ldots, n-1$ //mode selection following $P$
2.    $tid = x_p$; $M = ModeNum(tid)$; //task $tid$ has $M$ modes
3.    $E = \{0, 1, \ldots, M-1\}$; //set of modes of task $tid$
4.    $E = ResourceConstraint(E)$; //set of available modes
5.      $V = \mathrm{cut}_\alpha(\bar{V}_i^{tid}) \cap E$; $X = \bar{X}_i^{tid} \cap E$;
6.    if $V$ is not empty
7.      $mid = ChooseOneFrom(V)$;
8.    else if $rand(0, 1) < plearnX$ and $X$ is not empty
9.      $mid = ChooseOneFrom(X)$;
10.   else
11.     $mid = ChooseOneFrom(E)$;
12.   end if
13.   $MT[tid] = mid$; // mode $mid$ is selected for task $tid$
14. end for
```

plearnX is used to limit the probability of learning from X (line 8). A smaller *plearnX* means that the particle learns less from its old position and explores new space more by learning from E, which should result in more diversity. In "*ChooseOneFrom*," instead of always choosing the element with the best heuristic information, the best element can only be chosen in a certain probability, i.e., pMax. Otherwise, a roulette selection is applied. Thus, a larger pMax means a lower probability to apply roulette selection and hence less diversity.

After permutation construction, a solution completion process needs to be performed only when extra decisions are required. The process is problem-dependent since different extra decisions can be required by different problems. However, based on the element-based representation defined for the extra decisions, it is also possible to formalize a constructive approach for the solution completion process akin to the permutation construction. As a result, even for a problem with a requirement for extra decisions, a constructive approach can still be realizable so as to generate only feasible solutions.

Taking MMRCPSP for an example, we demonstrate the completion process can also be formalized in a constructive approach. Suppose a task permutation $P = (x_0, \ldots, x_{n-1})$ has been constructed by permutation construction. Solution completion process is formalized as Algorithm 2, where mode selection is performed for each task following P according to the velocity and position components defined for multimode case. The mode selection is quite similar to the element selection in permutation construction. In Algorithm 2, set E contains all available modes of task *tid* that satisfy resource constraint (line 4). And sets V and X are formed by available elements in velocity and position respectively (line 5). Similarly, a mode is selected for task *tid* from sets V, X, and E with decreasing priority (lines 6–12). Then, the selected mode *mid* is recorded (line 13). Finally, the results of mode selection and task permutation constitute a complete solution and in "position update," it will be converted into an element-based position.

C. Heuristic Information

For the sake of efficiency and convergence rate, heuristic information is included in solution construction of MS-PSO/D and it is designed to reflect the probability of choosing a certain element out of several available elements. In particular, for permutation construction, given a problem with n elements, an $n \cdot n$ heuristic matrix Heu_i should be provided for each particle i, where $\mathrm{Heu}_i[j][k]$ represents the probability of particle i to choose element k after element j.

Generally, efficient *offline* heuristic information that needs to be computed only once at the beginning, is preferred in MS-PSO/D. But *online* heuristic information that must be updated in real time, is also practicable owing to the constructive solution construction. Furthermore, in order to combine information of multiple objectives, strategies such as decomposition and normalization are often necessary.

Here, we list the settings of heuristic information for MOTSP and MMRCPSP in our experiments. For MOTSP, the heuristic matrix is set according to $\mathrm{Heu}_i[j][k] = g(c_{j,k}|\lambda_i)$, where λ_i is the weight vector of particle i, g is the decomposition function, and $c_{j,k}$ is the cost vector corresponding to arc $<j, k>$. Normalization is necessary if values in the cost vector are of different orders of magnitude. For MMRCPSP, as for heuristic information used in permutation construction, the latest finish time (LFT) [69] of each task k is computed and used by the entire swarm, i.e., $\mathrm{Heu}_i[j][k] = \mathrm{LFT}(k)$. As for mode selection, real-time heuristic information is adopted for efficiency. That is, a heuristic vector $\mathrm{heu}_{i,j}$ is computed during mode selection of task j for particle i according to $\mathrm{heu}_{i,j}(k) = g(c_{j,k}|\lambda_i)$, where $\mathrm{heu}_{i,j}(k)$ represents the probability for particle i to choose mode k for task j. The cost vector $c_{j,k} = (ft_{j,k}, st_{j,k}, rc_{j,k})$ corresponds to objectives Makespan, WT, and RC as defined in (4). $rc_{j,k}$ is the resource cost of the kth mode of task j. $st_{j,k}$ and $ft_{j,k}$ are the earliest start time and finish time of task j if executed in its kth mode, and they need to be computed in real time.

D. MOEA/D Issues

To process multiobjective issues, in addition to the crucial settings for MOEA/D, learner(s) update and fitness evaluation in S-PSO must also be adjusted in the context of MOEA/D.

For settings of MOEA/D, weight vectors and the partition of neighborhood are important. Weight vectors are used to convert an MOP into a set of single-objective subproblems. A set of static weight vectors is often employed, although dynamic and adaptive weight vectors can also be found [68]. The evenly distributed weight vectors are implemented for simplicity in this paper. Besides, neighborhood $B(i)$ of particle i is defined as in [32] based on the distances between weight vectors, which consists of the T closest particles to particle i.

Learners update that is used to take place in the whole swarm in S-PSO, is restricted in neighborhood in MS-PSO/D. Suppose GPSO is implemented in MS-PSO/D, i.e., MS-GPSO/D. After every particle has finished constructing its new solution, the global learner of particle i, i.e., best-so-far solution $g\mathrm{Best}_i$, is updated according to all newly constructed solutions in its neighborhood $B(i)$. Besides, to control the dominion of a very good solution, a parameter *MaxUsedT* is introduced to limit the times of the new solution to be gBest. A new solution is flagged as *used* after it has been used for

MaxUsedT times. Thus, based on fitness values of all unused newly constructed solutions in $B(i)$, the best one is chosen to update $g\text{Best}_i$.

As for fitness evaluation, decomposition strategy is used to transform an objective vector into a scalar fitness. However, decomposition in MOEA/D is performed on the assumption that the values of different objectives are of a similar order of magnitude, which may not hold in many permutation-based MOCOPs. Thus, normalization is necessary in MS-PSO/D and decomposition-based fitness evaluation is performed only after normalization has been completed in each objective dimension. Generally, a uniform normalization based on the upper bound (max) and the lower bound (min) suffices. That is, in each objective dimension o, two values, i.e., $\max_o$ and $\min_o$, are maintained to represent the maximum and minimum objective values of solutions found so far. Once new solutions have been generated in each iteration, each objective value x_o of these solutions in dimension o should be used to update $\max_o$ or $\min_o$, if it is larger than $\max_o$ or smaller than $\min_o$, respectively. After that, an objective value x_o can be normalized into a value in [0, 1] using $(x_o - \min_o)/(\max_o - \min_o)$, and it will be used in fitness evaluation.

E. Overall MS-PSO/D

All in all, the proposed MS-PSO/D mainly makes use of the set-based representation and solution construction of S-PSO, and the decomposed neighborhood of MOEA/D. Specifically, element-based representation is defined for permutation-based MOCOPs, which facilitates the formalization of a step-by-step constructive approach for solution construction. Owing to the constructive approach, not only offline but also online heuristic information can be easily brought into MS-PSO/D. On the other hand, decomposition approach is employed to obtain scalar fitness and heuristic information. It is worth noting that normalization can be necessary before scalarization. Besides, neighborhood in MS-PSO/D plays a role in restricting the scope to update learners of particles. Furthermore, parameters are introduced in the process of solution construction and learners update to control diversity of MS-PSO/D.

In particular, to solve a permutation-based MOCOP of n elements and m objectives, the overall execution process of MS-PSO/D can be detailed as follows.

Step 1 (Initialization): Initialize parameters. Generate even weight vectors and assign each weight vector to a particle. Extract heuristic information of each objective and set heuristic matrix using the weight vectors. For each particle, initialize its velocity to include all feasible elements with the possibilities set as 1.0, and then construct a solution for its position initialization.

Step 2 (Evaluation and Normalization): Evaluate all particles to obtain objective vectors. Update the upper bound and lower bound of each objective. Accordingly, normalize the objective vector of each particle.

Step 3 (Learner(s) and Archive Update): Update learners for each particle in MOEA/D way using the particle's weight vector and all newly generated solutions in its neighborhood. Use all new solutions to update archive based on Pareto dominance.

Step 4 (Swarm Update): For each particle, update its velocity by performing element-based arithmetic operations in PSO's velocity update formula, and then construct a new solution to update its position.

Step 5 (Termination): If the stopping condition is met, stop and output archive. Otherwise, go to step 2.

IV. Behaviors on MOTSP

Fifteen bi-objective and 6 tri-objective instances are constructed from TSPLIB [70] for experiments on MOTSP, and the number of cities varies from 100 to 500. As shown in Table SI in supplementary material available online at http://ieeexplore.ieee.org, m is the number of objectives, n is the number of cities, the column "units" lists the single-objective TSP instances used to construct the corresponding MOTSP instances.

A. Performance Metrics

We use the inverted generational distance (IGD) [71] and the Hypervolume (HV) [72] as performance metrics in this paper.

The IGD [71] has been widely used as a performance metric for multiobjective optimization. Let P be a PF approximated by an algorithm, P^* be a set of uniformly distributed points in the actual PF of a multiobjective problem. The IGD from P^* to P is defined as

$$\text{IGD}(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|}.$$

$d(v, P)$ is the minimum Euclidean distance from point v to all points in P. The lower the IGD is, the better is the approximation. In this paper, for lack of a prior PF, P^* is always defined as the set of all nondominated points in PFs approximated by the algorithms to be compared and assessed.

The HV [72] measures the size of the space dominated by a certain approximated PF. Let $\boldsymbol{y}* = (y_1*, \ldots, y_m*)$ be a point dominated by any point in the approximated PF P. Then, the HV value of P with regard to the *reference point* $\boldsymbol{y}*$ is the volume of the region which is dominated by P but dominates $\boldsymbol{y}*$. A larger HV value implies a larger dominated space of the approximated PF as well as a better approximation. In this paper, we adopt a fast way namely WFG [73] to calculate exact HV. The worst objective values found by all compared algorithms are used to maintain the reference point.

B. Comparative Analysis

MOEA/D-ACO has been proposed recently by combining ACO and MOEA/D, and its efficiency for MOCOPs has been proven in experiments [13]. Thus, to some extent, comparison with MOEA/D-ACO can illustrate the efficiency of MS-PSO/D. The source code of MOEA/D-ACO is available online [74].

1) Basic Settings: Unless specified otherwise, 24*3000 solutions are generated in total by each run of an algorithm and all experimental results are based on 30 independent runs as in [13].

TABLE I
IGD AND HV STATISTICS OF THE FINAL APPROXIMATIONS OBTAINED BY MOEA/D-ACO^T AND MS-GPSO/D^W

INSTANCES		IGD					HV				
obj#	name	MOEA/D-ACO^T		MS-GPSO/D^W			MOEA/D-ACO^T		MS-GPSO/D^W		
		mean	std dev	mean	std dev	W-test	mean	std dev	mean	std dev	W-test
2	kroAB100	2385.43	284.83	**1271.81**	***170.98***	+	2.18E+10	2.48E+08	**2.20E+10**	***1.88E+08***	+
	kroAC100	2409.51	229.25	**1226.05**	***144.69***	+	2.17E+10	2.30E+08	**2.19E+10**	***1.84E+08***	+
	kroAD100	1966.48	235.62	**1115.81**	***131.01***	+	1.66E+10	1.82E+08	**1.67E+10**	***1.35E+08***	+
	kroAE100	2316.04	194.63	**1214.58**	***172.26***	+	2.07E+10	2.06E+08	**2.09E+10**	***1.79E+08***	+
	kroBC100	1986.12	203.87	**1161.51**	***162.99***	+	1.83E+10	2.34E+08	**1.85E+10**	***1.72E+08***	+
	kroBD100	2039.55	144.17	**1291.56**	***135.28***	+	1.74E+10	2.15E+08	**1.75E+10**	***1.18E+08***	+
	kroBE100	2312.74	***162.05***	**1549.37**	182.44	+	2.09E+10	***2.00E+08***	**2.10E+10**	2.18E+08	+
	kroAB150	4191.46	***303.95***	**3049.38**	465.71	+	4.26E+10	***5.00E+08***	**4.31E+10**	5.10E+08	+
	kroAB200	6015.10	***346.88***	**4507.94**	535.68	+	7.55E+10	***7.58E+08***	**7.64E+10**	7.84E+08	+
	euclidAB100	2253.18	200.06	**1521.55**	***176.57***	+	1.64E+10	2.55E+08	**1.66E+10**	***1.66E+08***	+
	euclidAB300	8894.67	***371.71***	**7713.76**	782.76	+	1.59E+11	***1.22E+09***	**1.61E+11**	1.76E+09	+
	euclidBC300	9259.40	***579.64***	**7946.22**	742.97	+	1.65E+11	***1.51E+09***	**1.67E+11**	1.59E+09	+
	euclidCD300	9129.04	***644.70***	**7916.04**	904.21	+	1.67E+11	1.85E+09	**1.69E+11**	***1.54E+09***	+
	euclidDE300	9642.32	***486.66***	**7988.70**	935.34	+	1.67E+11	***1.37E+09***	**1.69E+11**	1.98E+09	+
	euclidAB500	**12668.70**	***1026.73***	13798.30	1376.93	–	4.77E+11	3.81E+09	**4.79E+11**	***3.67E+09***	+
3	kroABC100	9.78E+03	***1.55E+02***	**4.78E+03**	1.91E+02	+	2.96E+15	4.26E+13	**3.18E+15**	***4.22E+13***	+
	kroBCD100	7.86E+03	2.39E+02	**4.82E+03**	***1.67E+02***	+	2.68E+15	***3.56E+13***	**2.78E+15**	3.72E+13	+
	kroCDE100	7.99E+03	2.60E+02	**5.18E+03**	***1.52E+02***	+	3.26E+15	4.95E+13	**3.33E+15**	***4.10E+13***	+
	euclidABC300	3.49E+04	***4.58E+02***	**1.87E+04**	5.38E+02	+	7.01E+16	***5.70E+14***	**7.44E+16**	8.79E+14	+
	euclidBCD300	3.20E+04	***4.13E+02***	**1.88E+04**	7.30E+02	+	6.85E+16	***7.76E+14***	**7.15E+16**	9.19E+14	+
	euclidCDE300	2.95E+04	***4.36E+02***	**1.90E+04**	7.39E+02	+	6.96E+16	***6.75E+14***	**7.15E+16**	1.39E+15	+

W-test (Wilcoxon signed-rank test) has been made at 5% significance level to compare the IGD/HV values of MOEA/D-ACO against at MS-GPSO/D. "+" denotes that mean IGD/HV value of MS-GPSO/D is significantly better than that of MOEA/D-ACO, and " – " denotes that the IGD/HV value of MOEA/D-ACO is better, "*" denotes that the IGD/HV values to be compared are not significantly different.

For comparison, GPSO is implemented in MS-PSO/D with the weighted-sum decomposition approach, i.e., MS-GPSO/D^W. Population size $ps = 45$ for each instance. Neighborhood size $T = 10$. About parameters for diversity control, $p\text{Max} = 1.0$, $MaxUsedT = 2$, $plearnX = 0.8$ for 100-city instances, $plearnX = 0.99$ for the others. For parameters in GPSO, $c_1 = c_2 = 2.0$, ω changes linearly from 0.9 to 0.4.

Parameter settings of MOEA/D-ACO for bi-objective TSPs have been recommended in [13]. For tri-objective TSPs, based on substantial experiments to explore MOEA/D-ACO, we set $ps = 45$, number of groups $K = 6$, and keep the other parameters unchanged. Considering MOEA/D-ACO with the Tchebycheff decomposition approach, i.e., MOEA/D-ACO^T, performs better than MOEA/D-ACO with the weighted-sum approach when dealing with MOTSP [13], only the results of MOEA/D-ACO^T are reported in this paper.

2) Compared Results: The IGD and HV statistics of the final approximations obtained by MOEA/D-ACO^T and MS-GPSO/D^W are compared in Table I. Better mean values of IGD and HV are marked in bold. Results of Wilcoxon signed-rank test of 5% significance level on IGD and HV are listed in columns "W-test." From Table I, we observe the following phenomena.

(1) The comparative results on an IGD metric and an HV metric are consistent. Except on instance euclidAB500, each algorithm has a consistent performance in terms of metric HV and metric IGD on all instances.

(2) Both the mean values and W-test results of IGD and HV show that MS-GPSO/D^W performs significantly better than MOEA/D-ACO^T on all 15 bi-objective and 6 tri-objective instances except that MOEA/D-ACO^T wins on euclidAB500 in terms of IGD. In fact, better IGD requires that the estimated PF obtained by an algorithm should be denser and closer to the true PF, with fewer missing parts. And better HV requires that the estimated PF of an algorithm should be generally wilder and farther from the reference point. Thus, the results can illustrate that compared with MOEA/D-ACO^T, the final approximations obtained by MS-GPSO/D^W are both denser and wilder. And the effectiveness of MS-GPSO/D^W for MOTSP is hence proven.

(3) Almost on all instances, quite obvious differences can be observed between mean values of IGD and HV obtained by MS-GPSO/D^W and MOEA/D-ACO^T. Particularly, on many instances of smaller scale, such as 100-city ones, the ratios of IGD values obtained by MS-GPSO/D^W to MOEA/D-ACO^T's are nearly as low as 1/2. The differences show MS-GPSO/D^W's performance advantage over MOEA/D-ACO^T.

(4) MS-GPSO/D^W obtains lower IGD deviation on 9 out of 21 instances and lower HV deviation on 11 out of 21 instances. Thus, it has no obvious superiority to MOEA/D-ACO^T as far as deviation is concerned. However, although poor deviation is possible,

MS-GPSO/D^W can still achieve better IGD or HV in a random run owing to its significantly better mean values as stated above. Therefore, the robustness of MS-GPSO/D^W can be promised by its pretty good performance.

3) Comparison Under More Evaluations: Meanwhile, final approximations of MS-GPSO/D^W and MOEA/D-ACO^T are further compared under more evaluations than 24*3000. That is, available evaluations is tripled on tri-objective instances, i.e., 24*9000, and on bi-objective instances, an algorithm stops only when its approximation cannot be improved in 500 iterations. Compared results are provided in Table SII in supplementary material, where better performance of MS-GPSO/D^W can still be observed on every instance in terms of both IGD and HV. Thus, the results show that the superiority of MS-GPSO/D^W to MOEA/D-ACO^T can sustain even when more computing resources are available.

4) Comparison of Time Complexity: In Table SIII in supplementary material, we show the average time (in seconds) of 30 runs of MOEA/D-ACO^T and MS-GPSO/D^W on Intel Core i3-3240. For MS-GPSO/D^W, we record the time consumed for updating velocities (updateV) and solutions (updateX); for MOEA/D-ACO^T, we record the time consumed for updating solutions (updateX), pheromone matrices (Pheromone), and probability matrices (ProbMatrix). As shown in bold, on each instance, MS-GPSO/D^W consumes less time compared with MOEA/D-ACO^T, and the differences are more obvious on larger-scale instances. Meanwhile, differences on time of updateX are consistent with the differences on total consumed time, which implies that the way of updating solutions in MS-GPSO/D^W mainly helps to reduce its time complexity.

In fact, the above results are as expected. Regardless of the common MOEA/D part in MS-PSO/D and MOEA/D-ACO, the main difference of their time complexity should come from the difference between S-PSO and ACO. When solving the same problem, the time of maintaining velocities in S-PSO is roughly equal to the time of maintaining pheromone matrix in ACO. As for solution update, the worst case in S-PSO is that all elements are learnt from the set E, but a more common case is that some elements are learnt from set V or set X. Correspondingly, ACO generates a new solution according to the probability matrix, which resembles the worst case in S-PSO, but it needs to spend extra time to update the probability matrix. Thus, theoretically, MS-PSO/D should have a similar or lower time complexity compared with MOEA/D-ACO.

C. Parameter Investigation

1) Population Size: To explore the appropriate setting of population size, we compare the performance of MS-GPSO/D^W with *ps* set as 24, 45, and 72 on bi-objective instances. IGD statistics are given in Table SIV in supplementary material, where we find that MS-GPSO/D^W with a larger population size is able to obtain better IGD. MS-GPSO/D^W (ps72) performs best on 8 out of 15 instances and MS-GPSO/D^W (ps45) performs best on 7 other instances. Recall that an MOP can be more thoroughly solved by more particles, but a smaller portion of the given computing resources can be allocated to each particle. Thus, a faster convergence is required by MS-PSO/D of a larger *ps*. Owing to the good convergence of GPSO, MS-GPSO/D^W with a larger *ps* can have a better performance.

For the sake of fairness, MOEA/D-ACO^T with *ps* set as 24, 45, and 72 are also compared on bi-objective instances, with results reported in Table SV in supplementary material. We observe that MOEA/D-ACO^T with 45 ants can obtain the best IGD on more instances. Therefore, it should be reasonable to compare MOEA/D-ACO and MS-PSO/D on bi-objective instances with *ps* set as 45.

2) Diversity Control: The diversity of MS-PSO/D is expected in a *reasonable* range such that MS-PSO/D can converge quickly as well as preserve the potential for finding more efficient solutions. Taking parameter *plearnX* as an example, we show efficiency of the diversity control mechanism suggested in this paper.

First, to explain how *plearnX* affects diversity of MS-PSO/D, we run MS-GPSO/D^W with *plearnX* set as 0.6, 0.8, and 0.99 separately and record the average percentage of elements chosen from velocity in each iteration on instances kroAB100, euclidAB500, and kroABC100. As shown in Fig. S3 in supplementary material, on each instance, the percentage converges to a smaller value as *plearnX* increases. Since in S-PSO, a larger diversity implies more available elements in velocity [33], the results prove that a larger value of *plearnX* leads to a lower diversity.

Then, we compare the final approximations obtained by MS-GPSO/D^W with *plearnX* set as 0.6, 0.8, and 0.99. IGD statistics are given in Table SVI in supplementary material, with the best IGD shown in bold. For bi-objective instances, MS-GPSO/D^W ($plearnX = 0.8$) obtain the best IGD on 100-city instances, but MS-GPSO/D^W($plearnX = 0.99$) performs best on almost all the other larger-scale instances. It seems that faster convergence is expected when exploring larger decision spaces. For tri-objective instances, MS-GPSO/D^W with smaller *plearnX*, i.e., 0.6 or 0.8, performs better. The reason may be that more diversity is needed to obtain an approximated PF to cover a larger objective space under limited evaluations.

In conclusion, above results reflect that the diversity control mechanism of MS-PSO/D does work and different levels of diversity can be provided to achieve a good performance with given computing resources for different instances.

V. Behaviors on MMRCPSP

Fifteen MMRCPSP instances of 120, 150, and 200 tasks are generated by project generator PROGEN [75], [76] using sample setup as shown in Table SVII in supplementary material. Nonrenewable resources are not considered to avoid a need for repair operations after solution construction. And different resources are regarded as of equal importance, with the cost per unit of each kind of resource set as 1. Three popular objectives as defined in Section II are considered in tri-objective MMRCPSP cases, and the first two objectives are considered in bi-objective MMRCPSP cases.

Metrics IGD and HV are applied as well. Unless specified otherwise, all experimental results are based on 30 independent

TABLE II
IGD AND HV STATISTICS OF THE FINAL APPROXIMATIONS OBTAINED BY MOEA/D-ACO^W AND MS-GPSO/D^W ON 15 BI-OBJECTIVE MMRCPSP INSTANCES

INSTANCES	IGD					Hypervolume				
	MOEA/D ACO^W		MS-GPSO/D^W			MOEA/D ACO^W		MS-GPSO/D^W		
	mean	std dev	mean	std dev	W-test	mean	std dev	mean	std dev	W-test
j120.1	21.59	2.28	**10.60**	**1.95**	+	1.21E+05	2.16E+03	**1.31E+05**	**2.65E+03**	+
j120.2	26.39	3.34	**12.80**	**1.77**	+	1.36E+05	4.15E+03	**1.54E+05**	**3.83E+03**	+
j120.3	28.20	1.73	**11.00**	**1.80**	+	1.36E+05	2.69E+03	**1.54E+05**	**4.35E+03**	+
j120.4	25.75	1.32	**12.08**	**1.90**	+	1.37E+05	3.06E+03	**1.64E+05**	**5.84E+03**	+
j120.5	23.51	2.59	**8.72**	**1.54**	+	1.18E+05	3.68E+03	**1.30E+05**	**3.69E+03**	+
j150.1	28.36	2.54	**12.62**	**2.80**	+	1.47E+05	5.68E+03	**1.70E+05**	**4.81E+03**	+
j150.2	36.36	2.19	**12.50**	**1.86**	+	1.66E+05	4.43E+03	**1.97E+05**	**4.34E+03**	+
j150.3	32.45	3.20	**13.66**	**2.85**	+	1.80E+05	5.14E+03	**2.07E+05**	**6.38E+03**	+
j150.4	47.34	4.09	**14.84**	**3.05**	+	1.73E+05	5.38E+03	**1.96E+05**	**5.18E+03**	+
j150.5	29.49	1.66	**12.83**	**1.71**	+	1.98E+05	2.80E+03	**2.22E+05**	**5.30E+03**	+
j200.1	27.55	3.10	**15.74**	**2.14**	+	2.38E+05	4.41E+03	**2.66E+05**	**8.76E+03**	+
j200.2	39.89	3.58	**15.49**	**1.82**	+	2.79E+05	7.08E+03	**3.23E+05**	**7.33E+03**	+
j200.3	31.95	5.97	**13.85**	**1.68**	+	2.25E+05	5.79E+03	**2.63E+05**	**1.20E+04**	+
j200.4	32.25	3.51	**14.32**	**1.53**	+	3.26E+05	7.13E+03	**3.49E+05**	**7.70E+03**	+
j200.5	30.59	2.68	**14.24**	**1.51**	+	1.70E+05	4.96E+03	**2.07E+05**	**5.90E+03**	+

W-test (Wilcoxon signed-rank test) has been made at 5% significance level to compare the IGD or HV values of MOEA/D-ACO against at MS-GPSO/D. "+" denotes that mean IGD or HV value of the corresponding algorithm is significantly better than that of MOEA/D-ACO.

TABLE III
IGD AND HV STATISTICS OF THE FINAL APPROXIMATIONS OBTAINED BY MOEA/D-ACO^W AND MS-GPSO/D^W ON 15 TRI-OBJECTIVE MMRCPSP INSTANCES

INSTANCES	IGD					HV				
	MOEA/D-ACO^W		MS-GPSO/D^W			MOEA/D-ACO^W		MS-GPSO/D^W		
	mean	std dev	mean	std dev	W-test	mean	std dev	mean	std dev	W-test
j120_3.1	91.42	18.68	**49.29**	**7.40**	+	7.25E+08	8.39E+07	**7.72E+08**	**2.69E+07**	+
j120_3.2	102.08	20.76	**33.89**	**3.94**	+	6.29E+08	1.14E+08	**7.11E+08**	**3.07E+07**	+
j120_3.3	93.14	25.43	**31.77**	**3.53**	+	7.07E+08	1.26E+08	**7.76E+08**	**3.51E+07**	+
j120_3.4	102.00	23.11	**42.17**	**5.09**	+	9.08E+08	1.67E+08	**1.00E+09**	**4.43E+07**	+
j120_3.5	92.92	22.46	**41.28**	**4.17**	+	7.71E+08	1.13E+08	**8.41E+08**	**3.59E+07**	+
j150_3.1	142.95	23.55	**44.48**	**7.46**	+	8.38E+08	1.08E+08	**9.59E+08**	**5.53E+07**	+
j150_3.2	153.96	20.46	**39.34**	**4.09**	+	1.17E+09	1.21E+08	**1.35E+09**	**3.90E+07**	+
j150_3.3	155.68	35.28	**54.04**	**6.38**	+	1.51E+09	3.06E+08	**1.74E+09**	**7.32E+07**	+
j150_3.4	159.79	32.59	**45.20**	**6.16**	+	1.20E+09	2.38E+08	**1.41E+09**	**7.09E+07**	+
j150_3.5	131.50	37.39	**45.44**	**4.50**	+	1.45E+09	2.60E+08	**1.61E+09**	**6.72E+07**	+
j200_3.1	247.71	69.43	**59.60**	**7.38**	+	2.65E+09	5.54E+08	**3.08E+09**	**1.03E+08**	+
j200_3.2	213.18	46.69	**56.66**	**7.10**	+	3.18E+09	6.68E+08	**3.63E+09**	**1.46E+08**	+
j200_3.3	250.20	47.56	**69.47**	**8.86**	+	2.31E+09	4.37E+08	**2.74E+09**	**1.32E+08**	+
j200_3.4	245.65	37.32	**74.30**	**12.28**	+	4.32E+09	3.81E+08	**4.96E+09**	**1.83E+08**	+
j200_3.5	237.78	68.24	**53.35**	**5.08**	+	1.65E+09	4.68E+08	**2.04E+09**	**1.07E+08**	+

W-test (Wilcoxon signed-rank test) has been made at 5% significance level to compare the IGD or HV values of MOEA/D-ACO against at MS-GPSO/D. "+" denotes that mean IGD or HV value of the corresponding algorithm is significantly better than that of MOEA/D-ACO.

runs. Referring to [41], 10 000 solutions/schedules can be generated in total by an algorithm in each run.

A. Comparative Analysis

There are only a few MOEAs specially developed for MMRCPSP and most of them are simply built upon NSGA-II, e.g., [16] and [77]. We have applied NSGA-II to solve MMRCPSP instances generated in this paper, but it performs much worse than MS-PSO/D. On the other hand, MOEA/D-ACO has been adopted to solve a variant of MOPSP [17] and there should be great potential to apply it to MMRCPSP. Thus, we would like to still compare MS-GPSO/D with MOEA/D-ACO in order to verify MS-PSO/D's efficiency for MMRCPSP. For comparison, MOEA/D-ACO is carefully implemented by us for MMRCPSP, where solution representation and heuristic information refer to our proposed MS-PSO/D and pheromone design refers to [13].

1) Basic Settings: For both MS-GPSO/D and MOEA/D-ACO, population size is set as 45 for bi-objective case and tri-objective case, and neighborhood size is set as 1/6 of population size. For MS-GPSO/D, we set $\omega = 0.9$, $plearnX = 0.8$, $p\text{Max} = 0.9$, and $MaxUsedT = 1$. The other parameters of MS-GPSO/D and MOEA/D-ACO refer to their parameter settings for MOTSP. According to our experiments, weighted-sum decomposition approach performs better than Tchebycheff approach on both MOEA/D-ACO and MS-PSO/D. Thus, only the results of MOEA/D-ACO^W and MS-GPSO/D^W are considered.

2) Compared Results: IGD and HV statistics of the final approximations obtained by MOEA/D-ACO^W and MS-GPSO/D^W on bi-objective and tri-objective MMRCPSP instances are shown in Tables II and III, respectively. The results show the following.

(1) MS-GPSO/D^W outperforms MOEA/D-ACO^W on all instances in terms of both Wilcoxon test and mean

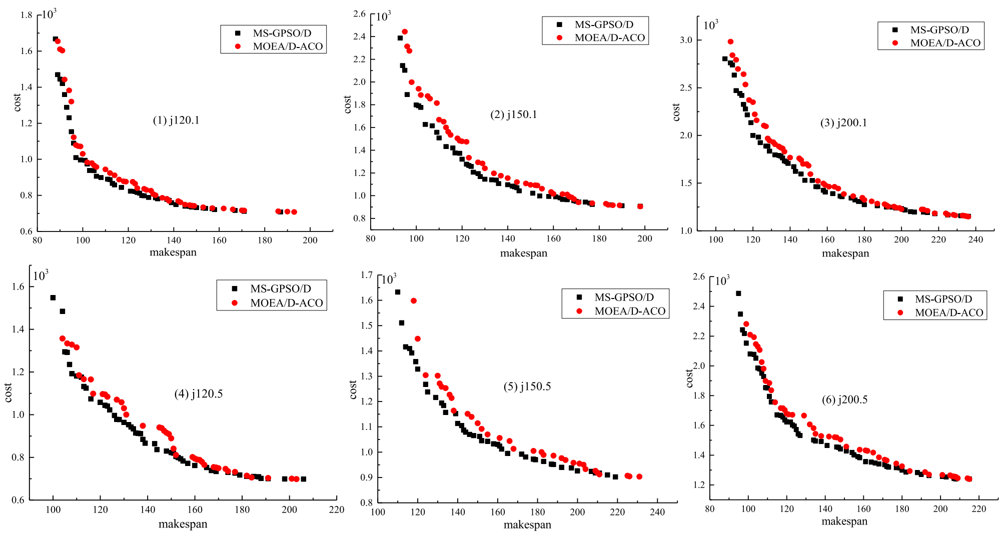

Fig. 3. Final Pareto fronts of 30 independent runs obtained by MS-GPSO/D^W and MOEA/D-ACO^W on bi-objective instances j120.1, j150.1, j200.1 and j120.5, j150.5, j200.5.

values of IGD/HV, regardless of the number of objectives or the scale of instances.

(2) On almost all instances, the ratio of mean IGD obtained by MS-GPSO/D^W to that obtained by MOEA/D-ACO^W is less than 1/2. On some tri-objective instances such as j150_3.4 and j200_3.4, the ratio is smaller than 1/3 even 1/4. The great difference demonstrates MS-GPSO/D's performance advantage over MOEA/D-ACO.

(3) Smaller deviation values of IGD and HV are obtained by MS-GPSO/D^W on all instances, which implies the stability of MS-GPSO/D when solving MMRCPSP.

For further insight into their performance, we show the final PFs of 30 independent runs obtained by MOEA/D-ACO^W and MS-GPSO/D^W on instances j120.1, j150.1, j200.1 and j120.5, j150.5, j200.5 in Fig. 3. Recall that an approximated PF is expected to be close to axes for multiobjective minimization problems. For each case shown in Fig. 3, we find that the PF of MS-GPSO/D^W is closer to axes and generally denser with fewer missing parts, compared with MOEA/D-ACO^W's.

In conclusion, MS-GPSO/D^W outperforms MOEA/D-ACO^W on MMRCPSP instances in terms of effectiveness, efficiency, as well as robustness.

B. Parameter Investigation

Here, we discuss how the performance of MS-GPSO/D^W on MMRCPSP can be influenced by parameters ω and *plearnX*.

1) Inertia Weight ω: A larger or smaller ω corresponds to a stronger focus on global search or local search, respectively. Given a certain amount of computing resources, different levels of global or local search should be expected by different problems. Final approximations obtained by MS-GPSO/D^W ($plearnX = 0.5$) with $\omega = 0.4$, linearly changed ω from 0.9 to 0.4, and $\omega = 0.9$, are compared and the IGD statistics are presented in Table SVIII (bi-objective instances) and Table SIX (tri-objective instances) in supplementary material.

The results show the following.

(1) For both bi-objective and tri-objective cases, MS-GPSO/D^W with $\omega = 0.4$ obtains the worst results, which means that a relatively higher ability of global search is expected to solve MMRCPSP problems.

(2) For the bi-objective case, there is no obvious difference between MS-GPSO/D^W($\omega = 0.9$) and MS-GPSO/D^W(linear ω) on most instances. However, for the tri-objective case, the former performs better on each instance. In fact, the different results on bi-objective and tri-objective cases are as expected, since the objective space grows exponentially as the number of objectives increases and a stronger global search is hence needed under limited computing resources.

2) Probability of Learning From X, plearnX: The discussions on MOTSP show that an appropriate setting of *plearnX* can improve MS-PSO/D's performance obviously. To discover how *plearnX* influences MS-PSO/D when solving MMRCPSP, we compare the final approximations obtained by MS-GPSO/D^W with $\omega = 0.9$ under different values of *plearnX*, i.e., 0.2, 0.5, and 0.8. IGD statistics are presented in Table SX (bi-objective case) and Table SXI (tri-objective case) in supplementary material.

MS-GPSO/D^W ($plearnX = 0.8$) is observed to perform best on each instance. Furthermore, MS-GPSO/D^W performs better as *plearnX* increases. Recall that a smaller value of *plearnX* often causes a higher diversity of MS-PSO/D. Therefore, the results reflect that MS-PSO/D is expected of more convergence rather than diversity when dealing with MMRCPSP with the given computing resources.

VI. Conclusion

In this paper, focusing on the permutation-based MOCOPs, a generic methodology termed MS-PSO/D, has been developed by combining S-PSO with MOEA/D. To customize MS-PSO/D for solving permutation-based MOCOPs, an element-based permutation representation and a constructive approach for permutation construction are formalized. For some problems with a more complex decision space, we demonstrate that an element-based representation can also be defined to represent extra decisions, and a constructive solution completion process can be further formalized to make extra decisions. Furthermore, owing to the constructive approach for solution construction, both online and offline heuristic information can be easily introduced into MS-PSO/D for efficiency.

Two representative MOCOPs, i.e., MOTSP and MMRCPSP, are extensively tested to verify the efficiency of the proposed MS-PSO/D methodology. In the experiments, the effectiveness of parameters for diversity control in MS-PSO/D has been proven and the efficiency of MS-PSO/D as a generic method has been verified through comparing with MOEA/D-ACO, which is one of the best-so-far decomposition-based MOEAs for solving MOCOPs.

REFERENCES

[1] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin, Germany: Springer, 2003.

[2] M. Ehrgott and X. Gandibleux, “A survey and annotated bibliography of multiobjective combinatorial optimization,” *OR-Spektrum*, vol. 22, no. 4, pp. 425–460, 2000.

[3] X. Cai, Y. Li, Z. Fan, and Q. Zhang, “An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization,” *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 508–523, Aug. 2015.

[4] 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.

[5] N. Chen *et al.*, “An evolutionary algorithm with double-level archives for multiobjective optimization,” *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1851–1863, Sep. 2015.

[6] Q. Lin *et al.*, “A hybrid evolutionary immune algorithm for multiobjective optimization problems,” *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 711–729, Oct. 2016.

[7] Z.-H. Zhan *et al.*, “Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems,” *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[8] Y.-L. Li, Y.-R. Zhou, Z.-H. Zhan, and J. Zhang, “A primary theoretical study on decomposition-based multiobjective evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 563–576, Aug. 2016.

[9] Y. Mei, K. Tang, and X. Yao, “Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem,” *IEEE Trans. Evol. Comput.*, vol. 15, no. 2, pp. 151–165, Apr. 2011.

[10] M. Ehrgott and X. Gandibleux, “Approximative solution methods for multiobjective combinatorial optimization,” *Top*, vol. 12, no. 1, pp. 1–63, 2004.

[11] V. A. Shim, K. C. Tan, and C. Y. Cheong, “A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 5, pp. 682–691, Sep. 2012.

[12] C. Changdar, G. S. Mahapatra, and R. K. Pal, “An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness,” *Swarm Evol. Comput.*, vol. 15, pp. 27–37, Apr. 2014.

[13] L. Ke, Q. Zhang, and R. Battiti, “MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony,” *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.

[14] A. Jaszkiewicz and P. Zielniewicz, “Pareto memetic algorithm with path-relinking for biobjective traveling salesman problem,” *Eur. J. Oper. Res.*, vol. 193, no. 3, pp. 885–890, 2009.

[15] V. Yannibelli and A. Amandi, “Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan,” *Eng. Optim.*, vol. 45, no. 1, pp. 45–65, Jan. 2013.

[16] S. C. Vanucci, E. G. Carrano, R. D. Bicalho, and R. H. Takahashi, “A modified NSGA-II for the multiobjective multi-mode resource-constrained project scheduling problem,” in *Proc. IEEE CEC*, Brisbane, QLD, Australia, 2012, pp. 1–7.

[17] J. Xiao, M.-L. Gao, and M.-M. Huang, “Empirical study of multi-objective ant colony optimization to software project scheduling problems,” in *Proc. GECCO*, Madrid, Spain, 2015, pp. 759–766.

[18] J. Xiong, J. Liu, Y. Chen, and H. A. Abbass, “A knowledge-based evolutionary multiobjective approach for stochastic extended resource investment project scheduling problem,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 742–763, Oct. 2014.

[19] K. Ghoseiri and S. F. Ghannadpour, “Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm,” *Appl. Soft Comput.*, vol. 10, no. 4, pp. 1096–1107, 2010.

[20] Y. Qi, Z. Hou, H. Li, J. Huang, and X. Li, “A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows,” *Comput. Oper. Res.*, vol. 62, pp. 61–77, Oct. 2015.

[21] J. Wang *et al.*, “Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms,” *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 582–594, Mar. 2016.

[22] B. Ombuki, B. J. Ross, and F. Hanshar, “Multi-objective genetic algorithms for vehicle routing problem with time windows,” *Appl. Intell.*, vol. 24, no. 1, pp. 17–30, Feb. 2006.

[23] J. J. S. Chávez, J. W. Escobar, and M. G. Echeverri, “A multi-objective Pareto ant colony algorithm for the multi-depot vehicle routing problem with backhauls,” *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 35–48, 2016.

[24] J. Zhang, W. Wang, Y. Zhao, and C. Cattani, “Multiobjective quantum evolutionary algorithm for the vehicle routing problem with customer satisfaction,” *Math. Problems Eng.*, vol. 2012, 2012, Art. no. 879614.

[25] O. Kaiwartya *et al.*, “Multiobjective dynamic vehicle routing problem and time seed based solution using particle swarm optimization,” *J. Sensors*, vol. 2015, 2015, Art. no. 189832.

[26] M. M. Yenisey and B. Yagmahan, “Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends,” *Omega*, vol. 45, pp. 119–135, Jun. 2014.

[27] B.-B. Li and L. Wang, “A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 576–591, Jun. 2007.

[28] B. Yagmahan and M. M. Yenisey, “Ant colony optimization for multi-objective flow shop scheduling problem,” *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 411–420, 2008.

[29] B.-B. Li, L. Wang, and B. Liu, “An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling,” *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 818–831, Jul. 2008.

[30] B. Qian *et al.*, “A hybrid differential evolution method for permutation flow-shop scheduling,” *Int. J. Adv. Manuf. Technol.*, vol. 38, nos. 7–8, pp. 757–777, 2008.

[31] 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.

[32] 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.

[33] W.-N. Chen *et al.*, “A novel set-based particle swarm optimization method for discrete optimization problems,” *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.

[34] Y. Shi, “Particle swarm optimization,” *IEEE Neural Netw. Soc.*, vol. 2, no. 1, pp. 8–13, Feb. 2004.

[35] Y.-J. Gong *et al.*, “Genetic learning particle swarm optimization,” *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[36] W.-N. Chen *et al.*, “Particle swarm optimization with an aging leader and challengers,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[37] Q. Yang *et al.*, “Segment-based predominant learning swarm optimizer for large-scale optimization,” *IEEE Trans. Cybern.*, to be published.

[38] V. Selvi and D. R. Umarani, “Comparative analysis of ant colony and particle swarm optimization techniques,” *Int. J. Comput. Appl.*, vol. 5, no. 4, pp. 1–6, 2010.

[39] A. Zhou *et al.*, “Multiobjective evolutionary algorithms: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[40] W.-N. Chen, J. Zhang, H. S.-H. Chung, R.-Z. Huang, and O. Liu, “Optimizing discounted cash flows in project scheduling—An ant colony optimization approach,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 1, pp. 64–77, Jan. 2010.

[41] F. Ballestín and R. Blanco, “Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems,” *Comput. Oper. Res.*, vol. 38, no. 1, pp. 51–62, 2011.

[42] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, NY, USA: Oxford Univ. Press, 1996.

[43] M. Kumar, M. Husian, N. Upreti, and D. Gupta, "Genetic algorithm: Review and application," *Int. J. Inf. Technol. Knowl. Manag.*, vol. 2, no. 2, pp. 451–454, 2010.

[44] X.-Y. Zhang *et al.*, "Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 695–710, Oct. 2016.

[45] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[46] W.-J. Yu *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, Jul. 2014.

[47] Y.-L. Li *et al.*, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.

[48] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. New York, NY, USA: Springer, 2011, pp. 760–766.

[49] Y.-H. Jia *et al.*, "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[50] M. Shen *et al.*, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7141–7151, Dec. 2014.

[51] Y.-J. Gong *et al.*, "Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 254–267, Mar. 2012.

[52] Y.-J. Gong *et al.*, "Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination," *IEEE Trans. Ind. Informat.*, vol. 8, no. 4, pp. 900–912, Nov. 2012.

[53] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, nos. 2–3, pp. 243–278, 2005.

[54] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 191–205, Apr. 2017.

[55] M. Shen, W.-N. Chen, J. Zhang, H. S.-H. Chung, and O. Kaynak, "Optimal selection of parameters for nonuniform embedding of chaotic time series using ant colony optimization," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 790–802, Apr. 2013.

[56] W.-N. Chen and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 1–17, Jan. 2013.

[57] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

[58] L. Tang and X. Wang, "A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 17, no. 1, pp. 20–45, Feb. 2013.

[59] 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.

[60] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1808–1820, Oct. 2014.

[61] N. Al Moubayed, A. Petrovski, and J. McCall, "D^{2}MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces," *Evol. Comput.*, vol. 22, no. 1, pp. 47–77, Mar. 2014.

[62] S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 442–446, Jun. 2012.

[63] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 82–97, Feb. 2014.

[64] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer Acad., 1999.

[65] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[66] P. K. Shukla, "On the normal boundary intersection method for generation of efficient front," in *Computational Science—ICCS 2007*, Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds. Berlin, Germany: Springer, 2007, pp. 310–317.

[67] Y. I. Lim, P. Floquet, and X. Joulia, "Efficient implementation of the normal boundary intersection (NBI) method on multiobjective optimization problems," *Ind. Eng. Chem. Res.*, vol. 40, no. 2, pp. 648–655, 2001.

[68] Y. Qi *et al.*, "MOEA/D with adaptive weight adjustment," *Evol. Comput.*, vol. 22, no. 2, pp. 231–264, Jun. 2014.

[69] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 333–346, Aug. 2002.

[70] *Traveling Salesman Problem Library—TSPLIB.* Accessed on Jul. 22, 2017. [Online]. Available: http://eden.dei.uc.pt/~paquete/tsp/

[71] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *Proc. MICAI*, Mexico City, Mexico, 2004, pp. 688–697.

[72] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[73] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 86–95, Feb. 2012.

[74] *Professor Qingfu Zhang.* Accessed on Jul. 22, 2017. [Online]. Available: http://dces.essex.ac.uk/staff/qzhang/

[75] R. Kolisch and A. Sprecher, "PSPLIB—A project scheduling problem library: OR software—ORSEP operations research software exchange program," *Eur. J. Oper. Res.*, vol. 96, no. 1, pp. 205–216, 1997.

[76] *Project Scheduling Problem Library—PSPLIB.* Accessed on Jul. 22, 2017. [Online]. Available: http://www.om-db.wi.tum.de/psplib/main.html

[77] M. Amiri, A.-R. Abtahi, and K. Khalili-Damghani, "Solving a generalised precedence multi-objective multi-mode time-cost-quality trade-off project scheduling problem using a modified NSGA-II algorithm," *Int. J. Service Oper. Manag.*, vol. 14, no. 3, pp. 355–372, 2013.

Xue Yu (S'15) received the bachelor's degree from Sun Yat-sen University, Guangzhou, China, in 2015, where she is currently pursuing the Ph.D. degree.

She is also a Research Assistant with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation algorithms and their applications on large scale computing, autonomous path planning, and intelligent transportation.

Wei-Neng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has published over 70 papers in international journals and conferences. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award for Doctoral Thesis in 2016 and the National Science Fund for Excellent Young Scholars in 2016.

Tianlong Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Huaxiang Zhang received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2004,

He was an Associated Professor with the Department of Computer Science, Shandong Normal University, Jinan, China, from 2004 to 2005, where he is currently a Professor with the School of Information Science and Engineering. He has authored over 100 journal and conference papers and has been granted eight invention patents. His current research interests include machine learning, pattern recognition, evolutionary computation, and Web information processing.

Huaqiang Yuan received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1996.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include computational intelligence and cyberspace security.

Sam Kwong (M'93–SM'04–F'14) received the B.Sc. degree from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from Fernuniversität Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Toronto, ON, Canada, where he designed the Diagnostic Software to detect the manufacture faults of the very large scale integration chips in the Cyber 430 machine. He is currently the Head and a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the *Journal of Information Science*. He was elevated to IEEE Fellow for his contributions on Optimization Techniques for Cybernetics and Video coding in 2014. He is also appointed as an IEEE Distinguished Lecturer for IEEE SMC society in 2017. He is currently the Vice President of Conferences and Meetings with IEEE Systems, Man and Cybernetics.

Jun Zhang (M'02–SM'08–F'16) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Professor with the South China University of Technology, Guangzhou, China, where he was also appointed as the Changjiang Chair Professor in 2013. His current research interests include computational intelligence, cloud computing, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 50 IEEE TRANSACTIONS papers in the above areas.

Prof. Zhang was a recipient of the National Science Fund for Distinguished Young Scholars in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and the Current Chair of the IEEE Guangzhou Subsection, IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters and the ACM Guangzhou Chapter.