# Many-Objective Job-Shop Scheduling: A Multiple Populations for Multiple Objectives-Based Genetic Algorithm Approach

Si-Chen Liu, *Student Member, IEEE*, Zong-Gan Chen, *Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Sang-Woon Jeon, *Member, IEEE*, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The job-shop scheduling problem (JSSP) is a challenging scheduling and optimization problem in the industry and engineering, which relates to the work efficiency and operational costs of factories. The completion time of all jobs is the most commonly considered optimization objective in the existing work. However, factories focus on both time and cost objectives, including completion time, total tardiness, advance time, production cost, and machine loss. Therefore, this article first time proposes a many-objective JSSP that considers all these five objectives to make the model more practical to reflect the various demands of factories. To optimize these five objectives simultaneously, a novel multiple populations for multiple objectives (MPMO) framework-based genetic algorithm (GA) approach, called MPMOGA, is proposed. First, MPMOGA employs five populations to optimize the five objectives, respectively. Second, to avoid each population only focusing on its corresponding single objective, an archive sharing technique (AST) is proposed to store the elite solutions collected from the five populations so that the populations can obtain optimization information about the other objectives from the archive. This way, MPMOGA can approximate different parts of the entire Pareto front (PF). Third, an archive update strategy (AUS) is proposed to further improve the quality of the solutions in the archive. The test instances in the widely used test sets are adopted to evaluate the performance of MPMOGA. The experimental results show that MPMOGA outperforms the compared state-of-the-art algorithms on most of the test instances.**

***Index Terms*—Archive sharing technique (AST), archive update strategy (AUS), genetic algorithm (GA), many-objective job-shop scheduling problem (MaJSSP), many-objective optimization, multiple populations for multiple objectives (MPMO).**

Manuscript received 16 March 2021; revised 22 June 2021; accepted 22 July 2021. Date of publication 13 September 2021; date of current version 15 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102; in part by the National Natural Science Foundations of China (NSFC) under Grant 62176094, Grant 61822602, Grant 61772207, and Grant 61873097; in part by the Key-Area Research and Development of Guangdong Province under Grant 2020B010166002; in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003; and in part by the Hong Kong GRF-RGC General Research Fund 9042816 (CityU 11209819). This article was recommended by Associate Editor H. Su *(Corresponding authors: Zong-Gan Chen; Zhi-Hui Zhan.)*

Si-Chen Liu and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, also with the Pazhou Laboratory, Guangzhou 510330, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

Zong-Gan Chen is with the Network Information Security Department, Guangdong Police College, Guangzhou 510440, China (e-mail: charleszg@qq.com).

Sang-Woon Jeon is with the Department of Military Information Engineering, Hanyang University, Ansan 15588, South Korea.

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

Jun Zhang is with Hanyang University, Ansan 15588, South Korea, also with the Zhejiang Normal University, Jinhua 321004, China, and also with the Chaoyang University of Technology, Taichung 413310, Taiwan.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3102642.

Digital Object Identifier 10.1109/TCYB.2021.3102642

## I. Introduction

JOB-SHOP scheduling problems (JSSPs) exist in various industrial and engineering management fields, such as printed circuit board production [1], garment manufacturing supply chains [2], and cloud computing [3]. In JSSPs, there are a set of jobs with the same number of procedures to be processed, and an industrial factory should determine the process orders of the procedures of all jobs on the available machines for achieving certain objectives. Take Fig. 1 as an example to illustrate the details of JSSPs. There are four jobs (i.e., $J_1$, $J_2$, $J_3$, and $J_4$) and two machines (i.e., $M_1$ and $M_2$). Each job contains two procedures and each procedure is denoted by $O_{ji}$, where $j$ represents the job it belongs to and $i$ represents the process ranking. Note that the procedures of a job should be processed according to the ranking, for example, $O_{12}$ must be processed after $O_{11}$ is finished. Fig. 1(a) shows a procedure-to-machine allocation scheme that is static and fixed because each procedure, depending on its function, must be processed on a specific machine. Concretely, $O_{11}$, $O_{22}$, $O_{31}$, and $O_{42}$ should be processed on $M_1$, while $O_{12}$, $O_{21}$, $O_{32}$, and $O_{41}$ should be processed on $M_2$. The key issue in JSSPs is to arrange the process orders of the procedures for both $M_1$ and $M_2$. A feasible schedule is shown in Fig. 1(b), in which the process order of $M_1$ is $O_{31}$, $O_{22}$, $O_{11}$, and $O_{42}$, and the process order of $M_2$ is $O_{21}$, $O_{12}$, $O_{41}$, and $O_{32}$. Note that a machine can only process one procedure at the same time. The quality of the schedules can be evaluated by completion time, total tardiness, and some other objectives. In summary, JSSPs require algorithms to determine the process order of procedures on each machine for optimizing certain objectives under various constraints.

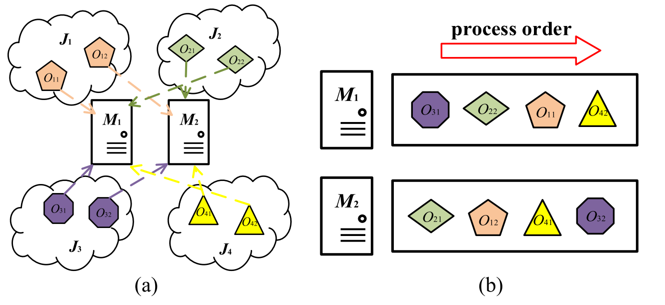

Fig. 1. Example of a JSSP. (a) Procedure-to-machine allocation scheme. (b) Schedule that determines the process order of procedures on each machine.

In the literature, JSSPs have been extensively studied and the related research can be mainly classified into the following three categories.

The first category is the single-objective JSSP model. Most of the single-objective JSSP models set the completion time as the optimization objective. Inspired by the connections between scheduling problems and mixed graph colorings [4], Al-Anzi *et al.* [5] converted the JSSP into a coloring problem in graph theory and solved the completion time minimization JSSP by searching for the optimal coloring solution of a special mixed graph. The tabu search (TS) shows the feasibility in solving single-objective JSSPs because of the strong local search ability [6]. However, the lack of diverse solutions weakens the performance of TS. Therefore, a hybrid algorithm based on particle swarm optimization and TS is proposed to enhance the global search ability for further reducing the completion time [7]. Besides, Peng *et al.* [8] combined the TS with a path relinking algorithm. Recently, Viana *et al.* [9] proposed a genetic algorithm (GA) with perturbation functions to avoid premature convergence.

The second category is the multiobjective JSSP model. An energy-efficient model that optimizes both completion time and the total energy consumption is adopted in [10] and [11]. To solve the energy-efficient model, a green GA (GGA) [10] and a multiobjective artificial bee colony [11] are proposed. Specifically, in GGA, each solution has a strength value to measure its performance on the two objectives. Nguyen *et al.* [12] built a three-objective dynamic JSSP model that considered completion time, total weighted tardiness, and mean absolute percentage error. Note that the mean absolute percentage error objective is related to the dynamic feature and measures the error in flowtime estimation. A diversified multiobjective cooperative coevolution algorithm is proposed to solve this three-objective dynamic JSSP model and shows better performance than some classic multiobjective evolutionary algorithms, including NSGA-II [13] and SPEA2 [14]. Meng *et al.* [15] constructed a model that aimed at minimizing completion time, average flow time, and machine idle time simultaneously. A dual-population particle swarm optimization algorithm is designed for the model.

The third category is the many-objective JSSP (MaJSSP) model with more than three objectives. Specifically, a four-objective model (i.e., mean flow time, maximum flow time, mean weighted tardiness, and maximum weighted tardiness) is constructed in [16]–[18]. To solve this model, a genetic programming-based nondominated sorting GA approach [16], a reference point adaption method [17], and a fitness-based selection technique [18] are proposed.

As seen from the literature, most of the existing work only considers the time-related objectives, regardliess of whether it is the single-objective or multiobjective, or even many-objective JSSP models. For example, in the multi/many-objective JSSP model, the flow time and tardiness are both calculated according to the completion time. Moreover, the mean and maximum flow time may not strongly conflict, the same would apply for the mean and maximum weighted tardiness. However, in industry and engineering management, the cost is also a deeply concerned objective besides time.

To alleviate the limitations of the existing models, we additionally consider the production cost and machine loss which can better meet the demands of factories. Hence, an MaJSSP model is proposed, which contains five optimization objectives: 1) completion time; 2) total tardiness; 3) advance time; 4) production cost; and 5) machine loss. Considering these five objectives to construct the MaJSSP is an innovation in the JSSP model research, which makes the model more practical in industry and engineering management. Since the time and cost in MaJSSP are conflicting objectives [19]–[21], the algorithm design should deal with the difficulty of how to balance the optimization of time and cost.

To efficiently solve the proposed MaJSSP, the multiple populations for multiple objectives (MPMO) [22] framework that has shown excellent performance on both multiobjective optimization [23], [24] and many-objective optimization [25] is adopted. Evolutionary computation algorithms are promising optimizers for various optimization problems [26]–[29]. In particular, the GA has shown powerful efficiency in solving discrete optimization problems [30], [31]. Therefore, the GA optimizer is embedded with the MPMO framework, thus generating a new MPMOGA approach to solve MaJSSP. Under the MPMO framework, MPMOGA employs five populations to optimize the five optimization objectives, respectively. During the evolutionary process, an archive is constructed to store the elite solutions found by the five populations. An archive sharing technique (AST) is designed to avoid the populations only focusing on their own optimization objective and achieve the coevolution of different populations. Moreover, an archive update strategy (AUS) is developed for enhancing the quality of the solutions in the archive.

The contributions of this article are as follows.

1) To the best of our knowledge, this article is the first paper that comprehensively considers the completion time, total tardiness, advance time, production cost, and machine loss to construct an MaJSSP model. The proposed MaJSSP model considers not only the time-related objectives but also the production cost and machine loss. Thus, our MaJSSP model can reflect the various demands of factories in a more practical way.
2) To effectively solve the MaJSSP, the MPMOGA is proposed, which is based on the MPMO framework. In addition, an AST and an AUS are incorporated into MPMOGA to enhance search efficiency.
3) The experiments are conducted on two widely used test sets (i.e., FT test set [32] and LA test set [33]). Compared to four state-of-the-art algorithms,

including three many-objective evolutionary algorithms (MaOEAs) and a JSSP algorithm, our proposed MPMOGA approach not only can find more diverse solutions distributed in the objective space but also can obtain better-converged solutions on each objective.

The remainder of this article is organized as follows. Section II introduces the basic concepts of the many-objective optimization problem (MaOP) and the proposed MaJSSP model. Then, the detailed descriptions about the proposed MPMOGA approach are given in Section III. In Section IV, the experimental results are presented and analyzed, and the advantages of MPMOGA in solving the MaJSSP are also discussed. Finally, we draw a conclusion in Section V.

## II. MaOP and MaJSSP Model

### A. MaOP

In recent years, MaOPs have attracted wide attention [34]–[36]. In general, a minimization MaOP can be formulated as

$$\min F(x) = (f_1(x), \ldots, f_k(x), \ldots, f_K(x)) \tag{1}$$

where $x$ is a decision vector that represents a solution of the MaOP. $F(x)$ represents the fitness value of $x$ on each objective. $K$ denotes the number of optimization objectives and is larger than three in MaOPs. Usually, one solution cannot be optimal on every objective due to the conflict among the objectives [37]. Therefore, the Pareto dominance rule is used to compare two solutions in MaOPs. As shown in (2), the solution $x_1$ dominates (i.e., is better than) the solution $x_2$ if $x_1$ performs better than $x_2$ on at least one objective and does not perform worse than $x_2$ on any objective. The solutions that are not dominated by any other solution are called nondominated solutions. The Pareto set (PS) stores all the nondominated solutions and the Pareto front (PF) stores the corresponding fitness of the nondominated solutions in PS

$$\left.\begin{array}{ll} \forall k \in [1, K] & f_k(x_1) \le f_k(x_2) \\ \exists k' \in [1, K] & f_{k'}(x_1) < f_{k'}(x_2) \end{array}\right\} \Rightarrow x_1 \prec x_2. \tag{2}$$

### B. MaJSSP Model

In MaJSSP, the number of jobs is denoted by $J$ and each job contains $M$ procedures of different types. The $M$ procedures of a job should be processed on $M$ respective machines, and the processing time of each procedure is predefined and fixed. The task of MaJSSP is to determine the process order of procedures on each machine while satisfying the following constraints.

*Constraint 1:* All jobs are released at time 0.

*Constraint 2:* One machine can only process one procedure simultaneously.

*Constraint 3:* No interruption is allowed during the processing of a machine. That is, one machine cannot process another procedure until finishing the current processing procedure.

*Constraint 4:* The procedures of a job should be processed sequentially according to a predetermined ranking.

*Constraint 5:* Each procedure can only be processed once.

The MaJSSP considers completion time, total tardiness, advance time, production cost, and machine loss as the five optimization objectives, which are formulated as follows:

$$\min F = (f_1, f_2, f_3, f_4, f_5) \tag{3}$$

$$f_1 = \max_{j=1}^{J} C_j \tag{4}$$

$$f_2 = \sum_{j=1}^{J} \max\left(C_j - D_j, 0\right) \tag{5}$$

$$f_3 = \sum_{j=1}^{J} \min\left(C_j - D_j, 0\right) \tag{6}$$

$$f_4 = \sum_{m=1}^{M} (work\_time_m \times wpc + sleep\_time_m \times spc) \tag{7}$$

$$f_5 = \sum_{m=1}^{M} count_m \tag{8}$$

where $f_1$ represents the maximum completion time of all jobs and $C_j$ represents the completion time of job $j$. $f_2$ represents the total tardiness and $D_j$ denotes the due date of job $j$. If $C_j$ is larger than $D_j$, $\max(C_j - D_j, 0) = C_j - D_j$ is viewed as the penalty, yielding a larger $f_2$. Otherwise, job $j$ is completed before the due date and $\max(C_j - D_j, 0)$ equals 0. $f_3$ represents the total advance time of jobs that finish before the due date. To save energy, machines will switch to the sleeping mode when idle and if a new procedure arrives, they will turn to the working mode. $f_4$ calculates the production cost generated by each machine during working and sleeping modes, where *wpc* and *spc* represent the unit production cost of the working and sleeping modes, respectively. $f_5$ measures the machine loss. $count_m$ counts the time that the machine $m$ changes from the sleeping mode to the working mode. A larger $count_m$ will cause more machine loss on $m$. It is worth mentioning that the machines are in the shutdown mode in the beginning, that is, the *count* will not increase when machines turn to the working mode at the first time.

The previous work mainly considers time-related objectives, but the cost is also a deeply concerned objective since it directly influences the profit of companies. Therefore, to better meet the QoS requirements, our proposed MaJSSP model considers both time-related and cost-related objectives. Specifically, $f_1$, $f_2$, and $f_3$ describe the work efficiency of industrial factories from the aspect of time, including the completion time, total tardiness, and advance time. $f_4$ and $f_5$ estimate the operational costs of industrial factories in terms of production cost and maintenance cost. Note that higher machine loss will induce higher maintenance costs. Obviously, smaller values of these five objectives are preferred. In this article, we develop an MaOEA, called MPMOGA, to efficiently solve our proposed MaJSSP.

## III. Proposed Algorithm

To solve the MaJSSP, we propose an MPMOGA approach that incorporates the MPMO framework with GA. The MPMO framework helps the algorithms optimize each objective sufficiently, which has been validated in [22]–[25]. In MPMOGA, an archive is constructed to store elite solutions found during the evolutionary process. Besides, the AST is proposed in the crossover operation to achieve coevolution among all the populations. To further improve the quality of elite solutions in

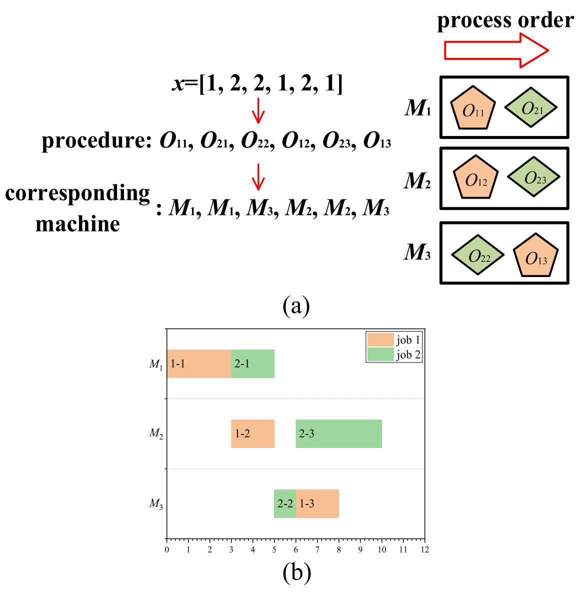
process order
x=[1, 2, 2, 1, 2, 1]
procedure: O11, O21, O22, O12, O23, O13
corresponding machine : M1, M1, M3, M2, M2, M3
M1
M2
M3
(a)
job 1
job 2
1-1
2-1
1-2
2-3
2-2
1-3
(b)

Fig. 2. Encoding example. (a) Feasible schedule $x$. (b) Gantt chart of $x$.

the archive, the AUS is developed. The detailed procedures of MPMOGA are described as follows.

## A. Solution Encoding

MPMOGA adopts an operation-based encoding method [38] that uses a sequence of $J\times M$ dimensions to represent a schedule. Variable $j \in [1, J]$ identifies $J$ different jobs and each variable $j$ appears $M$ times in a solution to denote the $M$ procedures of job $j$. Fig. 2 further illustrates the operation-based encoding method through an encoding example.

Given an instance with $J = 2$, $M = 3$, job 1 = [(1, 3), (2, 2), (3, 2)], and job 2 = [(1, 2), (3, 1), (2, 4)]. Each job has three procedures and each procedure is labeled with ($m$, $pt$), which means that this procedure must be processed on machine $m$ for $pt$ processing time. For example, in job 1, the first procedure (i.e., $O_{11}$) needs to be processed on machine 1 with a processing time of 3, the second procedure (i.e., $O_{12}$) needs to be processed on machine 2 with a processing time of 2, and the third procedure (i.e., $O_{13}$) needs to be processed on machine 3 with a processing time of 2.

A feasible schedule $x$ in this instance can be encoded as [1, 2, 2, 1, 2, 1] to represent a process order of procedures in each machine, as shown in Fig. 2(a). The genes encoded with 1 represent the procedures of job 1, while the genes encoded with 2 represent the procedures of job 2. Since 1 appears for the first time in the first dimension of $x$, it means the first procedure of job 1 (i.e., $O_{11}$). In the fourth dimension, 1 appears for the second time which means the second procedure of job 1 (i.e., $O_{12}$). Accordingly, the six dimensions in $x$ represent procedure $O_{11}$, $O_{21}$, $O_{22}$, $O_{12}$, $O_{23}$, and $O_{13}$, respectively. The corresponding machines for processing these six procedures are $M_1$, $M_1$, $M_3$, $M_2$, $M_2$, and $M_3$, which can be obtained by the data in job 1 and job 2. Hence, the process order of each machine is determined by $x$, that is, $M_1$ first processes $O_{11}$ and then processes $O_{21}$, $M_2$ first processes $O_{12}$ and then processes $O_{23}$, and $M_3$ first processes $O_{22}$ and then processes $O_{13}$.

Based on the process order and processing time, the Gantt chart of $x$ is obtained as shown in Fig. 2(b). Note that the blocks with different colors denote different jobs, the label $j$-$i$ in each block represents the procedure $O_{ji}$, and the length of each block indicates the corresponding processing time. The Gantt chart is explained as follows. First, $M_1$ processes $O_{11}$ from the beginning to time 3, as all jobs are released at time 0. Second, due to Constraints 2 and 3 in the MaJSSP model, $M_1$ processes $O_{21}$ from time 3 to time 5. Third, although $M_3$ is idle, $O_{22}$ cannot be processed immediately due to Constraint 4. Specifically, $O_{21}$ ($O_{22}$'s prior-order procedure) is finished in time 5; thus, $M_3$ starts to process $O_{22}$ in time 5. Other procedures are processed according to the above rules and, finally, the Gantt chart is obtained.

**Algorithm 1** *AST* (*Pop*, *A*, $Pr_c$)

```
Input: $Pop_k$ (current populations, $k \in [1, 5]$), $A$ (archive),
       $Pr_c$ (crossover probability).
1: For $k$=1 to 5
2:   $Pop'_k \leftarrow \emptyset$;
3:   For $i$=1 to $|Pop_k|$
4:     $p_1 \leftarrow Pop_k^i$;
5:     Randomly select one elite solution $es$ in $A$;
6:     $p_2 \leftarrow es$;
7:     $r=rand(0, 1)$;
8:     If $r \leq Pr_c$
9:       $(c_1, c_2)$=POX$(p_1, p_2)$;
10:      $x = \arg\min_{x\in\{p_1,c_1,c_2\}} \{f_k(x)\}$;
11:      $Pop'_k \leftarrow Pop'_k \bigcup \{x\}$;
12:    Else
13:      $x = \arg\min_{x\in\{p_1,p_2\}} \{f_k(x)\}$ ;
14:      $Pop'_k \leftarrow Pop'_k \bigcup \{x\}$;
15:    End If
16:  End For
17:End For
Output: $Pop'_k$ (populations generated by crossover, $k \in$
        $[1, 5]$)
```

## B. MPMO Framework

Under the MPMO framework, five populations are employed to optimize five objectives, respectively. The performance of the solution in each population is evaluated by its corresponding optimization objective rather than the Pareto dominance rule, as the condition of the Pareto dominance is hard to satisfy in MaOPs and, thus, may slow down the evolutionary process [39]. Note that the GA-based operators (i.e., crossover operator and mutation operator) are adopted as the optimizer in each population to generate offspring since GA is suitable for solving discrete optimization problems. More details of the GA-based operators are illustrated in Section III-C. As the MPMO framework can obtain a deep search on each objective, the AST is proposed to avoid each population fully focusing on one objective and lead the coevolution among the different populations, which are described in detail in Section III-D.

## C. Crossover and Mutation Operators

The precedence operation crossover (POX) proposed in [40] can retain the good characteristics of parents and always

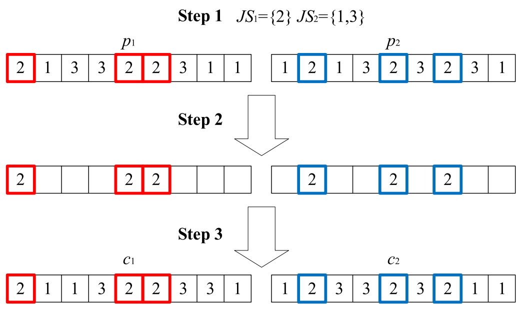

Fig. 3. Illustration of the POX operator.

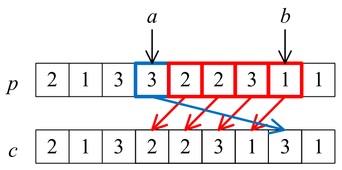

Fig. 4. Illustration of the IM operator.

generate legal offspring in solving JSSPs [41]. Hence, the POX is adopted in the MPMOGA to operate genetic crossover. The procedures of POX are illustrated as follows.

*Step 1:* The job set is divided into two nonempty subsets $JS_1$ and $JS_2$ randomly.

*Step 2:* The genes of parent $p_1$ that belong to $JS_1$ are copied to offspring $c_1$ with the same positions, and the genes of $p_2$ that belong to $JS_1$ are copied to offspring $c_2$ also with the same positions.

*Step 3:* The genes of parent $p_1$ that belong to $JS_2$ are copied to offspring $c_2$ with the same order, and the genes of $p_2$ that belong to $JS_2$ are copied to offspring $c_1$ also with the same order.

Take Fig. 3 as an example. Given $J = 3$, $M = 3$, parent $p_1 = [2, 1, 3, 3, 2, 2, 3, 1, 1]$, and parent $p_2 = [1, 2, 1, 3, 2, 3, 2, 3, 1]$. Considering that two nonempty subsets are randomly generated as $JS_1 = \{2\}$ and $JS_2 = \{1, 3\}$. The genes of $p_1$ that are marked with red (i.e., the 1st, 5th, and 6th dimensions) belong to $JS_1$, so that they are copied to $c_1$ and their locations remain the same as in $p_1$. Similarly, the genes of $p_2$ that are marked with blue (i.e., the 2nd, 5th, and 7th dimensions) are copied to $c_2$ with the same positions. Afterward, the remaining genes in $p_1$ that belong to $JS_2$ are copied to $c_2$ in the order of [1, 3, 3, 3, 1, 1], while the genes in $p_2$ that belong to $JS_2$ are filled in $c_1$ in the order of [1, 1, 3, 3, 3, 1]. Finally, two offspring $c_1 = [2, 1, 1, 3, 2, 2, 3, 3, 1]$ and $c_2 = [1, 2, 3, 3, 2, 3, 2, 1, 1]$ are generated.

The insertion mutation (IM) operator [42] is adopted in the MPMOGA. First, two dimensions $a$ and $b$ ($a \le b$) are randomly selected from $[1, J \times M]$. Then, the genes from dimension $a + 1$ to dimension $b$ are all shifted one place left, and the gene in $a$ is moved to dimension $b$. Note that no gene will change if $a$ equals $b$. Fig. 4 shows an example of the IM operator, where $a = 4$ and $b = 8$. Specifically, the genes marked with red (i.e., the 5th, 6th, 7th, and 8th dimensions) of $p$ are all shifted one place left, and the gene marked with blue is moved to location $b$ (i.e., the 8th dimension). Then, a mutated solution $c$ is generated.

**Algorithm 2** *AUS* (*Pop*, *A*, *NA*)

```
Input: $Pop_k$ (current populations, $k \in [1, 5]$), $A$ (archive),
       $NA$ (maximum size of the archive).
1:  $S \leftarrow \emptyset$;
2:  For $k$=1 to 5
3:      $S \leftarrow S \bigcup Pop_k$;
4:  End For
5:  $S \leftarrow S \bigcup A$;
6:  For each solution $x$ in $A$
7:      $x' \leftarrow x$;
        //local perturbation
8:      Execute IM operator on $x'$;
9:      $S \leftarrow S \bigcup \{x'\}$;
10: End For
    //duplication elimination
11: Execute duplication elimination on $S$ to get $S'$;
12: $A \leftarrow \emptyset$;
13: If $|S'|$ does not exceed $NA$
14:     $A \leftarrow S'$;
15: Else
16:     Select $NA$ solutions with smaller nondomination rank
        and larger crowding distance from $S'$ into $A$;
17: End If
Output: $A$ (updated archive)
```

## D. AST

The archive, denoted by $A$, is constructed in MPMOGA to store the elite solutions found during the evolutionary process. Note that $A$ is initialized as all the randomly generated solutions in five populations without duplication. After that, the AST is developed to exchange information among populations and assist the crossover operator, which helps achieve efficient population coevolution.

The AST is designed to assist the crossover operator, that is, the POX in Algorithm 1. Specifically, for the $k$th ($k \in [1, 5]$) population, each solution in population $k$ is selected as $p_1$ and a solution in $A$ is randomly chosen as $p_2$ (lines 4–6). In each crossover operation, a random value $r$ ranging in [0, 1] is generated. If $r$ is less than or equals the crossover probability $Pr_c$, the POX operator is conducted on $p_1$ and $p_2$ to generate offspring $c_1$ and $c_2$, and the one with the best performance on the $k$th objective among $p_1$, $c_1$, and $c_2$ is preserved to generate a new population (lines 8–11). Otherwise, the better one between $p_1$ and $p_2$ is preserved (lines 12–14). Therefore, the AST in MPMOGA not only can retain the good characteristics in its corresponding optimization objective but also can utilize the information on the other four objectives effectively.

## E. AUS

In MPMOGA, the AUS is proposed to update $A$ according to the newly generated populations, so as to generate solutions with better performance. The pseudocode of the AUS is shown in Algorithm 2. As the number of nondominated solutions increases during the evolution, it is recommended to define a threshold for the archive's size [43]. Therefore, in MPMOGA, the maximal size of the archive is set to $NA$. For

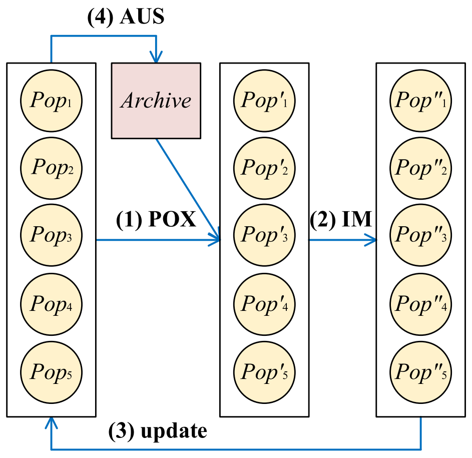

Fig. 5. Mechanism of MPMOGA.

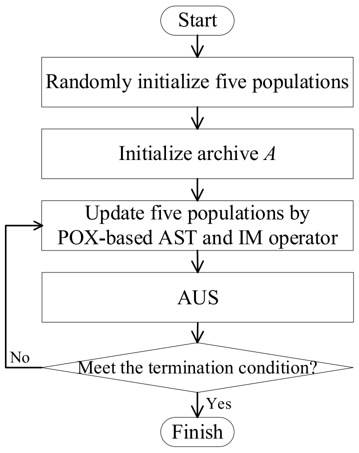
Start
Randomly initialize five populations
Initialize archive A
Update five populations by POX-based AST and IM operator
AUS
Meet the termination condition?
No
Yes
Finish

Fig. 6. Flowchart of MPMOGA.

each generation, all solutions in the current five populations are added to an empty set $S$ (lines 2–4), all elite solutions in $A$ are also added to $S$ (line 5), and each elite solution in $A$ is perturbed and then added to $S$ (lines 6–10). The perturbation operation on $A$ is implemented based on the IM operator introduced in Section III-C. Based on local perturbation, the AUS mainly improves the convergence of elite solutions thus enhancing the exploitation ability of the MPMOGA. Since the solutions gathered from the five populations and the archive may have duplications, the duplication elimination operation not only can save the storage but also can help preserve more diverse solutions in the archive. If the size of $S$' is less than or equals the predefined size $NA$, all the solutions in $S$' are copied to $A$, otherwise, the nondominated sorting approach and diversity preservation method in [13] are executed on $S$' to select $NA$ elite solutions into $A$ (lines 12–17). The AUS can enhance the quality of the solutions in the archive, so as to help the AST better lead the coevolutionary process among populations. Besides, the detailed experimental analysis of the AUS is presented in Section IV-D.

## F. Complete Procedures of MPMOGA

The mechanism of our proposed MPMOGA is shown in Fig. 5. Initially, five populations are initialized to optimize the five optimization objectives, respectively. In each generation, first, the current population $Pop_k$ ($k \in [1, 5]$) cooperates with $A$ to implement POX for generating crossover offspring. The one with the best performance on the $k$th objective is preserved to generate the new population $Pop'_k$. Second, each solution $p'_i$ in $Pop'_k$ may generate a mutated solution $c'_i$ with a mutation probability $Pr_m$ based on the IM operator. The solution with smaller fitness on its corresponding optimization objective between $p'_i$ and $c'_i$ is preserved to the mutated population $Pop''_k$. Third, $Pop_k$ is updated by $Pop''_k$. Fourth, the AUS is executed based on the newly updated populations $Pop_k$.

In summary, the MPMO framework helps each population optimize its corresponding objective sufficiently. To avoid each population only concentrating on its corresponding optimization objective and performing poorly on the other objectives, the AST is developed to let populations share information through the archive. In addition, the AUS aims to improve the quality of the elite solutions, so as to better lead the evolution of all the populations. Therefore, the MPMOGA mechanism can approximate the true PF of MaJSSP in a cooperative coevolutionary way.

The flowchart of MPMOGA is shown in Fig. 6. First, five populations are randomly initialized by the operation-based encoding method with equal size. Then $A$ is initialized as all solutions in five populations. During the iteration process, five populations focus on optimizing five objectives, respectively. The POX-based AST and IM operators are executed to update each population, and then $A$ is updated by AUS. Repeat the evolutionary process until the termination condition is met. The elite solutions in $A$ are the final output of the MPMOGA.

## G. Computational Complexity of One Generation of MPMOGA

Given the population size (i.e., the number of solutions in five populations) $NP$, the dimension of each solution $D$, and the maximum size of archive $NA$. Note that $NA$ is set as the same as $NP$ in MPMOGA. Typically, the computational complexity of one generation of MPMOGA is determined by the genetic operators to generate well-performed offspring and the AUS to enhance the quality of the elite solutions.

When generating the offspring, the crossover operation which generates $NP$ crossover offspring with $D$ dimensions requires $O(NP \times D)$ computations. To select solutions with the best performance, the evaluation of the newly generated solutions is needed, which consumes $O(NP)$ computations. Besides, the complexity of the mutation operation is $O(NP \times D)$ and the complexity of solution evaluation for the mutated solutions is $O(NP)$. Therefore, the overall computational complexity of generating offspring by the POX-based AST and IM operator is $O(2NP \times D + 2NP)$.

In the AUS, first, the local perturbation and solution evaluation require $O(NA \times D + NA)$ computations. Second, the duplication elimination operation requires $O(|S|^2)$ computations due to the pairwise comparison. Note that the maximum size of $S$ is $NP + 2NA$ and $NA$ is equal to $NP$; thus, $O(|S|^2)$

can be reduced as $O(NP^2)$. Third, the complexity of the nondominated sorting approach and diversity preservation method is $O(NP^2)$ according to [13]. Hence, the overall computational complexity of the AUS is $O(NA \times D + NA + 2NP^2)$.

To sum up, the overall complexity of MPMOGA in genetic operators and AUS is $O(3NP \times D + 3NP + 2NP^2)$, where $NA$ is replaced by $NP$. After simplifying, the computational complexity of one generation of MPMOGA is $O(NP^2 + NP \times D)$.

## IV. Experimental Results

### A. Experimental Design

Two widely used test sets (i.e., FT test set [32] and LA test set [33]) are adopted in experiments. Specifically, three typical test instances of the FT test set (i.e., FT06, FT10, and FT20) and all 40 test instances of the LA test set (i.e., LA01–LA40) are selected to evaluate the algorithm as they have different dimension sizes vary from 36 (i.e., $J = 6$ and $M = 6$) to 300 (i.e., $J = 30$ and $M = 10$). According to [16], the due date $D_j$ of job $j$ is calculated as

$$D_j = t \times \sum_{i=1}^{M} pt_{ij} \tag{9}$$

where $t$ is a coefficient which is set to 1.8 and $pt_{ij}$ represents the processing time required for the $i$th procedure of job $j$. The unit production cost of the working mode $wpc$ and the unit production cost of the sleeping mode $spc$ in (7) are set to 4.0 and 2.0, respectively.

The experimental results of our proposed algorithm are compared to three effective MaOEAs and a GGA [10] proposed for JSSPs. Specifically, three MaOEAs are: 1) the Pareto based nondominated sorting GA III (NSGA-III) [44]; 2) the decomposition-based multiobjective EA with distance-based update strategy (MOEA/D-DU) [45]; and 3) the indicator-based stochastic ranking algorithm (SRA) [46]. Since MaJSSP is a discrete optimization problem and the crossover and mutation operators of these three MaOEAs are originally designed to solve continuous problems, the crossover and mutation operators of the three MaOEAs are adjusted to POX and IM, respectively, to make them capable to solve MaJSSP. Note that the other techniques are still the same as in their corresponding reference. Besides, the GGA is adopted for comparison without any modification.

In MPMOGA, crossover probability $Pr_c$ is set to 0.9; mutation probability $Pr_m$ is set to 0.1; population size $NP$ is set to 210; and the maximum size of archive $NA$ is also set to $NP$ (i.e., 210). Note that each population has 210/5 solutions. For the competitor algorithms, the settings of $Pr_c$, $Pr_m$, and $NP$ are the same as those in MPMOGA, except that the $Pr_c$ of GGA is set to 1.0 as suggested by the authors. Besides, the settings of other parameters are consistent with their original algorithms.

The termination condition for all the tested algorithms is the maximum running time. For the test instances whose dimension sizes are smaller than or equal to 100 (i.e., $J \times M \le 100$), the maximum running time is 10 s. For the test instances with dimension sizes in the range of (100, 200], the maximum running time is 15 s. For the test instances with dimension sizes larger than 200, the maximum running time is 20 s. The algorithms tested in this article are all implemented with C++ and experiments are performed on a PC with Intel Core i5-7400 CPU 3.00 GHz and 8.00-GB RAM. For a fair comparison, all algorithms conduct 20 independent runs.

The $C$-metric [47], invert generational distance (IGD) [48], and hypervolume (HV) [47] are used as performance metrics to evaluate all algorithms. $C(U, V)$ is calculated by (10), which compares the dominance relationship between two solution sets $U$ and $V$. Specifically, $C(U, V)$ represents the percentage that the solutions in $U$ dominate or are at least equal to the solutions in $V$. $C(U, V)$ will equal 1 if each solution in $V$ is dominated by or equal to at least one solution in $U$. If the solutions in $V$ are not dominated by or equal to any solution in $U$, then $C(U, V)$ will be 0. Note that the sum of $C(U, V)$ and $C(V, U)$ does not necessarily equal 1. $C(U, V) > C(V, U)$ means that the overall quality of the solutions in $U$ is better than that in $V$. In the experiments, all the nondominated solutions obtained in 20 runs are collected to compute $C$-metric on each test instance

$$C(U, V) = \frac{|\{v \in V | \exists u \in U : u \text{ dominates } v \text{ or equals } v\}|}{|V|}. \tag{10}$$

Before calculating IGD and HV, the PFs obtained by the algorithms and the true PF are normalized as

$$\text{norm\_}f_k(x) = \frac{f_k(x) - Z_{\min,k}}{Z_{\max,k} - Z_{\min,k} + 10^{-6}} \tag{11}$$

where $f_k(\cdot)$ represents the fitness value on the $k$th objective, norm_$f_k(\cdot)$ represents the normalized fitness value on the $k$th objective, and $Z_{\max,k}$ and $Z_{\min,k}$ represent the maximum value and minimum value of the true PF in the $k$th objective, respectively. Since the true PF is unknown in the real-world optimization problems, the nondominated solutions among the final solutions obtained by all algorithms in 20 runs are treated as the true PS, then the objectives of true PS are treated as the true PF.

After that, IGD is formulated as

$$IGD(T, R) = \frac{\sum_{v \in T} dist(v, R)}{|T|} \tag{12}$$

where $T$ represents the true PF and $R$ is the PF obtained by the algorithm. $dist(v, R)$ represents the minimum Euclidean distance between $v$ in $T$ and the objective vectors in $R$ in the normalized objective space. A smaller IGD indicates the better performance of the algorithm in terms of convergence and diversity.

Besides, HV measures the volume of the region bounded by a nadir point and the normalized PF obtained by the algorithm. A larger HV indicates a better performance of the algorithm. In this article, the widely used Monte-Carlo sampling method is adopted to calculate the HV in MaOPs. Specifically, the nadir point is set to (1.1, 1.1, 1.1, 1.1, 1.1), and 100 000 sampling points are randomly generated in $[0.0, 1.1]^5$.

Note that both the IGD and HV metrics are calculated in 20 runs independently, and both the average result and standard deviation result among 20 runs are presented.

TABLE I
*C*-METRIC RESULTS BETWEEN MPMOGA AND THE COMPETITOR ALGORITHMS ON 43 TEST INSTANCES

| Test Instance | NSGA-III | | MOEA/D-DU | | SRA | | GGA | |
|---|---|---|---|---|---|---|---|---|
| | *C*(MPMOGA, –) | *C*(–, MPMOGA) | *C*(MPMOGA, –) | *C*(–, MPMOGA) | *C*(MPMOGA, –) | *C*(–, MPMOGA) | *C*(MPMOGA, –) | *C*(–, MPMOGA) |
| FT06 | **1.0000** | 0.1818 | **0.9818** | 0.2857 | **0.9308** | 0.5909 | **1.0000** | 0.0779 |
| FT10 | **1.0000** | 0.0000 | **0.8621** | 0.0404 | **0.9268** | 0.0000 | **1.0000** | 0.0000 |
| FT20 | **0.8356** | 0.1268 | **0.5072** | 0.2958 | **0.8261** | 0.0986 | **1.0000** | 0.0000 |
| LA01 | **0.9508** | 0.0000 | **0.8673** | 0.0123 | **0.9364** | 0.0082 | **1.0000** | 0.0000 |
| LA02 | **0.9571** | 0.0081 | **0.8857** | 0.0121 | **0.9682** | 0.0000 | **1.0000** | 0.0000 |
| LA03 | **0.9905** | 0.0000 | **0.7143** | 0.1277 | **0.9841** | 0.0000 | **1.0000** | 0.0000 |
| LA04 | **0.9851** | 0.0000 | **0.9818** | 0.0000 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA05 | **1.0000** | 0.0000 | **0.8191** | 0.0591 | **0.9381** | 0.0364 | **1.0000** | 0.0000 |
| LA06 | **0.9000** | 0.0227 | **0.9580** | 0.0057 | **0.9588** | 0.0568 | **1.0000** | 0.0000 |
| LA07 | **0.9857** | 0.0000 | **0.6875** | 0.2339 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA08 | **0.9167** | 0.0511 | **0.8261** | 0.0284 | **0.9870** | 0.0000 | **1.0000** | 0.0000 |
| LA09 | **1.0000** | 0.0000 | **0.9365** | 0.0136 | **0.9892** | 0.0000 | **1.0000** | 0.0000 |
| LA10 | **0.8462** | 0.0179 | **0.8761** | 0.0323 | **0.9903** | 0.0000 | **1.0000** | 0.0000 |
| LA11 | **0.8302** | 0.0128 | **0.8852** | 0.0128 | **1.0000** | 0.0000 | **0.9000** | 0.0000 |
| LA12 | **0.7826** | 0.0690 | **0.8077** | 0.0460 | **0.9091** | 0.0287 | **1.0000** | 0.0000 |
| LA13 | **0.9032** | 0.0157 | **0.5417** | 0.3216 | **0.9677** | 0.0118 | **0.9091** | 0.0039 |
| LA14 | **0.9143** | 0.0070 | **0.7576** | 0.1469 | **0.9672** | 0.0140 | **0.8438** | 0.0420 |
| LA15 | **0.8116** | 0.0628 | **0.5455** | 0.5314 | **0.9390** | 0.0293 | **1.0000** | 0.0000 |
| LA16 | **0.9867** | 0.0000 | **0.7213** | 0.0791 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA17 | **0.9730** | 0.0000 | **0.7358** | 0.0902 | **1.0000** | 0.0000 | **0.9615** | 0.0000 |
| LA18 | **1.0000** | 0.0000 | **0.7179** | 0.0854 | **1.0000** | 0.0000 | **0.9333** | 0.0000 |
| LA19 | **0.9333** | 0.0000 | **0.6800** | 0.1967 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA20 | **1.0000** | 0.0000 | **0.8780** | 0.0385 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA21 | **1.0000** | 0.0000 | **0.8689** | 0.0061 | **1.0000** | 0.0000 | **0.9429** | 0.0000 |
| LA22 | **0.9483** | 0.0202 | **0.9079** | 0.0000 | **0.9880** | 0.0000 | **0.9643** | 0.0000 |
| LA23 | **1.0000** | 0.0000 | **0.9298** | 0.0000 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA24 | **0.9796** | 0.0000 | **0.9512** | 0.0123 | **1.0000** | 0.0000 | **0.9091** | 0.0000 |
| LA25 | **0.9375** | 0.0000 | **0.3750** | 0.2564 | **1.0000** | 0.0000 | **0.9630** | 0.0000 |
| LA26 | **0.8704** | 0.1681 | **0.5385** | 0.0973 | **1.0000** | 0.0000 | **0.9615** | 0.0000 |
| LA27 | **0.7308** | 0.0473 | **0.8800** | 0.0473 | **1.0000** | 0.0000 | **0.7368** | 0.1622 |
| LA28 | **0.7188** | 0.0198 | **0.7800** | 0.0891 | **0.9516** | 0.0297 | **0.7838** | 0.0495 |
| LA29 | **0.8158** | 0.0556 | **1.0000** | 0.0000 | **1.0000** | 0.0000 | **0.8966** | 0.0185 |
| LA30 | **0.8276** | 0.0441 | **0.8431** | 0.0221 | **1.0000** | 0.0000 | **0.9655** | 0.0074 |
| LA31 | **0.3684** | 0.0930 | **0.9643** | 0.2093 | **1.0000** | 0.0000 | **0.2857** | 0.1047 |
| LA32 | **0.9333** | 0.0000 | **0.6000** | 0.1569 | **0.9643** | 0.0000 | **0.5000** | 0.1569 |
| LA33 | **0.7586** | 0.0169 | **0.5000** | 0.4492 | **0.8571** | 0.0000 | **0.6000** | 0.1949 |
| LA34 | **0.4865** | 0.3621 | **0.7391** | 0.0000 | **1.0000** | 0.0000 | 0.2500 | **0.3793** |
| LA35 | **0.8000** | 0.1071 | **0.6667** | 0.0952 | **1.0000** | 0.0000 | **0.7857** | 0.0000 |
| LA36 | **0.9000** | 0.0288 | **0.8868** | 0.0288 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA37 | **0.6818** | 0.1747 | **0.8810** | 0.0843 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA38 | **0.9756** | 0.0000 | **0.6333** | 0.2188 | **1.0000** | 0.0000 | **0.8235** | 0.0563 |
| LA39 | **0.9016** | 0.0351 | **0.9333** | 0.0000 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| LA40 | **0.9153** | 0.0000 | **0.8333** | 0.0887 | **1.0000** | 0.0000 | **1.0000** | 0.0000 |
| >/< | 43/0 | | 43/0 | | 43/0 | | 42/1 | |

On each test instance, the Wilcoxon rank-sum test with a significant level of 0.05 is performed to detect whether there is a significant difference between the IGD (or HV) results of the two algorithms. The symbols "+," "≈," and "−" denote our proposed MPMOGA approach performs significantly better than, not significantly different from, and significantly worse than the competitor algorithms, respectively.

## *B. Experimental Results With Competitor Algorithms*

The solutions obtained by MPMOGA are compared to competitor algorithms from three aspects. The first aspect is the *C*-metric, which compares the dominance relationship of two solution sets obtained by two algorithms. The second aspect is the IGD and HV metrics, which can comprehensively evaluate the algorithm's ability in balancing convergence and diversity. The third aspect is the best objectives obtained during evolution which reveals the capability of the algorithm to optimize each objective.

*1) C-Metric:* Table I shows the comparisons between MPMOGA and four competitor algorithms on *C*-metric. The better one between $C(U, V)$ and $C(V, U)$ is **bolded**. In the last row of Table I, we count and present the numbers of test instances that the *C*(MPMOGA, –) value is larger and smaller than the *C*(–, MPMOGA) value, respectively. According to Table I, *C*(MPMOGA, –) values are larger than *C*(–, MPMOGA) values on all 43 test instances compared to three MaOEAs. In addition, *C*(MPMOGA, GGA) is significantly better than *C*(GGA, MPMOGA) on 42 test instances while slightly worse than *C*(GGA, MPMOGA) on the remaining 1 test instance. The above experimental results show that

TABLE II
IGD RESULTS BETWEEN MPMOGA AND THE COMPETITOR ALGORITHMS ON 43 TEST INSTANCES

| Test Instance | MPMOGA | NSGA-III | MOEA/D-DU | SRA | GGA |
|---|---|---|---|---|---|
| FT06 | **0.1195 (0.0161)** | 0.2342 (0.0181) + | 0.2016 (0.0183) + | 0.1516 (0.0151) + | 0.2765 (0.0174) + |
| FT10 | **0.2121 (0.0334)** | 0.4002 (0.0545) + | 0.3663 (0.0461) + | 0.3922 (0.0444) + | 0.5911 (0.0555) + |
| FT20 | **0.2121 (0.0245)** | 0.3247 (0.0451) + | 0.3288 (0.0570) + | 0.3108 (0.0501) + | 0.6005 (0.0589) + |
| LA01 | **0.1689 (0.0172)** | 0.3464 (0.0394) + | 0.2723 (0.0317) + | 0.2723 (0.0334) + | 0.4234 (0.0305) + |
| LA02 | **0.1841 (0.0195)** | 0.3681 (0.0354) + | 0.3086 (0.0376) + | 0.2832 (0.0328) + | 0.5087 (0.0497) + |
| LA03 | **0.1925 (0.0253)** | 0.3875 (0.0475) + | 0.3024 (0.0441) + | 0.3376 (0.0385) + | 0.6259 (0.0710) + |
| LA04 | **0.1654 (0.0239)** | 0.3531 (0.0584) + | 0.3259 (0.0594) + | 0.3263 (0.0427) + | 0.6161 (0.0624) + |
| LA05 | **0.1454 (0.0105)** | 0.2940 (0.0343) + | 0.2589 (0.0442) + | 0.2333 (0.0155) + | 0.3564 (0.0266) + |
| LA06 | **0.1667 (0.0155)** | 0.3417 (0.0373) + | 0.2889 (0.0426) + | 0.2890 (0.0270) + | 0.4463 (0.0364) + |
| LA07 | **0.1562 (0.0125)** | 0.3082 (0.0420) + | 0.2697 (0.0436) + | 0.3061 (0.0381) + | 0.4934 (0.0502) + |
| LA08 | **0.1729 (0.0272)** | 0.3357 (0.0390) + | 0.2996 (0.0472) + | 0.3106 (0.0348) + | 0.4382 (0.0390) + |
| LA09 | **0.1520 (0.0140)** | 0.3249 (0.0475) + | 0.2956 (0.0406) + | 0.3162 (0.0505) + | 0.3571 (0.0299) + |
| LA10 | **0.1321 (0.0137)** | 0.2348 (0.0206) + | 0.2195 (0.0237) + | 0.2319 (0.0185) + | 0.3134 (0.0340) + |
| LA11 | **0.1636 (0.0258)** | 0.3209 (0.0450) + | 0.2963 (0.0450) + | 0.3267 (0.0410) + | 0.3866 (0.0334) + |
| LA12 | **0.1754 (0.0275)** | 0.2927 (0.0346) + | 0.2728 (0.0478) + | 0.3185 (0.0429) + | 0.4179 (0.0440) + |
| LA13 | **0.1658 (0.0189)** | 0.3064 (0.0316) + | 0.2864 (0.0410) + | 0.3285 (0.0444) + | 0.4033 (0.0328) + |
| LA14 | **0.1825 (0.0348)** | 0.3147 (0.0267) + | 0.2826 (0.0502) + | 0.3159 (0.0387) + | 0.3790 (0.0521) + |
| LA15 | **0.1936 (0.0203)** | 0.3200 (0.0307) + | 0.3013 (0.0349) + | 0.3240 (0.0401) + | 0.4321 (0.0246) + |
| LA16 | **0.1938 (0.0232)** | 0.3783 (0.0493) + | 0.3102 (0.0379) + | 0.3553 (0.0383) + | 0.4271 (0.0376) + |
| LA17 | **0.1925 (0.0221)** | 0.4687 (0.0791) + | 0.3091 (0.0585) + | 0.5124 (0.0697) + | 0.3915 (0.0639) + |
| LA18 | **0.2330 (0.0605)** | 0.5678 (0.1000) + | 0.3622 (0.1125) + | 0.6717 (0.0986) + | 0.4770 (0.0791) + |
| LA19 | **0.3069 (0.0776)** | 0.9216 (0.2299) + | 0.5777 (0.2249) + | 1.1043 (0.2180) + | 0.6780 (0.1274) + |
| LA20 | **0.2597 (0.0414)** | 0.5190 (0.0504) + | 0.4164 (0.0644) + | 0.5011 (0.0575) + | 0.5001 (0.0499) + |
| LA21 | **0.1708 (0.0183)** | 0.4426 (0.0497) + | 0.3434 (0.0616) + | 0.4243 (0.0509) + | 0.4197 (0.0469) + |
| LA22 | **0.1914 (0.0262)** | 0.4428 (0.0606) + | 0.3502 (0.0453) + | 0.4457 (0.0466) + | 0.5005 (0.0415) + |
| LA23 | **0.2405 (0.0455)** | 0.4195 (0.0507) + | 0.3900 (0.0591) + | 0.4667 (0.0572) + | 0.4630 (0.0461) + |
| LA24 | **0.2250 (0.0397)** | 0.3672 (0.0378) + | 0.3676 (0.0439) + | 0.4489 (0.0399) + | 0.4728 (0.0421) + |
| LA25 | **0.2154 (0.0438)** | 0.3709 (0.0487) + | 0.3596 (0.0562) + | 0.4322 (0.0416) + | 0.4525 (0.0342) + |
| LA26 | **0.2433 (0.0550)** | 0.3651 (0.0513) + | 0.3974 (0.0760) + | 0.4981 (0.0611) + | 0.5162 (0.0577) + |
| LA27 | **0.1930 (0.0349)** | 0.3082 (0.0546) + | 0.3141 (0.0514) + | 0.3684 (0.0457) + | 0.3860 (0.0525) + |
| LA28 | **0.1911 (0.0389)** | 0.3085 (0.0429) + | 0.2891 (0.0416) + | 0.3412 (0.0357) + | 0.3453 (0.0433) + |
| LA29 | **0.2190 (0.0460)** | 0.4084 (0.0588) + | 0.3490 (0.0526) + | 0.4239 (0.0454) + | 0.4219 (0.0490) + |
| LA30 | **0.2032 (0.0345)** | 0.3517 (0.0450) + | 0.3411 (0.0519) + | 0.3797 (0.0523) + | 0.4265 (0.0245) + |
| LA31 | **0.4860 (0.1918)** | 0.5729 (0.1529) ≈ | 0.6466 (0.1252) + | 0.5613 (0.0873) ≈ | 0.6102 (0.0876) ≈ |
| LA32 | **0.4330 (0.1558)** | 0.5352 (0.0699) ≈ | 0.4850 (0.1361) ≈ | 0.4925 (0.0695) ≈ | 0.5305 (0.0815) + |
| LA33 | **0.3361 (0.1063)** | 0.4210 (0.0634) + | 0.4481 (0.1639) + | 0.4152 (0.0416) + | 0.5394 (0.1097) + |
| LA34 | **0.4767 (0.1183)** | 0.4987 (0.1030) ≈ | 0.6180 (0.1627) + | 0.5271 (0.0705) ≈ | 0.5190 (0.0894) ≈ |
| LA35 | **0.4133 (0.1168)** | 0.5035 (0.0490) + | 0.6928 (0.2385) + | 0.5725 (0.0585) + | 0.5509 (0.0425) + |
| LA36 | **0.1892 (0.0313)** | 0.3047 (0.0320) + | 0.3225 (0.0444) + | 0.4125 (0.0441) + | 0.4047 (0.0383) + |
| LA37 | **0.1758 (0.0215)** | 0.3356 (0.0444) + | 0.3239 (0.0669) + | 0.4275 (0.0679) + | 0.5033 (0.0551) + |
| LA38 | **0.1825 (0.0346)** | 0.4202 (0.0713) + | 0.3350 (0.0662) + | 0.4521 (0.0673) + | 0.4048 (0.0530) + |
| LA39 | **0.1992 (0.0370)** | 0.3645 (0.0418) + | 0.4031 (0.0655) + | 0.5052 (0.0502) + | 0.4495 (0.0508) + |
| LA40 | **0.1989 (0.0298)** | 0.3182 (0.0309) + | 0.3439 (0.0507) + | 0.4143 (0.0466) + | 0.4716 (0.0455) + |
| | +/≈/– | 40/3/0 | 42/1/0 | 40/3/0 | 41/2/0 |

MPMOGA significantly outperforms the four competitor algorithms on *C*-metric. Specifically, most of the *C*(MPMOGA, –) values are larger than 0.7 which indicates that most of the solutions obtained by the competitor algorithms are dominated by the solutions of MPMOGA. Besides, for SRA and GGA, the *C*(MPMOGA, –) values are equal to 1 on more than half of the test instances, which means each solution obtained by these two algorithms is dominated by or at least equal to one solution in MPMOGA. In summary, the *C*-metric results illustrate that MPMOGA can obtain solutions with good quality.

*2) IGD and HV Metrics:* The IGD results of all algorithms are listed in Table II. Note that the average results and standard deviation results are presented in the form of *avg* (*std*). The best results on each test instance are **bolded**. It can be seen from Table II that MPMOGA performs the best on all test instances. According to the results of the Wilcoxon rank-sum test, MPMOGA is significantly better than NSGA-III, MOEA/D-DU, SRA, and GGA on 40, 42, 40, and 41 test instances, and is not significantly different from them on the remaining 3, 1, 3, and 2 test instances.

Table S.I in the supplementary material shows the HV results of all tested algorithms. We can see that MPMOGA performs significantly better than NSGA-III, MOEA/D-DU, SRA, and GGA on 43, 42, 43, and 39 test instances, respectively. GGA performs better than MPMOGA on only one test instance while the other three competitor algorithms cannot perform better than MPMOGA on any test instance. Note that on the LA34 instance, MPMOGA obtains the best average HV, but due to the large standard deviation, the Wilcoxon rank-sum test shows that MPMOGA performs worse than GGA.

Specifically, NSGA-III and MOEA/D-DU have shown their superiorities in solving MaOPs in the benchmark. The shapes of true PF in MaOP benchmark are uniform and regular so that the uniformly generated reference points in these two

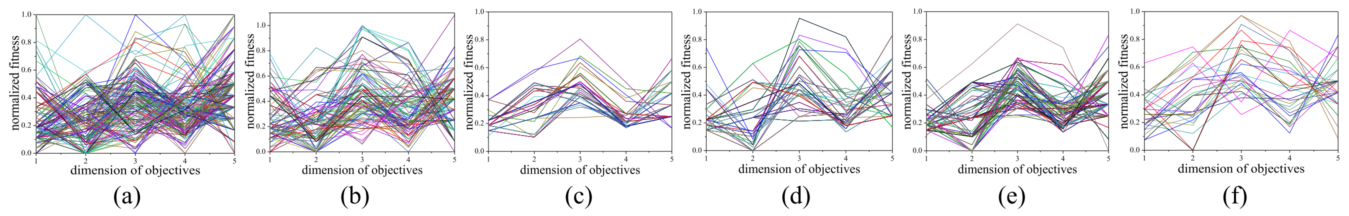

Fig. 7. On FT06 test instance, (a) normalized true PF, (b) normalized PF obtained by MPMOGA in an independent run, (c) normalized PF obtained by NSGA-III in an independent run, (d) normalized PF obtained by MOEA/D-DU in an independent run, (e) normalized PF obtained by SRA in an independent run, and (f) normalized PF obtained by GGA in an independent run.

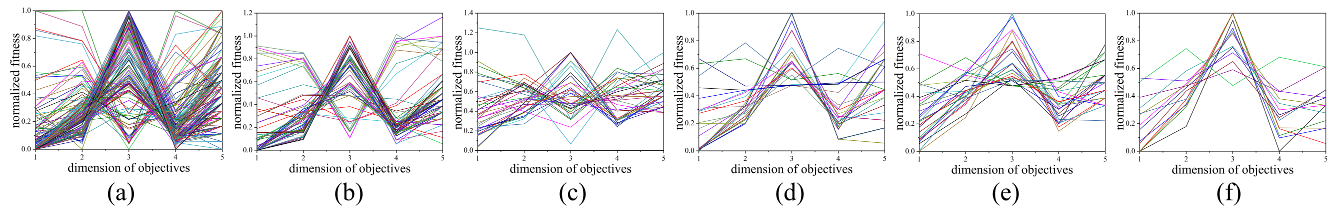

Fig. 8. On LA14 test instance, (a) normalized true PF, (b) normalized PF obtained by MPMOGA in an independent run, (c) normalized PF obtained by NSGA-III in an independent run, (d) normalized PF obtained by MOEA/D-DU in an independent run, (e) normalized PF obtained by SRA in an independent run, and (f) normalized PF obtained by GGA in an independent run.

algorithms can provide efficient guidance. However, the shapes of true PF in MaJSSPs may not be uniform and regular, thus resulting in the unsatisfactory performance of NSGA-III and MOEA/D-DU in IGD and HV metrics. Besides, the objectives in MaJSSP have different scales and ranges, which may lead to inaccurate calculation of convergence indicator, diversity indicator, and density estimator, thus degrading the performance of SRA and GGA. Compared to these four competitor algorithms, the advantages of MPMOGA are as follows. First, the MPMOGA approach is a reference-point-free algorithm. Instead of using reference points to guide the evolution, MPMOGA employs multiple populations and an archive to efficiently solve MaJSSP in a cooperative coevolutionary way. Second, MPMOGA will not be affected by the different objective scales. Under the MPMO framework, the solutions in each population can be distinguished by their corresponding optimization objective. To sum up, MPMOGA uses five populations to optimize five different objectives and constructs an archive to achieve coevolution among the populations, which is beneficial to solve MaJSSP.

To further compare the performance of all tested algorithms, the normalized true PF and normalized PFs obtained by different algorithms on the test instances are plotted. Figs. 7–10 present the normalized PFs obtained by all tested algorithms on FT06, LA14, LA26, and LA38 test instances, where subfigure (a) is the normalized true PF.

Based on the normalization, the distribution of true PF can be maintained in the range of [0, 1]. As for the PFs obtained by the tested algorithms, if the normalized objective of PF is larger than 1, it means the algorithm does not fully converge to true PF. The higher degree of the normalized objective greater than 1 indicates the worse convergence of the algorithm. Besides, a more similar distribution between PF obtained by the algorithm and true PF represents the better diversity of the algorithm.

TABLE III
COMPARISONS BETWEEN MPMOGA AND THE COMPETITOR ALGORITHMS IN TERMS OF OPTIMIZING EACH OBJECTIVE

| Metrics | NSGA-III | MOEA/D-DU | SRA | GGA | MPMOGA |
|---|---|---|---|---|---|
| Total times of obtaining the best objectives | 20 | 45 | 12 | 31 | **164** |
| Average rank | 2.79 | 2.33 | 3.14 | 2.60 | **1.12** |
| Total times of the first rank | 0 | 4 | 0 | 3 | **38** |

Concretely, on the FT06 and LA14 test instances, although the normalized objectives of four competitor algorithms do not exceed 1, these four competitor algorithms only converge to a subregion of the true PF. On the contrary, MPMOGA achieves the results closest to the true PF. Due to the large problem sizes of the FT26 and FT38 test instances, all tested algorithms fail to converge to the true PF. Among them, MPMOGA can better approximate the true PF while the other algorithms perform poorly on both convergence and diversity. In addition, MPMOGA successfully obtains the minimum or near-minimum value on most objectives on the FT26 and FT38 test instances, while the other algorithms fail. The above experimental results show the effectiveness of MPMOGA in solving MaJSSP.

*3) Capability of Optimizing Each Objective:* Table III shows the overall performance of the algorithms in terms of optimizing each objective. To save space, the detailed results of the best objectives obtained by the algorithms are listed in Table S.II in the supplementary material. In Table III, the "Total times of obtaining the best objectives" represents the sum of the times that an algorithm obtains the best objectives on 43 test instances, the "Average rank" is the average of the rank of an algorithm on 43 test instances, and the "Total times of the first rank" records the number of the first ranks obtained by an algorithm on 43 test instances.

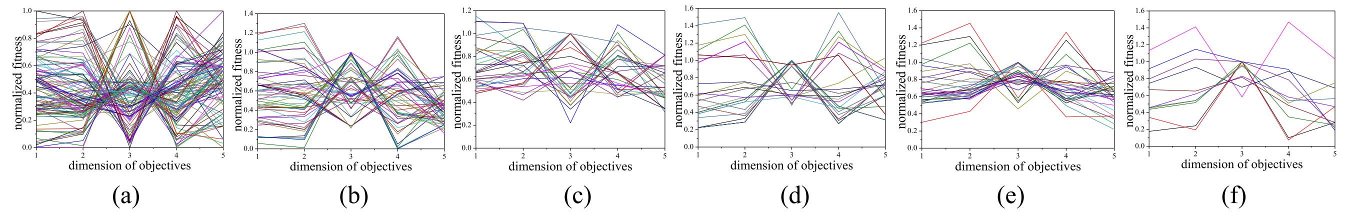

Fig. 9. On LA26 test instance, (a) normalized true PF, (b) normalized PF obtained by MPMOGA in an independent run, (c) normalized PF obtained by NSGA-III in an independent run, (d) normalized PF obtained by MOEA/D-DU in an independent run, (e) normalized PF obtained by SRA in an independent run, and (f) normalized PF obtained by GGA in an independent run.

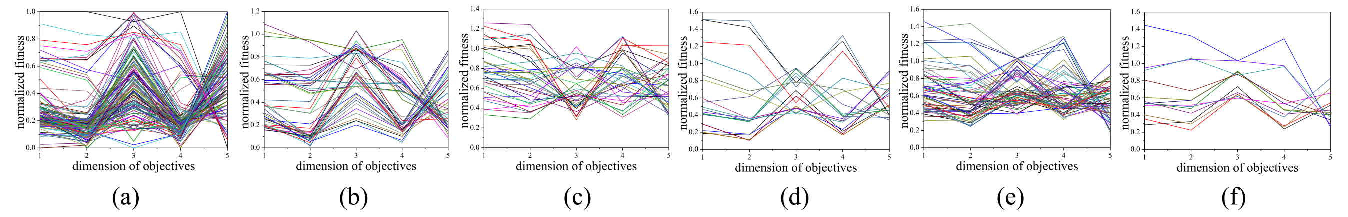

Fig. 10. On LA38 test instance, (a) normalized true PF, (b) normalized PF obtained by MPMOGA in an independent run, (c) normalized PF obtained by NSGA-III in an independent run, (d) normalized PF obtained by MOEA/D-DU in an independent run, (e) normalized PF obtained by SRA in an independent run, and (f) normalized PF obtained by GGA in an independent run.

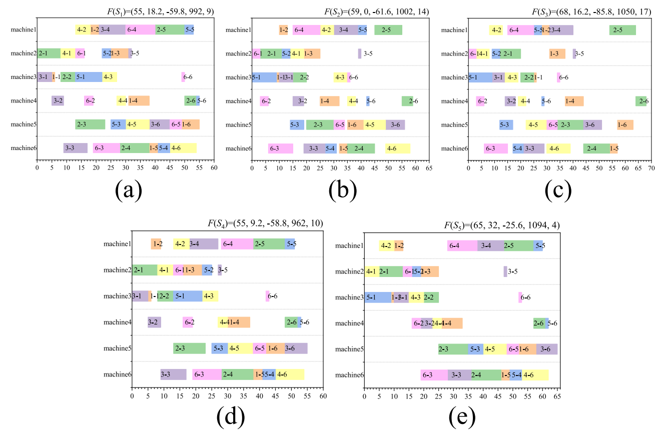

Fig. 11. Gantt charts of five schedules obtained by MPMOGA on FT06 test instance. (a) Schedule $S_1$. (b) Schedule $S_2$. (c) Schedule $S_3$. (d) Schedule $S_4$. (e) Schedule $S_5$.

As shown in Table III, the MPMOGA obtains the smallest objective values for 164 times in total, which is significantly better than the four competitor algorithms. In addition, MPMOGA achieves the best average rank, that is, 1.12. Besides, MPMOGA ranks first on 38 out of all 43 test instances, which is significantly superior to other competitor algorithms. The experimental results shown in Table III and Table S.II in the supplementary material illustrate the great capability of MPMOGA to optimize each objective.

## C. Analysis of Schedules Obtained by MPMOGA

To further discuss the features of MPMOGA in solving MaJSSP, we select five schedules from the output of the MPMOGA for detailed analysis. The Gantt charts of the five schedules obtained by the MPMOGA approach on the FT06 test instance are plotted in Fig. 11. Note that FT06 is a test instance with $J = 6$ and $M = 6$. In the figure, blocks with different colors denote different jobs, the label *j*-*i* in each block represents the *i*th procedure of the job *j*, and the length of each block indicates the corresponding processing time. It can be seen from Fig. 11(a) that schedule $S_1$ is optimal on the first objective with a maximum completion time of 55 (i.e., the completion time of both job 1 and job 5). The schedule $S_2$ achieves the best performance on the second objective. According to (9), the due date for each job on FT06 can be calculated (i.e., $D_1 = 46.8, D_2 = 84.6, D_3 = 61.2, D_4 = 63, D_5 = 45$, and $D_6 = 54$), and Fig. 11(b) presents that the completion time of each job does not exceed the corresponding due date (i.e., $C_1 = 41, C_2 = 59, C_3 = 56, C_4 = 58, C_5 = 43$, and $C_6 = 36$); therefore, the total tardiness of $S_2$ is 0. As shown in Fig. 11(c), schedule $S_3$ obtains the best value on the third objective. The completion time of each job in $S_3$ is $C_1$=63, $C_2 = 68$, $C_3 = 51$, $C_4 = 39$, $C_5 = 29$, and $C_6 = 35$. In particular, job 2, job 3, job 4, job 5, and job 6 are finished 16.6, 10.2, 24, 16, and 19 time units before the due date, respectively, yielding the best advance time. The best value of the fourth objective is obtained by schedule $S_4$. Since the work time for each machine is fixed, the production cost is affected by the sum of the sleep time on each machine. The sleep time of all machines in $S_4$ (i.e., 87) is the smallest among the five schedule solutions, while the sleep time in the other four schedules is 102, 107, 131, and 153, respectively. Thus, schedule $S_4$ has the lowest production cost. The machine loss of schedule $S_5$ is only 4 that is optimal on the fifth objective. In conclusion, the five schedules obtained by MPMOGA obtain the minimum value on the five objectives, respectively. Each population in MPMOGA focuses on optimizing one objective while learning the information on the remaining four objectives from the other populations through the AST. Consequently, the obtained schedules are well-converged and well-distributed, which can meet the various demands of factories.

## D. Validation of Perturbation in Archive

To investigate the effectiveness of the perturbation (i.e., the IM operator) in the AUS, we design a variant algorithm

TABLE IV
*C*-METRIC RESULTS BETWEEN MPMOGA AND ITS VARIANT ALGORITHMS ON 43 TEST INSTANCES

| Test Instance | MPMOGA-np | | MPMOGA-sm | | Test Instance | MPMOGA-np | | MPMOGA-sm | |
|---|---|---|---|---|---|---|---|---|---|
| | *C*(MPMOGA, –) | *C*(–, MPMOGA) | *C*(MPMOGA, –) | *C*(–, MPMOGA) | | *C*(MPMOGA, –) | *C*(–, MPMOGA) | *C*(MPMOGA, –) | *C*(–, MPMOGA) |
| FT06 | **0.9709** | 0.4416 | **0.8768** | 0.7208 | LA20 | **0.8696** | 0.3462 | **1.0000** | 0.0000 |
| FT10 | **0.9359** | 0.0303 | **0.7517** | 0.0505 | LA21 | **0.7763** | 0.1768 | **0.9268** | 0.0854 |
| FT20 | **0.7143** | 0.4155 | 0.2727 | **0.5986** | LA22 | **0.7629** | 0.0404 | **0.9677** | 0.404 |
| LA01 | **0.6692** | 0.1270 | **0.4975** | 0.1516 | LA23 | **0.9394** | 0.0286 | **0.9785** | 0.0000 |
| LA02 | **0.7372** | 0.0769 | **0.8848** | 0.0486 | LA24 | **0.8431** | 0.1975 | **0.9762** | 0.0494 |
| LA03 | **0.8000** | 0.0426 | **0.9308** | 0.0284 | LA25 | **0.8846** | 0.0641 | **0.9861** | 0.0000 |
| LA04 | **0.7545** | 0.1460 | **0.8443** | 0.0354 | LA26 | **0.8500** | 0.0442 | **0.9459** | 0.0000 |
| LA05 | **0.8977** | 0.0000 | **0.7939** | 0.1136 | LA27 | **0.7500** | 0.2297 | **0.9146** | 0.0473 |
| LA06 | **0.7907** | 0.1591 | **0.8182** | 0.0455 | LA28 | **0.8471** | 0.0198 | **0.8125** | 0.0396 |
| LA07 | **0.7339** | 0.1210 | **0.9530** | 0.0040 | LA29 | **0.9701** | 0.0926 | **0.9416** | 0.0278 |
| LA08 | **0.7500** | 0.0455 | **0.9643** | 0.0398 | LA30 | **0.6066** | 0.3088 | **0.9712** | 0.0147 |
| LA09 | **0.8404** | 0.0340 | **0.9307** | 0.0204 | LA31 | **0.6216** | 0.2442 | **0.6897** | 0.0233 |
| LA10 | **0.6838** | 0.1434 | **0.8772** | 0.0287 | LA32 | **0.9091** | 0.0000 | **0.9636** | 0.0196 |
| LA11 | **0.6935** | 0.1859 | **0.7049** | 0.0449 | LA33 | **0.7755** | 0.1949 | **0.7059** | 0.2881 |
| LA12 | **0.7826** | 0.0747 | **0.8358** | 0.0172 | LA34 | **0.8750** | 0.1207 | **0.8929** | 0.0517 |
| LA13 | **0.8588** | 0.0745 | **0.8158** | 0.0510 | LA35 | **0.7442** | 0.1310 | **0.8269** | 0.0595 |
| LA14 | **0.8875** | 0.0420 | **0.6429** | 0.0909 | LA36 | **0.7500** | 0.0673 | **0.8657** | 0.2788 |
| LA15 | **0.6053** | 0.1632 | **0.7733** | 0.1297 | LA37 | **0.8354** | 0.0783 | **0.8596** | 0.1024 |
| LA16 | **0.6882** | 0.2203 | **0.8791** | 0.2712 | LA38 | **0.7344** | 0.0625 | **0.9186** | 0.0750 |
| LA17 | **0.6667** | 0.2707 | **0.9583** | 0.0075 | LA39 | **0.9674** | 0.0088 | **0.9800** | 0.0088 |
| LA18 | **0.5769** | 0.1585 | **0.8462** | 0.0488 | LA40 | **0.7347** | 0.2258 | **0.7476** | 0.1129 |
| LA19 | **0.4615** | 0.2295 | **0.8846** | 0.1803 | >/< | 43/0 | | 42/1 | |

TABLE V
IGD RESULTS BETWEEN MPMOGA AND ITS VARIANT ALGORITHMS ON 43 TEST INSTANCES

| Test Instance | MPMOGA | MPMOGA-np | MPMOGA-sm | Test Instance | MPMOGA | MPMOGA-np | MPMOGA-sm |
|---|---|---|---|---|---|---|---|
| FT06 | **0.1222 (0.0166)** | 0.1682 (0.0154) + | 0.1338 (0.0170) + | LA20 | **0.2621 (0.0475)** | 0.3097 (0.0357) + | 0.3782 (0.0509) + |
| FT10 | **0.2178 (0.0330)** | 0.2763 (0.0388) + | 0.2349 (0.0277) ≈ | LA21 | **0.2030 (0.0234)** | 0.2807 (0.0381) + | 0.3219 (0.0476) + |
| FT20 | **0.2035 (0.0212)** | 0.2520 (0.0253) + | 0.2195 (0.0312) ≈ | LA22 | **0.2190 (0.0226)** | 0.2709 (0.0284) + | 0.3058 (0.0249) + |
| LA01 | **0.1714 (0.0173)** | 0.2184 (0.0167) + | 0.1895 (0.0207) + | LA23 | **0.2386 (0.0439)** | 0.3183 (0.0338) + | 0.3648 (0.0357) + |
| LA02 | **0.1822 (0.0182)** | 0.2364 (0.0186) + | 0.2303 (0.0229) + | LA24 | **0.2442 (0.0431)** | 0.2894 (0.0449) + | 0.3823 (0.0462) + |
| LA03 | **0.1915 (0.0243)** | 0.2515 (0.0177) + | 0.2533 (0.0266) + | LA25 | **0.2082 (0.0400)** | 0.2549 (0.0291) + | 0.3141 (0.0380) + |
| LA04 | **0.1839 (0.0262)** | 0.2560 (0.0419) + | 0.2536 (0.0288) + | LA26 | **0.1939 (0.0547)** | 0.2671 (0.0368) + | 0.2982 (0.0449) + |
| LA05 | **0.1291 (0.0095)** | 0.1806 (0.0166) + | 0.1620 (0.0114) + | LA27 | **0.1953 (0.0353)** | 0.2599 (0.0324) + | 0.2823 (0.0386) + |
| LA06 | **0.1890 (0.0159)** | 0.2743 (0.0292) + | 0.2381 (0.0144) + | LA28 | **0.2118 (0.0409)** | 0.2796 (0.0286) + | 0.2904 (0.0438) + |
| LA07 | **0.1541 (0.0132)** | 0.2224 (0.0283) + | 0.2257 (0.0283) + | LA29 | **0.2001 (0.0439)** | 0.2885 (0.0404) + | 0.2870 (0.0589) + |
| LA08 | **0.1696 (0.0248)** | 0.2136 (0.0217) + | 0.2631 (0.0419) + | LA30 | **0.2136 (0.0415)** | 0.2706 (0.0449) + | 0.3140 (0.0427) + |
| LA09 | **0.1536 (0.0142)** | 0.2015 (0.0203) + | 0.2092 (0.0217) + | LA31 | **0.4889 (0.1902)** | 0.5173 (0.1424) ≈ | 0.5019 (0.1284) ≈ |
| LA10 | **0.1419 (0.0144)** | 0.1909 (0.0186) + | 0.1972 (0.0156) + | LA32 | **0.4242 (0.1510)** | 0.4354 (0.0893) ≈ | 0.4245 (0.1082) ≈ |
| LA11 | **0.1623 (0.0263)** | 0.2194 (0.0294) + | 0.2673 (0.0374) + | LA33 | **0.3662 (0.1566)** | 0.4518 (0.1181) ≈ | 0.3943 (0.0927) ≈ |
| LA12 | **0.1726 (0.0234)** | 0.2335 (0.0205) + | 0.2403 (0.0339) + | LA34 | **0.4370 (0.1167)** | 0.4991 (0.0397) + | 0.4989 (0.0635) + |
| LA13 | **0.1568 (0.0165)** | 0.2278 (0.0271) + | 0.2404 (0.0444) + | LA35 | **0.3846 (0.1107)** | 0.4239 (0.0522) ≈ | 0.4319 (0.0608) ≈ |
| LA14 | **0.1709 (0.0316)** | 0.2237 (0.0292) + | 0.2177 (0.0293) + | LA36 | **0.1861 (0.0265)** | 0.2432 (0.0376) + | 0.2424 (0.0371) + |
| LA15 | **0.1684 (0.0242)** | 0.2298 (0.0273) + | 0.2484 (0.0340) + | LA37 | **0.1795 (0.0210)** | 0.2573 (0.0329) + | 0.2547 (0.0301) + |
| LA16 | **0.2022 (0.0205)** | 0.2467 (0.0219) + | 0.2774 (0.0290) + | LA38 | **0.1831 (0.0323)** | 0.2483 (0.0369) + | 0.2598 (0.0264) + |
| LA17 | **0.1827 (0.0245)** | 0.2091 (0.0261) + | 0.2930 (0.0520) + | LA39 | **0.1996 (0.0366)** | 0.3125 (0.0458) + | 0.3208 (0.0619) + |
| LA18 | **0.2404 (0.0644)** | 0.2467 (0.0367) ≈ | 0.3912 (0.0827) + | LA40 | **0.2099 (0.0306)** | 0.2804 (0.0284) + | 0.2485 (0.0388) + |
| LA19 | **0.3364 (0.0778)** | 0.3932 (0.0874) + | 0.5758 (0.0986) + | +/≈/– | | 38/5/0 | 37/6/0 |

with no perturbation (called MPMOGA-np) and a variant algorithm with swap mutation (called MPMOGA-sm) for comparison in this section. Note that the swap mutation is a simple perturbation operation that is executed by randomly selecting two dimensions of a solution and exchanging their genes. The *C*-metric results, IGD results, and HV results between MPMOGA and its variant algorithms are listed in Tables IV and V and Table S.III in the supplementary material, respectively, where the best results are **bolded**.

According to Table IV, *C*(MPMOGA, MPMOGA-np) values are larger than *C*(MPMOGA-np, MPMOGA) values on all 43 test instances; *C*(MPMOGA, MPMOGA-sm) values are larger than *C*(MPMOGA-sm, MPMOGA) on 42 test instances, while worse than *C*(MPMOGA-sm, MPMOGA) on only one test instance. Specifically, most *C*(MPMOGA, –) values are close to 1 while most *C*(–, MPMOGA) values are close to 0, which shows that the IM operator in the AUS can efficiently improve the quality of solutions.

As for the IGD results, it can be seen from Table V that MPMOGA is superior to MPMOGA-np and MPMOGA-sm on 38 and 37 test instances and performs not significantly different from MPMOGA-np and MPMOGA-sm on the remaining five and six test instances, respectively.

From the HV results shown in Table S.III in the supplementary material, we can see that MPMOGA performs significantly better than MPMOGA-np and MPMOGA-sm on 40 and 39 test instances, respectively, and does not perform worse than these two variant algorithms on any test instance.

To sum up, the *C*-metric results, IGD results, and HV results show that the perturbation (i.e., the IM operator) in the AUS can enhance both the convergence and diversity of the solutions, thus improving the overall performance of MPMOGA.

## V. Conclusion

In this article, an MaJSSP model with five objectives covering the aspects of completion time, total tardiness, advance time, production cost, and machine loss was proposed. To efficiently solve this model, a novel algorithm based on the MPMO framework, called MPMOGA, was developed. Specifically, the MPMO framework aims to use different populations to optimize different objectives simultaneously. Besides, the AST is put forward within MPMOGA to guide the coevolution of all the populations, and the AUS is designed to enhance the quality of the elite solutions in the archive.

Two widely used test sets are adopted to evaluate the performance of the proposed MPMOGA. The experimental results show that our proposed MPMOGA performs well not only in terms of three widely used metrics (i.e., *C*-metric, IGD, and HV) but also in obtaining the best values on each objective. According to the experimental analysis, the MPMO framework is illustrated as an efficient way to deal with MaJSSP. In addition, we investigate the contribution of perturbation in the AUS for improving the overall performance of MPMOGA.

In the future, we will focus on other types of JSSPs (e.g., flexible JSSPs [49], [50], flow shop scheduling problems [51]–[53]), and extend the MPMO framework with other evolutionary computation [54] (e.g., particle swarm optimization [55]–[57], differential evolution [58]–[60], estimation of distribution algorithm [61], and gravitational search algorithm [62]) for efficiently solving them. Besides, the incorporation with distributed computing technique [63]–[65] or matrix-based technique [66] will be further studied to reduce the running time of the algorithm.

## References

[1] G.-Y. Zhu, X.-W. Ju, and W.-B. Zhang, "Multi-objective sequence optimization of PCB component assembly with GA based on the discrete Fréchet distance," *Int. J. Prod. Res.*, vol. 56, no. 11, pp. 4017–4034, Jun. 2016.

[2] T.-M. Choi, W.-K. Yeung, T. C. E. Cheng, and X. Yue, "Optimal scheduling, coordination, and the value of RFID technology in garment manufacturing supply chains," *IEEE Trans. Eng. Manag.*, vol. 65, no. 1, pp. 72–84, Feb. 2018.

[3] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Jul.–Sep. 2019.

[4] Y. N. Sotskov, V. S. Tanaev, and F. Werner, "Scheduling problems and mixed graph colorings," *Optimization*, vol. 51, no. 3, pp. 597–624, 2002.

[5] F. S. Al-Anzi, Y. N. Sotskov, A. Allahverdi, and G. V. Andreev, "Using mixed graph coloring to minimize total completion time in job shop scheduling," *Appl. Maths. Comput.*, vol. 182, no. 2, pp. 1137–1148, 2006.

[6] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under Industry 4.0," *J. Intell. Manuf.*, vol. 30, no. 4, pp. 1809–1830, 2019.

[7] H. Gao, S. Kwong, B. Fan, and R. Wang, "A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2044–2054, Nov. 2014.

[8] B. Peng, Z. P. Lü, and T. C. E. Cheng, "A tabu search/path relinking algorithm to solve the job shop scheduling problem," *Comput. Oper. Res.*, vol. 53, pp. 154–164, Jan. 2015.

[9] M. S. Viana, O. M. Junior, and R. C. Contreras, "A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem," *Sensors*, vol. 20, no. 18, p. 5440, 2020.

[10] G. May, B. Stahl, M. Taisch, and V. Prabhu, "Multi-objective genetic algorithm for energy-efficient job shop scheduling," *Int. J. Prod. Res.*, vol. 53, no. 23, pp. 7071–7089, 2015.

[11] X. Li, C. Lu, L. Gao, S. Xiao, and L. Wen, "An effective multiobjective algorithm for energy-efficient scheduling in a real-life welding shop," *IEEE Trans. Ind. Informat.*, vol. 14, no. 12, pp. 5400–5409, Dec. 2018.

[12] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 193–208, Apr. 2014.

[13] 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.

[14] E. Ziztler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control*, 2002, pp. 95–100.

[15] Q. Meng, L. Zhang, and Y. Fan, "Research on multi-objective job shop scheduling with dual particle swarm algorithm based on greedy strategy," *Wireless Pers. Commun.*, vol. 103, no. 1, pp. 255–274, 2018.

[16] A. Masood, Y. Mei, G. Chen, and M. Zhang, "Many-objective genetic programming for job-shop scheduling," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 2016, pp. 209–216.

[17] A. Masood, Y. Mei, G. Chen, and M. Zhang, "A PSO-based reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling," in *Proc. Aust. Conf. Artif. Life Comput. Intell.*, 2017, pp. 326–338.

[18] A. Masood, G. Chen, Y. Mei, H. Al-Sahaf, and M. Zhang, "A fitness-based selection method for Pareto local search for many-objective job shop scheduling," in *Proc. IEEE Congr. Evol. Comput.*, Glasgow, U.K., 2020, pp. 1–8.

[19] H. Mokhtari, R. B. Kazemzadeh, and A. Salmasnia, "Time-cost tradeoff analysis in project management: An ant system approach," *IEEE Trans. Eng. Manag.*, vol. 58, no. 1, pp. 36–43, Feb. 2011.

[20] M.-Y. Cheng and D.-H. Tran, "Two-phase differential evolution for the multiobjective optimization of time–cost tradeoffs in resource-constrained construction projects," *IEEE Trans. Eng. Manag.*, vol. 61, no. 3, pp. 450–461, Aug. 2014.

[21] D. Lei, L. Gao, and Y. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Trans. Eng. Manag.*, vol. 65, no. 2, pp. 330–340, May 2018.

[22] Z.-H. Zhan, J. Li, J. Cao, J. Zhang, H. S.-H. Chung, and Y.-H. Shi, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, 2013.

[23] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[24] S.-Z. Zhou, Z.-H. Zhan, Z.-G. Chen, S. Kwong, and J. Zhang, "A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 8, 2020, doi: 10.1109/TITS.2020.2994779.

[25] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[26] J.-Y. Li, Z.-H. Zhan, H. Wang, and J. Zhang, "Data-driven evolutionary algorithm with perturbation-based ensemble surrogates," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 3925–3937, Aug. 2021.

[27] X. Xia *et al.*, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[28] X. Zhang, K.-J. Du, Z.-H. Zhan, S. Kwong, T.-L. Gu, and J. Zhang, "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4454–4468, Oct. 2020.

[29] H. Zhao *et al.*, "Local binary pattern-based adaptive differential evolution for multimodal optimization problems," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3343–3357, Jul. 2020.

[30] T. Mao, A.-S. Mihăiţă, F. Chen, and H. L. Vu, "Boosted genetic algorithm using machine learning for traffic control optimization," *IEEE Trans. Intell. Transp. Syst.*, early access, Apr. 19, 2021, doi: 10.1109/TITS.2021.3066958.

[31] Y.-F. Ge *et al.*, "Distributed memetic algorithm for outsourced database fragmentation," *IEEE Trans. Cybern.*, early access, Nov. 4, 2020, doi: 10.1109/TCYB.2020.3027962.

[32] J. F. Muth and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, vol. 3. Englewood Cliffs, NJ, USA: Prentice-Hall, 1963, pp. 225–251.

[33] S. Lawrence, *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*, Dept. Ind. Admin., Carnegie-Mellon Univ., Pittsburgh, PA, USA, 1984.

[34] Q. Lin *et al.*, "A clustering-based evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 391–405, Jun. 2019.

[35] Q. Zhu *et al.*, "An elite gene guided reproduction operator for many-objective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 765–778, Feb. 2021.

[36] S. C. Liu, Z. H. Zhan, K. C. Tan, and J. Zhang, "A multi-objective framework for many-objective optimization," *IEEE Trans. Cybern.*, early access, doi: 10.1109/TCYB.2021.3082200.

[37] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 233–246, Jan. 2020.

[38] M. Gen, "Solving job-shop scheduling problem using genetic algorithm," in *Proc. Int. Conf. Comput. Ind. Eng.*, 1994, pp. 576–579.

[39] Z. N. He, G. G. Yen, and J. Zhang, "Fuzzy-based Pareto optimality for many-objective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 269–285, Apr. 2014.

[40] C. Zhang, P. Li, Y. Rao, and S. Li, "A new hybrid GA/SA algorithm for the job shop scheduling problem," in *Proc. Eur. Conf. Evol. Comput. Comb. Optim.*, 2005, pp. 246–259.

[41] J. Yan and X. Wu, "A genetic based hyper-heuristic algorithm for the job shop scheduling problem," in *Proc. 7th Int. Conf. Intell. Hum. Mach. Syst. Cybern.*, Hangzhou, China, 2015, pp. 161–164.

[42] B. Akay and X. Yao, "Recent advances in evolutionary algorithms for job shop scheduling," in *Automated Scheduling and Planning*, vol. 505. Berlin, Germany: Springer, 2013, pp. 191–224.

[43] N. Al Moubayed, A. Petrovski, and J. McCall, "D2MOPSO: 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.

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

[45] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 180–198, Apr. 2016.

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

[47] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[48] Y. A. Sun, G. G. Yen, and Z. Yi, "IGD indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 173–187, Apr. 2019.

[49] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019.

[50] J.-Q. Li *et al.*, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Trans. Autom. Sci. Eng.*, early access, Mar. 17, 2021, doi: 10.1109/TASE.2021.3062979.

[51] X. Li and S. Ma, "Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times," *IEEE Trans. Eng. Manag.*, vol. 64, no. 2, pp. 149–165, May 2017.

[52] J.-Q. Li *et al.*, "Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2425–2439, Jun. 2020.

[53] Y.-Z. Li, Q.-K. Pan, K.-Z. Gao, M. F. Tasgetiren, B. Zhang, and J.-Q. Li, "A green scheduling algorithm for the distributed flowshop problem," *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107526.

[54] Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, to be published, doi: 10.1007/s10462-021-10042-y.

[55] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[56] J.-R. Jian, Z.-G. Chen, Z.-H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021.

[57] Q. Lin *et al.*, "Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 32–46, Feb. 2018.

[58] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 6, pp. 3954–3967, Jun. 2021.

[59] Z.-H. Zhan, Z.-J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.

[60] Z.-G. Chen, Z.-H. Zhan, H. Wang, and J. Zhang, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 708–719, Aug. 2020.

[61] Z.-G. Chen, Y. Lin, Y.-J. Gong, Z.-H. Zhan, and J. Zhang, "Maximizing lifetime of range-adjustable wireless sensor networks: A neighborhood-based estimation of distribution algorithm," *IEEE Trans. Cybern.*, early access, Apr. 1, 2020, doi: 10.1109/TCYB.2020.2977858.

[62] Y. Wang, S. Gao, M. Zhou, and Y. Yu, "A multi-layered gravitational search algorithm for function optimization and real-world problems," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 1, pp. 94–109, Jan. 2021.

[63] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2715–2729, Jun. 2020.

[64] Z.-J. Wang, Z.-H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1175–1188, Mar. 2021.

[65] J.-Y. Li, Z.-H. Zhan, R.-D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation-level parallelism for evolutionary computation: A pipeline-based parallel particle swarm optimization," *IEEE Trans. Cybern.*, early access, Nov. 4, 2020, doi: 10.1109/TCYB.2020.3028070.

[66] Z.-H. Zhan *et al.*, "Matrix-based evolutionary computation," *IEEE Trans. Emerg. Topics Comput. Intell.* early access, Jan. 21, 2021, doi: 10.1109/TETCI.2020.3047410.

**Si-Chen Liu** (Student Member, IEEE) received the B.S. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2020, where she is currently pursuing the M.S. degree in computer science and technology with the School of Computer Science and Engineering.

Her research interests mainly include multiobjective optimization, many-objective optimization, and their applications in real-world problems.

**Zong-Gan Chen** (Member, IEEE) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016, and the Ph.D. degree in computer science and technology from the South China University of Technology, Guangzhou, in 2021.

He is currently a Lecturer with the Network Information Security Department, Guangdong Police College, Guangzhou. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world optimization problems.

**Zhi-Hui Zhan** (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from the Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation, swarm intelligence, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Early Career Award in 2021, the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018, and the Wu Wen-Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. His doctoral dissertation was awarded the IEEE CIS Outstanding Ph.D. dissertation and the China Computer Federation Outstanding Ph.D. dissertation. He is listed as one of the Highly Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Neurocomputing*, and *Memetic Computing*.

**Sang-Woon Jeon** (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Yonsei University, Seoul, South Korea, in 2003 and 2006, respectively, and the Ph.D. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011.

He has been an Associate Professor with the Department of Military Information Engineering (undergraduate school) and the Department of Electronics and Communication Engineering (graduate school), Hanyang University, Seoul, since 2017. From 2011 to 2013, he was a Postdoctoral Associate with the School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. From 2013 to 2017, he was an Assistant Professor with the Department of Information and Communication Engineering, Andong National University, Andong, South Korea. His research interests include network information theory, wireless communications, sensor networks, and their applications to the Internet of Things and big data.

Dr. Jeon was a recipient of the Haedong Young Scholar Award in 2017, which was sponsored by the Haedong Foundation and given by the Korea Institute of Communications and Information Science (KICS), the Best Paper Award of the KICS journals in 2016, the Best Paper Award of the IEEE International Conference on Communications in 2015, the Best Thesis Award from the Department of Electrical Engineering, KAIST, in 2012, the Best Paper Award of the KICS Summer Conference in 2010, and the Bronze Prize of the Samsung Humantech Paper Awards in 2009.

**Sam Kwong** (Fellow, IEEE) received the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

He is currently a Chair Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong is the President-Elect of the IEEE Systems, Man, and Cybernetics (SMC). He was also appointed as IEEE Distinguished Lecturer of the IEEE SMC Society in 2017. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was elevated to an IEEE Fellow for his contributions to optimization techniques for cybernetics and video coding in 2014.

**Jun Zhang** (Fellow, IEEE) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Korea Brain Pool Fellow Professor with Hanyang University, Seoul, South Korea, and a Visiting Professor with Chaoyang University of Technology, Taichung, Taiwan. His current research interests include computational intelligence, cloud computing, operations research, and power-electronic circuits. He has published more than 150 IEEE TRANSACTIONS papers in his research areas.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, The National Science Fund for Distinguished Young Scholars of China 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 CYBERNETICS and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.