IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 19, 2018, accepted December 12, 2018, date of publication December 24, 2018,
date of current version January 16, 2019.

*Digital Object Identifier 10.1109/ACCESS.2018.2889373*

# Multi-Objective Migrating Birds Optimization Algorithm for Stochastic Lot-Streaming Flow Shop Scheduling With Blocking

**YUYAN HAN[1], (Member, IEEE), JUN-QING LI[2], (Member, IEEE), DUNWEI GONG[3], (Member, IEEE), AND HONGYAN SANG[1]**

[1]School of Computer Science, Liaocheng University, Liaocheng 252059, China
[2]School of Information and Engineering, Shandong Normal University, Jinan 250014, China
[3]School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China

Corresponding authors: Yuyan Han (hanyuyan@lcu-cs.com) and Jun-Qing Li (lijunqing@lcu-cs.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61803192, Grant 61773192, Grant 61503170, Grant 61503220, Grant 61603169, Grant 61773246, Grant 71533001, Grant 61876075, and Grant 61573362 and in part by the Natural Science Foundation of Shandong Province under Grant ZR2017BF039.

**ABSTRACT** Blocking lot-streaming flow shop scheduling problem with the stochastic processing time has a wide range of applications in various industrial systems. However, this problem has not yet been well studied. In this paper, the above-mentioned problem is transformed into a determinate multi-objective optimization one using the Monte Carlo sampling method. A Multi-Objective Migrating Birds Optimization (MOMBO) algorithm is then proposed to solve the above-mentioned re-formulated multi-objective scheduling problem, in which the multiple-based PFE is proposed to yield the initial solutions with high quality, the information of the non-dominated solutions is learned and sampled to improve the global searching ability of MOMBO, and a reference-point-assisted local search method for multi-objective optimization is applied to further enhance the exploitation capability of MOMBO. To evaluate the performance of the MOMBO, several comparative experiments are executed on 180 test scheduling instances. The experimental results demonstrate that the MOMBO outperforms the compared algorithms in convergence and distributivity and has capacities to tackle the uncertainties.

**INDEX TERMS** Scheduling, multi-objective, blocking lot-streaming flow shop, stochastic processing time, migrating birds optimization.

## I. INTRODUCTION

Lot-Streaming Flow Shop (LSFS) problem is a typical scheduling problem with strong engineering background, which has important application in different industries including production process of solar cell modules and chemical industry [1]–[4]. Due to LSFS can split a job into several sublots, in which each sublot can be transferred to the downstream machine after it is completed in the current one, it can reduce the production cycle, accelerate the manufacturing process, thereby enhancing the production efficiency [4]. However, in most practical production process, no intermediate buffer between machines is used to store completed jobs, resulting in these jobs have to block in the current machine, until their following one is available for processing. Previous research has already been done to tackle a Blocking Flow Shop (BFS) scheduling problem [5], [6]. Similarly, in LSFS scheduling problems, each sublot will also be blocked due to no intermediate buffer exists. Thus, the blocking case in the production process encourages us to consider the blocking constraint to a LSFS scheduling problem, and formula the model of a Blocking LSFS (BLSFS) scheduling problem.

Many intelligent optimization algorithms have been proposed to solve the single and multi-objective problems in different research fields [7]–[18]. For LSFS scheduling problems, there include a novel Hybrid Multi-Objective Artificial Bee Colony (HDABC) [4], an Improved Migrating Birds Optimization (IMBO) [19], an effective Modified Migrating Birds Optimization (MMBO) [20], a Multi-Objective Evolutionary Algorithm (MOEA) [21], a chaos-induced Discrete Self Organizing Migrating Algorithm (DSOMA) [22], a Discrete Invasive Weed Optimization (DIWO) [23], an Improved Sheep Flock Heredity (ISFH) [24], a Genetic Algorithms(GA) [25], a Local-best Harmony Search (LHS) [26], an Improved Non-dominated

Sorting Genetic Algorithm II (INSGA-II) [27], a Estimation of Distribution Algorithms (EDA) [28], and so on.

There exist some difficulties when effectively solve BLSFS, such as a large number of constraints, high complexities, various disruptions and unforeseen events [29]. The uncertainties in BLSFS scheduling problems mainly include machine breakdowns, material shortage, the arrival of new jobs and changes in process time. Although a number of efforts have been made on solving single or multiple objectives LSFS scheduling problems, most of them do not take uncertainties into account. Thus, in this paper, we are motivated to solve the multi-objective BLSFS scheduling problem with stochastic process time, guarantying that some better schedules are relatively insensitive to unforeseen processing time. The twofold novelties of this paper are given in the following:

(1) To weaken the negative influence of stochastic process time on the makespan, mean and standard variance values of expected makespan are simultaneously optimized. The former aims to minimize the complete time, and the latter is designed to reduce the disturbance caused by stochastic process time. Thus, in this paper, we translate a stochastic scheduling problem into a traditional multi-objective BLSFS, which can better reflect real-world applications, and easier compare to scheduling problems in previous work.

(2) A multi-objective migrating birds optimization algorithm is employed to solve the above re-formulated multi-objective scheduling problem. Three contributions of the proposed MOMBO lie in: (a) a variable single-objective heuristic is proposed to initializing the population; (b) all of non-dominated solutions are taken advantage to generate the solutions with high quality; (c) a reference-point-assisted local search is adopted to enhance the exploitation capability of the algorithm.

The rest of this paper is organized as follows. After a brief introduction in Section II, the BLSFS scheduling problem with stochastic processing time is converted into a conventional multi-objective scheduling problem in Section III. Section IV states the basic MBO. Section V gives the proposed algorithm. Section VI lists and discusses the experimental results. Section VII concludes the paper.

## II. LITERATURE REVIEW

### A. FLOW SHOP AND JOB SHOP SCHEDULING PROBLEMS WITH UNCERTAINTIE

For deterministic scheduling problems, all the information of jobs and machines is assumed to be fixed and known in advance. However, in practical manufacturing process, some uncertainties such as uncertain processing time, machine breakdowns, arrival of new jobs, and so on [30], [31] usually occur, and their effects may be detrimental to the manufacturing outcomes [32]. For uncertain processing time, many scholars have proposed metaheuristics, i.e., stochastic, fuzzy, and interval programming algorithms [33]. If the parameters are initially described in terms of probability distributions, then the problem is named as the stochastic scheduling [34].

In stochastic approaches, we assume that the processing time obeys a known probability distribution, and under the circumstances the stochastic processing time can be converted into the deterministic counterpart. Since this approach is quite straightforward, increasing attention has been paid to tacking stochastic scheduling problems [35], [36]. For the flow shop scheduling with random processing times, Nagasawa *et al.* [37] considered to insert idle time into the schedule in order to reduce the likelihood of simultaneous operations. Lei proposed an efficient GA to solve the stochastic job shop scheduling problem with normal processing time. In this paper, some operations of normal processing times are defined to build the schedule and genetic operators are separated from the handling of random breakdown [38]. Fu *et al.* [39] addressed a two-agent stochastic flow shop deteriorating scheduling problem with multiple objectives. In this work, two populations are utilized to execute the global and local searches, and one archive is used to guide the computation resource allocation in the search process. Almeder and Hartl [40] adopted multiple scenarios to evaluate the objective of a stochastic flexible flow shop problem with limited buffer, in which the proposed acceptance criterion for the real-word case can lead to reduce runtimes while the solution quality still remains at a high level.

For the uncertain processing times in the flow shop scheduling problem, their upper and the lower bounds are often easy to know [21]. In our previous works, we utilize the upper and lower bounds of the process time to convert a multi-objective interval BLSFS scheduling problem into a conventional one. In this work, the objective interval is converted into a deterministic real value through dynamically weighting [21]. Ćwik and Józefczyk [41] adopted interval-valued processing times to represent the uncertain parameters used the lower bound instead of solving the internal deterministic flow-shop, and employed the maximum regret to evaluate the uncertainty. For the single machine scheduling problems with uncertain processing time, Allahverdi *et al.* [42] addressed some heuristics based on the polynomial time using the values of the upper and lower bounds of processing time. Feng *et al.* [43] designed a min-max-regret scheduling model to reflect the uncertain processing time, in which a worst-case scenario and heuristic methods are proposed to solve the above problem. One common idea in the above literature is that the upper and lower bounds of the interval processing time are taken advantage of so as to convert the interval values into deterministic values.

Monte Carlo (MC) is an analysis method that can simulate randomly uncertainties. It has great capabilities to forecast the uncertainties, and to offer more accurate solutions of the results generated. For the scheduling problem, MC simulation experiments can generate some sample values of processing time that are utilized to calculate the expected makespan [44]–[46]. A system in a stochastic context is more realistic than in a deterministic one and works remain to be done when concerning the stochastic version, Mokhtari and Salmasnia [44] executed MC simulation to solve the

parallel processor problem with stochastic processing time. Juan *et al.* [45] adopted MC and iterative local search methods to solve permutation flow shop scheduling problem with stochastic processing time. Asta *et al.* [46] considered a carefully-designed hybrid MC tree search, memetic algorithm, and hyper-heuristic methods to compute power of multicore machines of multi-project scheduling problem.

From the above literature of MC, we can see that the only expected makespan is considered, but the influence of the stochastic process time on makespan is not considered. Generally, to seek a stable and robust solution for the stochastic scheduling problem, we should concern the average and stability of uncertain objective function(s). In view of this, we can capture the uncertainty by calculating the mean and standard variance values of makespan, and then convert a stochastic scheduling problem into a determinate one.

### B. MIGRATING BIRDS OPTIMIZATION ALGORITHMS

Very recently, a new metaheuristic intelligence approach named the Migrating Birds Optimization (MBO) algorithm, which simulates the V flight formation of migrating birds, as the name implies, was presented by Duman *et al.* [47]. In the canonical MBO algorithm, all solutions are treated as birds aligned in a V formation, in which each solution can derive benefit from the solution in front of it. In fact, there are many formations that bird flocks use. However, two motives are described to explain the use of V formation. One is that the V formation is better than others in saving energy during flight. The other is that the V formation reflects a mechanism by which birds avoid collisions with each other and stay in visual contact. MBO is a neighborhood search method with good exploration, thus it has already been successfully applied to solve engineering optimization problems, continuous function optimization problems, and scheduling problems.

Duman proposed MBO to optimize the quadratic assignment problem, and the experimental results demonstrated that the proposed algorithm has better performance than the Simulated Annealing (SA), Tabu Search (TS), GA, Scatter Search (SS), Particle Swarm Optimization (PSO), and Differential Evolution (DE) [48]. Shen *et al.* [49] proposed a modified MBO to optimize the university course timetabling problem, in which an improved neigbourhood sharing mechanism and iterated local search are utilized with aim of generating promising solutions. For the several well-known continuous functions, Alkaya *et al.* [50] developed a novel neighbor generating function in MBO so as to generate good solutions in multidimensional continuous spaces.

For the scheduling problems, Tongur and Ülker [51] first applied the basic MBO algorithm to optimize the discrete flow shop sequencing problem. Following that Pan and Dong designed an improved MBO to minimize the total flow time of the hybrid flow shop scheduling. In this work, the authors presented a diversified method to initialize population with high quality, and constructed a mixed neighborhood based on insertion and pairwise exchange operators to generate promising neighboring solutions for the leader and the following birds [52]. Similarly, Niroomand *et al.* [53] also proposed a novel MBO algorithm to optimize the closed loop layout with exact distances in flexible manufacturing systems, which are different from IMBO considered by Pan and Dong. In MMBO, the authors employed crossover and mutation operators to yield the neighbor regeneration.

### C. MOTIVATIONS

For the problems considered in this paper, although there are various literatures about how to solve determinate single or multiple BLSFS scheduling problems, most of them do not take uncertainties into account, which cannot guarantee that an optimal schedule is relatively insensitive to stochastic processing time. The stochastic processing time will result in uncertain objective values of multi-objective BLSFS, which makes it non-trivial to determine the dominance relationship of between different solutions. Therefore, it is high time that efforts are dedicated to convert a BLSFS scheduling problem with stochastic processing time into a conventional multi-objective BLSFS scheduling problem whose criteria, such as mean and standard variance of makespan, are subject to uncertain processing time.

For MBO proposed in this paper, in addition to an easy implementing, a simple structure, and few mathematical requirements, MBO is a parallel processing, which can somehow be regarded an inherited to genetic algorithms and scatter search [52]. Thus, many intelligent algorithms, problem-dependent heuristics, and neighborhood search operators can be embedded in the above search framework to further enhance exploration and exploitation of the basic MBO. Moreover, the simulation experimental results of MBO have verified that it is appropriate and competitive for solving continuous and discrete optimization problems. To the best of our knowledge, MBO has not been applied to the multi-objective BLSFS scheduling problem with stochastic processing time.

Thus, with the above motivations, we proposed a MOMBO algorithm to solve the above reformulate BLSFS scheduling problem with stochastic process time. This paper extends the deterministic BLSFS to the stochastic one that is modeled as a multi-objective BLSFS, where the process time is obtained using MC simulation algorithm.

## III. CONVERSION OF BLSFS SCHEDULING PROBLEM WITH STOCHASTIC PROCESSING TIME

In a real-world manufacturing environment, the processing time of jobs might be highly uncertain due to quality problems, equipment downtime, tool wear, and operator availability [40]–[46]. If the processing time of a scheduling problem is stochastic, its value of corresponding objective will be difficultly computed, which thus increases the difficulty in selecting the superior solution. To overcome this, it is of necessity to convert the BLSFS scheduling problem with stochastic processing time into a conventional and determinate optimization problem.

MC simulation has the ability to consider the possible outcomes, which can help to obtain richer information for the flow shop scheduling problem with stochastic processing times. Thus, a set included $SP$ sample matrix of processing time is first obtained using MC simulation, i.e., $P = \{P^1, P^2, \ldots P^s, \ldots, P^{SP}\}$ in which

$$P^s = \begin{bmatrix} p^s_{\pi(1),1} & p^s_{\pi(1),2} & \cdots & p^s_{\pi(1),m} \\ p^s_{\pi(2),1} & p^s_{\pi(2),2} & \cdots & p^s_{\pi(2),m} \\ \cdots & \cdots & \cdots & \cdots \\ p^s_{\pi(n),1} & p^s_{\pi(n),2} & \cdots & p^s_{\pi(n),m} \end{bmatrix},$$

$s = 1, 2, \ldots, SP$, is a matrix included $n \times m$ processing time, the solution $\pi$, $\pi = (\pi(1), \pi(2), \ldots, \pi(j), \ldots, \pi(n))$, $\pi(j) \in \{1, 2, \ldots, n\}$, is a integer scheduling sequence and its dimension depends on the number of jobs. Variables $n$ and $m$ are the total number of jobs and machines, respectively. $p^s_{\pi(j),t}$ is the $s$-th the sample value of the processing time of $\pi(j)$ on machine $t$, $j = 1, 2, \ldots, n$, $t = 1, 2, \ldots, m$, and is assumed to obey a known probability distribution. Then, $SP$ complete times are calculated using the above $SP$ process times. We aim to seek some robust and stable scheduling strategies by minimizing both the mean and standard variance values of the expected makespan. Thus, this work focuses on BLSFS scheduling problem with mean and standard variance values of the expected makespan as two criterions. Except for the first 5 constraints listed in [21], the scheduling problem is subject to the sixth and seventh constraints.

(1) Each job can be split into several sublots, and each subloth as different processing time on different machines;

(2) A job can be processed on the current machine only when all sublots of its foregoing job are completed on the machine;

(3) At any time, each machine can process at most one sublot, and each sublot can be processed on at most one machine at the same time;

(4) All sublots of the same job should be continuously processed;

(5) Both the setup time and the sublot transportation time are included in processing time.

(6) A sublot must be blocked on the current machine before its downstream machine is not available;

(7) Only one type of uncertainties, i.e., stochastic processing times is considered in our paper.

The BLSFS scheduling problem can be formulated as follows:

$$\min_{\pi \in \Pi} f_1 = \frac{1}{SP} \sum_{s=1}^{SP} C_{\pi(n),m,w_{\pi(n)}}(P^s) \tag{1}$$

$$\min_{\pi \in \Pi} f_2 = \sqrt{\frac{\sum_{s=1}^{SP} (C_{\pi(n),m,w_{\pi(n)}}(P^s) - f_1)^2}{SP - 1}} \tag{2}$$

where $\Pi$ is the set of all the permutations, $f_1$ and $f_2$ are the mean and standard variance values of makespan, respectively, $w_{\pi(n)}$ is the number of sublots of $\pi(n)$, $C_{\pi(n),m,w_{\pi(n)}}(P^s)$ is the completion time of the last job on the last machine in which the sample values of the processing time are given in $P^s$. Its calculation processes are given as follows.

$s = 1, 2, \ldots, SP$,

$$\begin{cases} S_{\pi(1),1,1}(P^s) = 0 \\ C_{\pi(1),1,1}(P^s) = S_{\pi(1),1,1}(P^s) + p^s_{\pi(1),1} \end{cases} \tag{3}$$

$$\begin{cases} S_{\pi(1),t,1}(P^s) = C_{\pi(1),t-1,1}(P^s) \\ C_{\pi(1),t,1}(P^s) = S_{\pi(1),t,1}(P^s) + p^s_{\pi(1),t} \end{cases} \quad t = 2, 3, \ldots m \tag{4}$$

$$\begin{cases} S_{\pi(j),1,1}(P^s) = \max(C_{\pi(j-1),1,w_{\pi(j-1)}}(P^s), \\ \qquad\qquad S_{\pi(j-1),2,w_{\pi(j-1)}}(P^s)) \\ C_{\pi(j),1,1}(P^s) = S_{\pi(j),1,1}(P^s) + p^s_{\pi(j),1} \end{cases}$$

$j = 2, 3, \ldots, n$ (5)

$$\begin{cases} S_{\pi(j),t,1}(P^s) = \max(C_{\pi(j),t-1,1}(P^s), \\ \qquad\qquad S_{\pi(j-1),t+1,w_{\pi(j-1)}}(P^s)) \\ C_{\pi(j),t,1}(P^s) = S_{\pi(j),t,1}(P^s) + p^s_{\pi(j),t} \end{cases}$$

$j = 2, 3, \ldots, n$

$t = 2, 3, \ldots, m$ (6)

$$\begin{cases} S_{\pi(j),m,1}(P^s) = \max(C_{\pi(j),m-1,1}(P^s), \\ \qquad\qquad C_{\pi(j-1),m,w_{\pi(j-1)}}(P^s)) \\ C_{\pi(j),m,1}(P^s) = S_{\pi(j),m,1}(P^s) + p^s_{\pi(j),m} \end{cases}$$

$j = 2, 3, \ldots, n$ (7)

$$\begin{cases} S_{\pi(j),1,e}(P^s) = \max(C_{\pi(j),1,e-1}(P^s), \\ \qquad\qquad S_{\pi(j),2,e-1}(P^s)) \\ C_{\pi(j),1,e}(P^s) = S_{\pi(j),1,e}(P^s) + p^s_{\pi(j),1} \end{cases}$$

$j = 1, 2, \ldots, n$

$e = 2, 3, \ldots, w_{\pi(j)}$ (8)

$$\begin{cases} S_{\pi(j),t,e}(P^s) = \max(C_{\pi(j),t-1,e}(p^s_{\pi(j),t}), S_{\pi(j),t+1,e-1}(P^s)) \\ C_{\pi(j),t,e}(P^s) = S_{\pi(j),t,e}(P^s) + p^s_{\pi(j),t} \end{cases}$$

$j = 1, 2, \ldots, n$

$e = 2, 3, \ldots, w_{\pi(j)}$

$t = 2, 3, \ldots, m$ (9)

$$\begin{cases} S_{\pi(j),m,e}(P^s) = \max(C_{\pi(j),m-1,e}(P^s), C_{\pi(j),m,e-1}(P^s)) \\ C_{\pi(j),m,e}(P^s) = S_{\pi(j),m,e}(P^s) + p^s_{\pi(j),m} \end{cases}$$

$j = 1, 2, \ldots, n$

$e = 2, 3, \ldots, w_{\pi(j)}$ (10)

where $j$ and $t$ are the subscript, $w_{\pi(j)}$ is the total sublots number of job $\pi(j)$, and $e$ is its $e$th sublot. $S_{\pi(j),t,e}(P^s)$ and $C_{\pi(j),t,e}(P^s)$ are the start and the completion time of the $e$th sublot of $\pi(j)$ on machine $t$. To clearly illustrate the aforementioned conversion process, an example of computing $f_1$ and $f_2$ is given here. Suppose that there are 3 jobs whose numbers of associated sublots are 1, 2, and 1, respectively, processed on 3 machines, and $SP$=3.

First, 3 specific process time sets are listed using MC simulation as follows,

$$P^1 = \begin{bmatrix} 2 & 6 & 4 \\ 2 & 3 & 5 \\ 3 & 7 & 4 \end{bmatrix}, \quad P^2 = \begin{bmatrix} 4 & 7 & 5 \\ 6 & 7 & 6 \\ 5 & 4 & 2 \end{bmatrix},$$
$$P^3 = \begin{bmatrix} 4 & 5 & 5 \\ 6 & 5 & 4 \\ 4 & 6 & 4 \end{bmatrix}$$

Second, for the first determinate process time set, the corresponding determinate makespan value, $C_{\pi(n),m,w_{\pi(n)}}(P^1)$, is obtained using Eqs. (3-10), that is,

(1) $S_{\pi(1),1,1}(P^1) = 0$,
$C_{\pi(1),1,1}(P^1) = S_{\pi(1),1,1}(P^1) + p^1_{\pi(1),1} = 2$,
$S_{\pi(1),2,1}(P^1) = C_{\pi(1),1,1}(P^1) = 2$,
$C_{\pi(1),2,1}(P^1) = S_{\pi(1),2,1}(P^1) + p^1_{\pi(1),2} = 2 + 6 = 8$,
$S_{\pi(1),3,1}(P^1) = C_{\pi(1),2,1}(P^1) = 8$,
$C_{\pi(1),3,1}(P^1) = S_{\pi(1),3,1}(P^1) + p^1_{\pi(1),3} = 8 + 4 = 12$,

(2) $S_{\pi(2),1,1}(P^1) = \max(C_{\pi(1),1,1}(P^1), \quad S_{\pi(1),2,1}(P^1)) = 2$,
$C_{\pi(2),1,1}(P^1) = S_{\pi(2),1,1}(P^1) + p^1_{\pi(2),1} = 2 + 2 = 4$,
$S_{\pi(2),2,1}(P^1) = \max(C_{\pi(2),1,1}(P^1), \quad S_{\pi(1),3,1}(P^1)) = 8$,
$C_{\pi(2),2,1}(P^1) = S_{\pi(2),2,1}(P^1) + p^1_{\pi(2),2} = 8 + 3 = 11$,
$S_{\pi(2),3,1}(P^1) = \max(C_{\pi(2),2,1}(P^1), \quad C_{\pi(1),3,1}(P^1)) = 12$,
$C_{\pi(2),3,1}(P^1) = S_{\pi(2),3,1}(P^1) + p^1_{\pi(2),3} = 12 + 5 = 17$

(3) $S_{\pi(2),1,2}(P^1) = \max(C_{\pi(2),1,1}(P^1), \quad S_{\pi(2),2,1}(P^1)) = 8$,
$C_{\pi(2),1,2}(P^1) = S_{\pi(2),1,2}(P^1) + p^1_{\pi(2),1} = 8 + 2 = 10$,
$S_{\pi(2),2,2}(P^1) = \max(C_{\pi(2),1,2}(P^1), \quad S_{\pi(2),3,1}(P^1)) = 12$,
$C_{\pi(2),2,2}(P^1) = S_{\pi(2),2,2}(P^1) + p^1_{\pi(2),2} = 12 + 3 = 15$,
$S_{\pi(2),3,2}(P^1) = \max(C_{\pi(2),2,2}(P^1), C_{\pi(2),3,1}(P^1)) = 17$,
$C_{\pi(2),3,2}(P^1) = S_{\pi(2),3,2}(P^1) + p^1_{\pi(2),3} = 17 + 5 = 22$

(4) $S_{\pi(3),1,1}(P^1) = \max(C_{\pi(2),1,2}(P^1), \quad S_{\pi(2),2,2}(P^1)) = 12$,
$C_{\pi(3),1,1}(P^1) = S_{\pi(3),1,1}(P^1) + p^1_{\pi(3),1} = 12 + 3 = 15$,
$S_{\pi(3),2,1}(P^1) = \max(C_{\pi(3),1,1}(P^1), S_{\pi(2),3,2}(P^1)) = 17$,
$C_{\pi(3),2,1}(P^1) = S_{\pi(3),2,1}(P^1) + p^1_{\pi(3),2} = 17 + 7 = 24$,
$S_{\pi(3),3,1}(P^1) = \max(C_{\pi(3),2,1}(P^1), \quad C_{\pi(2),3,2}(P^1)) = 24$,
$C_{\pi(3),3,1}(P^1) = S_{\pi(3),3,1}(P^1) + p^1_{\pi(3),3} = 24 + 4 = 28$

So, $C_{\pi(n),m,w_{\pi(n)}}(P^1) = 28$.

Similarly the corresponding makespan values of $C_{\pi(n),m,w_{\pi(n)}}(P^2)$ and $C_{\pi(n),m,w_{\pi(n)}}(P^3)$ are yielded using Eq. (3-10), respectively, that is, $C_{\pi(n),m,w_{\pi(n)}}(P^2) = 33$ and $C_{\pi(n),m,w_{\pi(n)}}(P^3) = 32$.

Third, we can obtain $f_1$ and $f_2$ by plugging the values yielded in Step 2 into Eqs. (1-2),

$$f_1 = \frac{1}{3}\sum_{s=1}^{3} C_{\pi(n),m,w_{\pi(n)}}(P^s) = \frac{28 + 33 + 32}{3} = 31$$

$$f_2 = \sqrt{\frac{\sum_{s=1}^{3} (C_{\pi(n),m,w_{\pi(n)}}(P^s) - f_1)^2}{3 - 1}} = 2.65$$

It is notable that the values of the mean and the standard variance of makespan have different ranges. For easier trade-off decisions, it is helpful to normalize their values into the same range, [0, 1], using $\frac{f_1 - \min(f_1)}{\max(f_1) - \min(f_1)}$ and $\frac{f_2 - \min(f_2)}{\max(f_2) - \min(f_2)}$, where $\min(f_1)$ and $\max(f_1)$ are the minimum and maximum mean values of the makespan, respectively; $\min(f_2)$ and $\max(f_2)$ are the minimum and maximum standard variance values of the makespan, respectively.

## IV. INTRODUCTION TO THE BASIC MBO ALGORITHM

MBO algorithm, a neighborhood search technique, is inspired from the V flight formation [48]. The MBO algorithm is an iterative process, like the other swarm intelligence based algorithm, and consists of four main parts. One is the initialization, in which a number of initial solutions corresponding to birds are randomly generated, and place them on a hypothetical V formation arbitrarily (showing in lines 2-3). Then, the remaining three parts, i.e., improving the leading solution, improving the other solutions in the population (except leading solution) and replacing the leading solution, are repeated until the termination criterion is satisfied. In the improving the leading solution phase (referring to lines 8-12), some neighbor solutions are generated by pairwise exchange of any two locations of the current leading solution, and the best solution is selected to update the leading one. In the improving the other solutions phase (lines 13-20), each solution is tried to be improved by its neighbor one and the best neighbor solution of the previous one. Following that a replacing process of the leading solution is implemented, in which the leading solution is moved to the end and the rest solutions are moved forward a position in turn. The detailed process of the above procedure is stated in Algorithm 1 [47], [48].

In Algorithm 1, the parameters, *PS*, *nTour*, *nBor*, and *nShare* are suggested to set 51, 10, 3, and 1, respectively by Duman *et al.* [48]. Furthermore, to verify the impact of different values of these five parameters on the performance of algorithm for the flow shop scheduling problem, Zhang *et al.* [20] have proposed four reasonable levels for each of the five parameters. From the experimental results, it can be observed that the delta of response values for the parameter *PS* is highest, but for the other three parameters the delta is much lower. This can indicate that the parameter *PS* has the important significance, whereas the other parameters are less critical. Finally, the above five parameters are same suggested to set 51, 10, 3, and 1, respectively. So, for the same flow shop scheduling problem, in our proposed algorithm,

**Algorithm 1** The Basic MBO Algorithm

**Procedure** *basic MBO*($PS$, $Iter_{\max}$, $nTour$, $nBor$, $nShare$, $x$)
**Input**:
the number of initial solutions (birds), $PS$, the maximal iteration limit, $Iter_{\max}$, the lifetime of the leader bird, $nTour$, the number of neighbor solutions, $nBor$, the number of neighbor solutions to be share with the next solution, $nShare$, the solution in the population, $x$
**Output**: the best solution found so far
01: **Begin**
02: Randomly initialize $PS$ solutions ***//initialization population***
03: Select a best solution as a leading one and $PS-1$ solutions place on a hypothetical V formation arbitrarily, and $iter$=1
04: **While** $iter<=Iter_{\max}$ **do**
05: $nt$=1
06 **While** $nt<=nTour$ **Do** ***// the lifetime of the leader bird***
07: $nb$=1
08: **While** $nb<=nBor$ **Do** ***//improve the leading solution***
09: Generated a neighbor solution by pairwise exchange of any two locations of the leading solution, and calculate its function value
10: $nb=nb+1$
11: **End While**
12: Find the best neighbor solution to update the leading one, and the remaining $nBor-1$ solutions are put into two shared neighbor sets, respectively
13: ***// improve the other solutions in the population(except leading solution)***
14: $s = 2$
15: **While** $s <=PS$**Do**
16: Generate ($nBor$-$nShare$) solutions by pairwise exchange of any two locations of the current solution, $x_s$, and calculate the corresponding to function value
17: Select the best solutions among ($nBor$-$nShare$) solutions and the $nShare$ unused best neighbors of the previous solutions to update the current one
18: $s = s + 1$; $iter=iter+nbor-nShare$
19: **End While**
20: **End While**
21: Move the leader solution to the end and forward one of the solutions following it to the leader position, and $iter=iter+1$ ***//replace the leading solution***
22: **End While**
23:**End**

we fix the parameters $PS$, $nTour$, $nBor$, and $nShare$ at the values recommended by Zhang *et al.* [20] and Duman *et al.* [48]. For more details about the parameters setting, please refer to [20] and [48].

From Algorithm 1, MBO distinguishes from other metaheuristic methods, its properties are that it is a parallel processing and exists benefit mechanism for the solutions (birds) from the solutions in front of them, in which the best unused neighbors are shared with the solutions that follow (here 'unused' refers to a neighbor solution that is not used to replace the existing solution). Although MBO appears to have some similarities to swarm intelligence algorithms, i.e., an artificial bee colony algorithm, in particular in which better solutions are explored more, the benefit mechanism is totally unique to MBO [48].

## V. THE PROPOSED MOMBO

The basic MBO algorithm is of continuous nature and often originally used to design continuous function optimization. In order to generate a feasible job sequence for the problem considered, in this section, we present a discrete MOMBO. The proposed algorithm mainly includes the initialization population, improving the leading solution, improving the other solutions and local search except for the termination condition and the parameter setting.

### A. INITIALIZATION POPULATION

To generate an initial population with a certain level of quality and diversity, many heuristics, i.e., PFE (Profile Fitting combining NEH (Nawaz, Enscore and Ham)) have been successfully adapted to initialize the seeds of the population [54]. The PFE heuristic from Ronconi [54] is a combination of the PF and NEH heuristics. PFE can explore the blocking constraint due to it is to seek a schedule (solution) that makes the makespan of the total time of blocking and idle time of machines minimal. Furthermore, PFE has been claimed to have a better performance in optimizing the blocking

**Algorithm 2** Multiple-Based PFE

**Input:** the number of jobs, $n$
**Output:** $\lambda$ solutions
01:**Begin**
02: Let $\pi_i = \phi$, $\pi' = \phi$, $j' = 0$
03: **For** $i = 1$ **to** $n$
04: $\pi(i) = i$
05: **End for**
05: Set $\pi'(1) = \arg\min_{j\in\pi} \sum_{t=1}^{m} \overline{p}_{\pi(j),t}$; $\pi = \pi\ \pi'(1)$
06: **For** $j' = 1$ **to** $n$
07: **For** $i = 1$ **to** $n - j'$
08: Put $\pi(i)$ into behind of $\pi'(j')$ and considered it as $\pi'(j'+1)$
09: Calculate $\eta(i) = \sum_{t=1}^{m} (C_{\pi'(j'+1),m,w_{\pi'(j'+1)}}(\overline{P}) - C_{\pi'(j'),m,w_{\pi'(j')}}(\overline{P}) - \overline{p}_{\pi'(j'+1),t})$ (seeing to Fig.1)
10: **End For**
11: Set $\pi'(j'+1) = \arg\min_{i=(1,\ldots,n-j')} \eta(i)$;
12: Set $\pi = \pi\ \pi'(j'+1)$ and $j' = j' + 1$
13: **End For**
14: Pick the first 2 jobs of $\pi'$, form two subsequences, $\{\pi'(1), \pi'(2)\}$ and $\{\pi'(2), \pi'(1)\}$, evaluate the quality of the 2 subsequences, and select the one with the minimal makespan
15: **For** $i = 3$ **to** $n-1$
16: Pick the $i$th job from $\pi'$, obtain $i$ subsequences by inserting it into the current sequence, $\pi^*$, at $i$ possible positions, and select the subsequence with the minimal makespan as the current sequence, $\pi^*$
17:**End for**
18: /* *There is no following statements for PFE** */
19: Insert $\pi'(n)$ into the current sequence $\pi^*$ at $n$ possible positions, calculate their makespan using all sample values of processing time, and simultaneously evaluate two objectives, $f_1$ and $f_2$. Denote the $n$ complete sequences as $TS$;
20: Set $O = \phi$ and $TS' \leftarrow TS$
19: **While** $|O| < \phi$ **do**
21: Seek non-dominated solution(s) in $TS' \rightarrow D$ based on the Pareto dominance relation;
22: Set $k = |D|$
23: **If** $k \geq (\lambda - |O|)$ **then**
24: Randomly select $\beta - |O|$ non-dominated solution(s) from $D \rightarrow \varepsilon$
25: $O = O \cup \varepsilon$
26: **Else**
27: Set $O = O \cup D$ and $TS' = TS'\ D$
28: **End If**
29: **End While**
30: **Output** $\lambda$ solutions in $O$
31: **End**

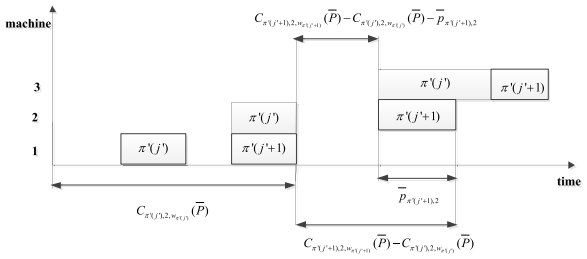



**FIGURE 1.** The process of calculating $\eta$.

flow shop scheduling problem than other heuristics [55]. However, PFE can only generate a single solution. To efficiently produce multiple solutions for the multi-objective scheduling problem considered in this study, a multiple-based PFE, denoted as m-PFE is designed to generate individuals for initializing part of the population. To maintain the diversity of the population, the rest individuals are randomly generated in the neighborhood of these individuals. The following algorithm shows the detailed steps of $m$-PFE, in which $\overline{p}_{\pi(j),t}$ is the average sample value of the process time of job $\pi(j)$ on machine $t$. The detailed process of generating $\lambda$ solutions is shown in Algorithm 2.

### B. IMPROVING THE LEADING SOLUTION

The exploitation ability of the leading solution will be enhanced by slightly disturbing the neighboring solution. In this section, we adopt three strategies based on insert,

swap, and inverse operators to disturb the current leading solution. The motives are that, (1) insertion, swap and inverse operators are commonly used to produce a promising neighboring solution to enhance the solution's exploitation ability, and they have been demonstrated their superiority to generate a neighboring the solution; (2) to enrich the neighborhood structure and diversify the population, more strategies can generate different solutions with a larger probability than a single strategy, and avoid the population trapping in local optima. Thus, with the above motivations, one of the above three strategies are randomly chosen to generate solutions, in which the best neighbor solution is selected to update the leading solution, and the remaining solutions are put into two shared neighbor sets, respectively. The three strategies are given: (1) perform insert once; (2) apply swap one time; (3) conduct inverse once. For more details about the above operators, please refer to [5] and [6].

### C. IMPROVING THE OTHER SOLUTIONS IN THE POPULATION

The process of improving the other solutions in the population plays an important role, whose contribution is that it can lead the offspring to the global good solution, and improve the convergence of the algorithm. Due to the multi-objective optimization problem has many non-dominated solutions with a high quality, taking full advantage of the valuable information of non-dominated solutions will lead the population toward the Pareto-optimal front. Thus, to improve the algorithm's efficiency and effectiveness in this paper, we first utilize the valued information of the current archive that includes all the non-dominated founded so far to construct a probabilistic model, and then estimate the probability distributions to generate a number of solutions with high quality. The detailed description is given as follows.

(1) Learning and constructing the probabilistic model

First, select $PS$ promising solutions to put into the candidate population, $[\psi]_{PS\times n}$, according to Algorithm 3.

Second, according to the information of $[\psi]_{PS\times n}$, two matrixes, named $[\rho_{i,j}]_{n\times n}$ and $[\beta_{i,j}]_{n\times n}$, are established based on the order of jobs in the permutation and the similar blocks of jobs, respectively.

$$[\rho_{i,j}]_{n\times n} = \begin{bmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,n} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{n,1} & \rho_{n,2} & \cdots & \rho_{n,n} \end{bmatrix}$$

$$[\beta_{j',j}]_{n\times n} = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,n} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \beta_{n,1} & \beta_{n,2} & \cdots & \beta_{n,n} \end{bmatrix}$$

where $\rho_{i,j}$ is the number of times that job $j$ appears in position $i$ in $[\psi]_{PS\times n}$, and $\beta_{j',j}$ the number of times that job $j$ appears immediately after the scheduled job $j'$ ($j' \neq j$).

Third, build a probabilistic model $[\xi]_{n\times n}$, in which the probability of each job, $\xi_{i,j}$, is calculated according to Eq. (11).

$$\xi_{i,j} = \begin{cases} \dfrac{\rho_{i,j}}{\sum_{t\in\mu(i)}\rho_{i,t}} & i = 1 \\ \dfrac{\dfrac{\rho_{i,j}}{\sum_{t\in\mu(i-1)}\rho_{i,t}} + \dfrac{\beta_{j',j}}{\sum_{t\in\mu(i-1)}\beta_{j',t}}}{2} & i = 2,3\ldots n, j \in \mu(i-1) \end{cases} \tag{11}$$

where $\mu(i)$ is the unscheduled sequence set, $i$ is the position that job $j$ appears in the sequence, and $j'$ is the just now scheduled job.

**Algorithm 3** Constructing $[\psi]_{PS\times n}$

**Input:** the current archive, $AR$, the number of non-dominated solutions in $AR$, $|AR|$
**Output:** $[\psi]_{PS\times n}$
01: **Begin**
02: **If** $PS \geq |AR|$ **then**
03: $[\psi]_{PS\times n} \xleftarrow{put\ into}$ $AR$ // *solutions in AR are put into the candidate population*$[\psi]_{PS\times n}$
04: $[\psi]_{PS\times n} \xleftarrow{put\ into}$ $PS-|AR|$ solutions are selected using 2- tournament selection from the current population, and put them into $[\psi]_{PS\times n}$
05: **End If**
06: **If** $PS < |AR|$ **then**
07: $[\psi]_{PS\times n} \xleftarrow{put\ into}$ $|AR|-PS$ solutions are selected using 2- tournament selection from $AR$
08: **End If**
09: **Output** $[\psi]_{PS\times n}$

**Algorithm 4** Generate New Solutions

**Input:** Unscheduled sequence $\mu=\phi$, $[\xi]_{n\times n}=\phi$, $\pi_{\text{new}}=\varphi$
**Output:** some solutions
01: **Begin**
02: Randomly select $\tau$ solutions from the current population, and set $s = 1$
03: **While** $s < \tau$ **do**
04: Set $i = 1$
05: $\mu(i) \xleftarrow{assign}$ the $s$th solution of the population are assigned $\mu(i)$
06: $j \xleftarrow[are\ taken]{randomly}$ 5 jobs in $\mu(i)$ are randomly taken, respectively, and compute $\xi_{i,j}$
07: $\pi_{new}(i) = \arg\max_{j\in 5jobs}\xi_{i,j}$, and set $i = 2$
08: **While** $i <= n$ **do**
09: $\mu(i) = \mu(i-1)\ \pi_{new}(i-1)$, and calculate $\xi_{i,j}$, $j \in \mu(i-1)$
10: $\pi_{new}(i) = \arg\max_{j\in\mu(i)}\xi_{i,j}$, and set $i = i+1$
11: **End While**
12: Set $s = s+1$
13: **End While**
14: **Output** some solutions

(2) Sampling and generate solutions based on the probabilistic model, $[\xi]_{n\times n}$ according to Algorithm 4.

In lines 7 and 10, if there exist some jobs that their $\xi_{i,j}$ values are equal, one of them will be randomly selected. To clearly demonstrate the process of generating solutions using Algorithm 4, an example is provided here. According to Algorithm 3, construct $[\psi]_{PS\times n}$. Suppose $PS = 6$, $n = 7$. $[\psi]_{6\times 7}$, $[\rho_{i,j}]_{7\times 7}$, and $[\beta_{i,j}]_{7\times 7}$ are given as follows:

$$[\psi]_{6\times 7} = \begin{bmatrix} 1 & 5 & 7 & 3 & 2 & 4 & 6 \\ 6 & 5 & 3 & 1 & 2 & 7 & 4 \\ 2 & 3 & 5 & 1 & 4 & 7 & 6 \\ 6 & 1 & 2 & 5 & 7 & 4 & 3 \\ 2 & 7 & 6 & 4 & 3 & 5 & 1 \\ 3 & 5 & 1 & 4 & 6 & 7 & 2 \end{bmatrix}$$

$$[\rho_{i,j}]_{7\times 7} = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 3 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 & 0 & 3 \\ 1 & 1 & 1 & 1 & 0 & 2 & 0 \end{bmatrix}$$

$$[\beta_{j',j}]_{7\times 7} = \begin{bmatrix} - & 2 & 0 & 2 & 1 & 0 & 0 \\ 0 & - & 1 & 1 & 1 & 0 & 2 \\ 1 & 1 & - & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & - & 0 & 2 & 1 \\ 3 & 0 & 1 & 0 & - & 0 & 2 \\ 1 & 0 & 0 & 1 & 1 & - & 1 \\ 0 & 1 & 1 & 2 & 0 & 2 & - \end{bmatrix}$$

Set $i = 1$, the first solution of the current population assigns to $\mu(1)$, $\mu(1) = \{2, 4, 1, 3, 7, 6, 5\}$, $\pi_{\text{new}} = \varphi$ Randomly select 5 jobs from $\mu(1)$, i.e., 2,1,7, 6 , 5, and compute their probabilities.

$$\begin{aligned} \xi_{1,2} &= 2/(2+0+1+1+0+2+0) = 0.333 \\ \xi_{1,1} &= 1/(2+0+1+1+0+2+0) = 0.167 \\ \xi_{1,7} &= 0/(2+0+1+1+0+2+0) = 0 \\ \xi_{1,6} &= 2/(2+0+1+1+0+2+0) = 0.333 \\ \xi_{1,5} &= 0/(2+1+1+1+1+0+1) = 0 \end{aligned}$$

$\pi_{new}(1) = \arg\max_{j\in 5jobs} \xi_{i,j} =$ 2or 6, we randomly select one of them, such as, $\pi_{new}(1) = 6$.

Set $i = 2$, $\mu(i) = \mu(i-1)\backslash\pi_{new}(i-1)$, i.e., $\mu(i) = \{2, 4, 1, 3, 7, 5\}$. Calculate the probability of each job in $\mu(i)$:

$$\begin{aligned} \xi_{2,2} &= (0/(0+0+1+1+1+3) \\ &\quad +0/(0+1+1+0+1+1))/2 = 0 \\ \xi_{2,4} &= (0/(0+0+1+1+1+3) \\ &\quad +1/(0+1+1+0+1+1))/2 = 0.125 \\ \xi_{2,1} &= (1/(0+0+1+1+1+3) \\ &\quad +1/(0+1+1+0+1+1))/2 = 0.208 \\ \xi_{2,3} &= (1/(0+0+1+1+1+3) \\ &\quad +0/(0+1+1+0+1+1))/2 = 0.083 \\ \xi_{2,7} &= (1/(0+0+1+1+1+3) \\ &\quad +1/(0+1+1+0+1+1))/2 = 0.208 \\ \xi_{2,5} &= (3/(0+0+1+1+1+3) \\ &\quad +1/(0+1+1+0+1+1))/2 = 0.375 \end{aligned}$$

$\pi_{new}(2) = \arg\max_{j\in\mu(2)} \xi_{i,j} = 5$, $i = i+1$, if $i < n$,
go to lines 9-11;
otherwise, execute line 12.

### D. THE REFERENCE-POINT-ASSISTED LOCAL SEARCH

For the multi-objective optimization problem, an achievement scalar function is considered to select a solution among ones generated by local search [21], [56]. Inspired from the above ideas, a reference-point-assisted local search strategy is adopt to select solutions based on their distance to the reference point [21]. In Algorithm 5, firstly, randomly select a solution $\pi$ from the current population according to a probability of $pls \in [0, 1]$. Secondly, the selected solution will be performed an insert-neighborhood-based local search presented in [5] and [6], and obtain a number of neighborhood solutions. Then, an reference point is generated, the $k$-th dimension of which is equal to the best value of the $k$-th objective function in the current population, and lastly the distance between the objective values of $\pi_i$ (the $i$-th neighborhood solutions) generated by the insert-and-neighborhood-based local search and the reference point is computed as follows.

$$d(\pi_i) = \sqrt{\sum_{k=1}^{2} (F_k(\pi_i) - \overline{r_k})^2} \quad i = 1, 2, \ldots, n \qquad (12)$$

where $\overline{r} = (\overline{r_1}, \ldots, \overline{r_k})$ is the reference point, $\overline{r_k}$ is the best value of the $k$-th objective function in the current population. $d(\pi_i)$ is the distance between the objective value of the mutant

**Algorithm 5** The Reference-Point-Assisted Local Search (RPALS)

**Input:** An ideal point, $\overline{r}$; a reference solution, $\pi^{nd} = \{\pi^{nd}(1), \pi^{nd}(2), \ldots, \pi^{nd}(n)\}$, randomly selected from the non-dominated set
**Output:**A solution $\pi^*$
01: **Begin**
02: Select a solution, $\pi = \{\pi(1),\pi(2),\ldots,\pi(n)\}$, with a probability $pls$ from the current population
03: **For** $j = 1$ **to** $n$
04: $\pi' = \pi\ \pi^{nd}(j)$
05: $\pi'_i = \pi' \xleftarrow[\text{ith position}]{insert\ into} \pi^{nd}(j), i = 1, 2, \ldots n$
06: $\pi^* = \arg\min_{\pi'_i} d(\pi'_i), i = 1, 2, \ldots n$
07: **If** $d(\pi^*) < d(\pi)$ **then**
08: $\pi = \pi^*$
09: Set $j = j + 1$
10: **End For**
11**: End**

solution $\pi_i$, and the reference point. Clearly, the smaller $d(\pi_i)$, the closer the solution is to the reference point in the objective space. Based on the above distance, the solution with minimal distance value will be selected to replace $\pi$.

In line 5, insert $\pi^{nd}(j)$ into $\pi'$ at $n$ different positions, respectively, and obtain $n$ different sequences, $\pi'_i(i = 1, 2, \ldots n)$. In line 6, evaluate these sequences, and select the solution with minimal $d(\pi'_i)$ as $\pi^*$. The Fig. 2 provides an illustration of RPALS in which a dot represents a solution in the temporary population. Solution $\pi$ (denoted by the shaded circle) in Fig. 2(a) indicates a solution that is selected to perform local search; Fig. 2(b) shows four neighborhood solutions obtained using the local search proposed in [5] and [6], $\pi_1$, $\pi_2$, $\pi_3$, and $\pi_4$, each denoted by a circle; Fig. 2(c) shows that a reference point, $\bar{r}$ is generated, denoted by a diamond. $d(\pi)$, $d(\pi_1)$, $d(\pi_2)$, $d(\pi_3)$, and $d(\pi_4)$, notated as $d$, $d1$, $d2$, $d3$, and $d4$, for short are the distance between $\pi$, $\pi_1$, $\pi_2$, $\pi_3$, $\pi_4$ and $\bar{r}$, respectively; Fig. 2(d) indicates that the solution $\pi_4$ has the minimal distance to the reference point, which is now denoted by $\pi^*$, which is used to replace solution $\pi$.

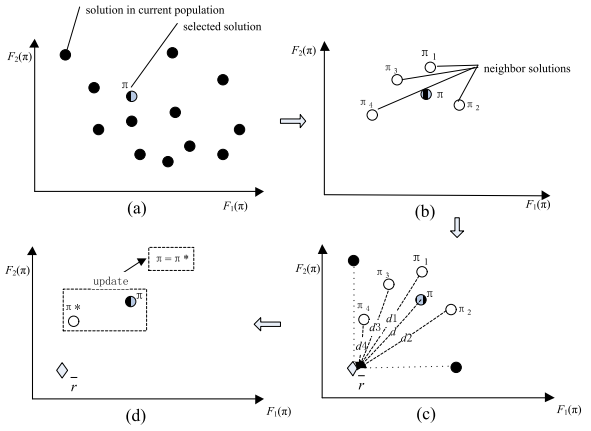



**FIGURE 2.** An example illustrating the RPALS method.

### E. COMPUTATIONAL COMPLEXITY OF MOMBO

In each generation, the main difference between the proposed MOMBO and most existing MOEAs lies in the initialization population, the process of generating new solutions, and local search. In the following, we analyze the computational complexity of these strategies.

Assume there is a population with its size of $PS$ to solve an optimization problem with $b$ objectives and $n$-dimension decision variable. The complexities of the initialization population, the process of generating new solutions and the local search are $O(n^3)$, $O(PS^*n^2)$ and $O(PS^*n^2)$, respectively. Thus, the overall complexity of the proposed algorithm will be $max(PS^*n^2, n^3)$. The most state-of-the-art MOEAs, e.g., NSGA-II, MOEA/D, MMOEA-C&D, are $O(b^*n^2)$ [57]. The complexity of MOMBO is slightly high to that of the above MOEAs for multi-objective optimization problems, but is the same or low to that of the above MOEAs for many objective optimization problems. In our paper, we adopt the maximal elapsed CPU time of milliseconds as the stopping criterion. Although the complexity of MOMBO is slightly high to the compared algorithms, the MOMBO outperforms the compared algorithms in convergence and distributivity as mentioned in Section VI.

## VI. SIMULATION RESULTS

### A. EXPERIMENT SETTINGS

In this section, the proposed algorithm is evaluated on 180 instances of the BLSFS scheduling problem with the stochastic processing time. The following comparisons have been performed.

- ✓ Comparison between the proposed $m$-PFE and PFE heuristics
- ✓ Comparison between the RPALS and LS methods
- ✓ Comparison with other existing multi-objective algorithms
- ✓ The convergence trends of all the compared algorithms on 6 instances
- ✓ The Pareto fronts of all the compared algorithms on 6 instances

All the algorithms adopt the same maximal elapsed CPU time with the unit of millisecond as the termination criterion. All the algorithms are written in Visual C++ 6.0 and the same library functions are adopted in this study to make a fair comparison. For their implementations, all the algorithms are realized on a PC with Pentium (R) Dual 2.79 GHz and 1.96 G memory, in which the operating system is Microsoft Windows 7 X64. In addition, the same background running environment is employed, the background processes that may occupy system resources are closed, and no other programs are executed in parallel during implementing an algorithm.

All performance comparisons are conducted using the Hypervolume (HV for short) [21]. HV is a comprehensive indicator, which reflects not only the convergence performance but also the spread performance of the algorithm. Its reference point is chosen as (1, 1). A larger HV indicates better performance. To further verify the convergence of the proposed algorithm, the distance between the reference set and the non-dominated solution set obtained by an algorithm is calculated as D-metric, and the number of non-dominated solutions obtained by an algorithm is computed as R-metric. For more details about the D-metric and R-metric, please refer to [4].

For the BLSFS scheduling problem focused on in this study, the standard test instances used in the experiments were proposed by Yoon and Ventura [58] and Tseng and Liao [59]. The test set is composed of 180 instances, which are divided into 18 subsets, with each subset consisting of ten instances of the same size. For each subset, the size of instances is changed from 30 jobs and 5 machines to 500 jobs and 20 machines. Each instance is independently executed five replications. The related data, i.e., the due date and the number of sublots of each job are provided based on the discrete uniform distributions listed in Table 1, and we randomly sample 300 cases

**TABLE 1. Parameter settings.**

| parameter | notation | Value |
|---|---|---|
| Test instance parameters | | |
| number of jobs | $n$ | $n$=30,50,70,90,110,200,500 |
| number of machines | $m$ | $m$=5,10,20 |
| due date of job $j$ | $d_j$ | $d_j = rand()\%(15\times m+1)+15\times n$ |
| number of sub-lots of job $j$ | $w_{\pi(j)}$ | $w_{\pi(j)} = rand()\%6+1$ |
| The algorithmic parameters | | |
| number of initialization seed | $\lambda$ | 10 |
| local search rate | $pls$ | 0.5 |
| archive size | $AR$ | 100 |
| stopping time | $T$ | $a\times n\times m$ milliseconds |

from the stochastic processing time between [1] and [31] using MC simulation.

In Table1, for the test instance parameters, we fix the parameters, i.e., $n$, $m$, $d_j$, $w_{\pi(j)}$, at the values recommended by Yoon and Ventura [58] and Tseng and Liao [59]. For the algorithmic parameters, we fix the parameters *PS*, *nTour*, *nBor*, and *nShare* at the values recommended by Zhang *et al.* [20] and Duman *et al.* [47]. For the parameters, $\lambda$ and *pls*, their values are set by two sensitivity analyses in Section VI.B and Section VI.C, respectively.

### B. COMPARISONS BETWEEN THE PROPOSED m-PFE, NEH AND PFE

To evaluate the performance of the proposed *m*-PFE, we consider non-dominated solutions obtained by the initialization strategy on the premise of the same values of the parameters. Table 2 lists the experimental results of PFE, NEH and *m*-PFE in terms of HV, D-metric, and R-metric for the 18 scheduling test sets. Fig. 3 gives a sensitivity analysis on the parameter $\lambda$. Without loss of generality, the value of $\lambda$ is set 0, 2, 4, 5, 7, 9, 10 and 11, respectively.

**TABLE 2. Experimental Results of NEH, PFE and *m*-PFE When $\alpha = 30$.**

| instance | HV | | | D-metric | | | R-metric | | |
|---|---|---|---|---|---|---|---|---|---|
| | NEH | PFE | *m*-PFE | NEH | PFE | *m*-PFE | NEH | PFE | *m*-PFE |
| 30×5 | 0.572709 | 0.565231 | **0.887847** | 0.69 | 0.71 | **0.16** | 0.29 | 0.23 | **0.66** |
| 30×10 | 0.572864 | 0.575885 | **0.798283** | 0.5 | 0.5 | **0.03** | 0.22 | 0.22 | **0.83** |
| 30×20 | 0.488421 | 0.566432 | **0.898531** | 0.84 | 0.76 | **0.38** | 0.11 | 0.16 | **0.57** |
| 50×5 | 0.674703 | 0.727112 | **0.897821** | 0.61 | 0.54 | **0.25** | 0.13 | 0.21 | **0.89** |
| 50×10 | 0.647204 | 0.667671 | **0.768839** | 0.51 | 0.48 | **0.21** | 0.21 | 0.33 | **0.89** |
| 50×20 | 0.669029 | 0.708219 | **0.789767** | 0.48 | 0.45 | **0.12** | 0.18 | 0.23 | **0.81** |
| 70×5 | 0.74908 | 0.768322 | **0.89983** | 0.36 | 0.34 | **0.28** | 0.46 | 0.53 | **0.87** |
| 70×10 | 0.649221 | 0.66856 | **0.811026** | 0.61 | 0.54 | **0.13** | 0.33 | 0.41 | **0.76** |
| 70×20 | 0.802766 | **0.869618** | 0.791945 | 0.15 | **0.12** | 0.34 | 0.61 | **0.79** | 0.56 |
| 90×5 | 0.551003 | 0.743727 | **0.873567** | 0.42 | 0.31 | **0.12** | 0.23 | 0.41 | **0.67** |
| 90×10 | 0.684738 | 0.670083 | **0.834053** | 0.24 | 0.34 | **0.02** | 0.25 | 0.21 | **0.89** |
| 90×20 | 0.755355 | **0.877278** | 0.794419 | 0.27 | **0.12** | 0.23 | 0.55 | **0.71** | 0.61 |
| 110×5 | 0.545548 | 0.774014 | **0.845065** | 0.42 | 0.22 | **0.08** | 0.11 | 0.21 | **0.79** |
| 110×10 | 0.583113 | 0.774468 | **0.891317** | 0.35 | 0.18 | **0.08** | 0.17 | 0.28 | **0.91** |
| 110×20 | 0.632048 | 0.714074 | **0.839378** | 0.61 | 0.54 | **0.12** | 0.25 | 0.31 | **0.92** |
| 200×10 | 0.857435 | 0.812353 | **0.907418** | 0.26 | 0.35 | **0.09** | 0.31 | 0.25 | **0.81** |
| 200×20 | 0.629985 | 0.853462 | **0.876245** | 0.31 | **0.05** | **0.05** | 0.63 | 0.85 | **0.86** |
| 500×20 | 0.7771 | 0.797076 | **0.937646** | 0.28 | 0.26 | **0.07** | 0.33 | 0.43 | **0.89** |

From Fig. 3, all the HV, D-metric and R-metric values for $\lambda = 10$ are significantly better than $\lambda = 0, 2, 4, 5, 7, 9$ and 11.

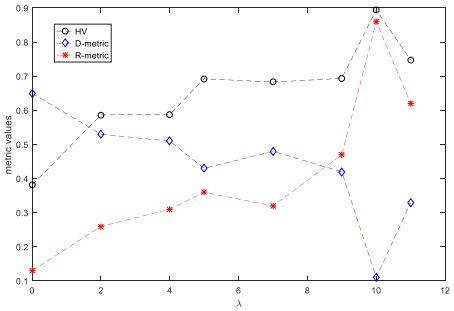



**FIGURE 3. Influences of λ on HV, D-metric, and R-me.**

Table 2 reports that, with respect to HV and R-metric, *m*-PFE is superior to NEH and PFE for 18 and 16 out of 18 (16/18=89%) scheduling instances, respectively. For D-metric, the values obtained by *m*-PFE are better than those of PFE on 15 out of 18(15/18=83%), and NEH on 18 scheduling instances. The reason why *m*-PFE is better than PFE and NEH is that PEF and NEH generate only one good solution to improve the quality of the initial population; whereas *m*-PFE retains a number of good solutions based on the Pareto dominance relation, by inserting the last job into the current sequence at some possible positions and simultaneously optimizing multiple objectives.

### C. COMPARISONS BETWEEN THE PROPOSED RPALS AND LS

The parameter, *pls*, is important for IPLS. Thus, we first give a sensitivity analysis on the parameter before demonstrating the effectiveness of the proposed local search algorithm. Without loss of generality, the value of *pls* is tuned to change from 0 to 1.0 with the step size of 0.1. The instances and the parameter settings are the same as those in Section VI. The results with respect to HV, D-metric, and R-metric are plotted in Fig. 3, respectively. In addition, Table 3 lists the non-parametric test results of six special parameter values on HV, D-metric and R-metric indicators by employing the Mann-Whitney U distribution test. For two arbitrary parameter values, $value_1$ and $value_2$, ($value_1, value_2 \in \{0, 0.4, 0.5, 0.6, 0.7, 0.9\}$), '+' ('−') suggests the indicator value obtained when $pls$=$value_1$ is significantly superior (inferior) to the one obtained when $pls$=$value_2$, while '0' indicates that there is no significant difference between them.

From Fig. 4, we can observe that although no clear relationship between the probability of using local search and performance enhancement can be observed, the metrics values obtained with local search are better than that without local search. This indicates that the local search is always beneficial, and the HV and D-metric values for $pls = 0.5$ are significantly better than $pls$=0, 0.4, 0.6, 0.7 and 0.9. Similarly, all R-metric value with local search are better than without, and the enhancement achieves the maximum when $pls = 0.6$. From these results, we set the value of *pls* to 0.5, which works well both in terms of HV, D-metric and R-metric indicators.

**TABLE 3.** Non-parametric test results when $\alpha = 30$.

| *pls* | HV | | | | | | D-metric | | | | | | R-metric | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0.4 | 0.5 | 0.6 | 0.7 | 0.9 | 0 | 0.4 | 0.5 | 0.6 | 0.7 | 0.9 | 0 | 0.4 | 0.5 | 0.6 | 0.7 | 0.9 |
| 0 | 0 | - | - | - | - | - | 0 | - | - | - | - | - | 0 | - | - | - | - | - |
| 0.4 | + | 0 | - | - | - | - | + | 0 | - | - | + | + | + | 0 | - | - | - | - |
| 0.5 | + | + | 0 | 0 | + | + | + | + | 0 | + | + | + | + | + | 0 | 0 | + | + |
| 0.6 | + | + | 0 | 0 | + | + | + | + | - | 0 | + | + | + | + | 0 | 0 | + | + |
| 0.7 | + | + | - | - | 0 | 0 | + | - | - | - | 0 | 0 | + | + | - | - | 0 | - |
| 0.9 | + | + | - | - | 0 | 0 | + | - | - | - | 0 | 0 | + | + | - | - | + | 0 |

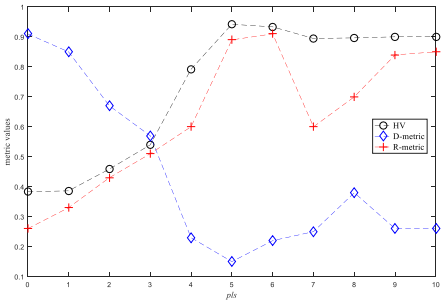



**FIGURE 4.** Influences of *pls* on HV, D-metric, and R-metric.

Based on the above observations regarding *pls*, we compare the proposed local search (RPALS for short) with existing one [4]. In [4], a Pareto local search (PLS, for short) is proposed to enhance the exploitation performance of the algorithm.

Table 4 shows the comparative results in terms of HV, D-metric and R-metric indicators, where the best result of the comparative methods is highlighted. From Table 4, we can see that on small-scale instances, e.g., 30×5, 30×10, 50×5, PLS shows a good performance in terms of HV, D-metric and R-metric indicators. However, IPLS is superior to the compared PLS on all other instances, and it can be seen that the superiority of IPLS becomes more significant as the size of the scheduling problem increases. To summarize, the proposed local search using the ideal point-based solution selection criterion is able to guide the search efficiently, thereby improving the convergence capability of the proposed algorithm.

**TABLE 4.** Performances of the proposed and the compared local search strategies when $\alpha = 30$.

| instance | HV | | D-metric | | R-metric | |
|---|---|---|---|---|---|---|
| | PLS | RPALS | PLS | RPALS | PLS | RPALS |
| 30×5 | **0.896688** | 0.638043 | **0.11** | 0.52 | **0.75** | 0.25 |
| 30×10 | **0.900172** | 0.738886 | **0.25** | 0.65 | **0.72** | 0.33 |
| 30×20 | 0.740538 | **0.864038** | 0.37 | **0.29** | 0.41 | **0.57** |
| 50×5 | **0.911653** | 0.740781 | **0.13** | 0.54 | **0.87** | 0.23 |
| 50×10 | 0.814802 | **0.863326** | 0.33 | **0.24** | 0.53 | **0.64** |
| 50×20 | 0.502867 | **0.650319** | 0.66 | **0.23** | 0.43 | **0.81** |
| 70×5 | 0.811201 | **0.817554** | 0.20 | **0.19** | **0.67** | 0.66 |
| 70×10 | 0.604535 | **0.750591** | 0.25 | **0.20** | 0.41 | **0.78** |
| 70×20 | 0.550463 | **0.812318** | 0.23 | **0.18** | 0.33 | **0.75** |
| 90×5 | 0.620751 | **0.854053** | 0.22 | **0.11** | 0.30 | **0.73** |
| 90×10 | 0.407762 | **0.785367** | 0.45 | **0.16** | 0.15 | **0.91** |
| 90×20 | 0.398350 | **0.897952** | 0.46 | **0.09** | 0.37 | **0.86** |
| 110×5 | 0.578368 | **0.858879** | 0.21 | **0.17** | 0.27 | **0.87** |
| 110×10 | 0.688680 | **0.900389** | 0.31 | **0.07** | 0.22 | **0.92** |
| 110×20 | 0.710318 | **0.910476** | 0.25 | **0.06** | 0.41 | **0.91** |
| 200×10 | 0.480352 | **0.771626** | 0.35 | **0.11** | 0.33 | **0.87** |
| 200×20 | 0.541102 | **0.806651** | 0.31 | **0.12** | 0.28 | **0.85** |
| 500×20 | 0.600170 | **0.886368** | 0.32 | **0.11** | 0.33 | **0.89** |

### D. COMPARISON WITH OTHER MULTI-OBJECTIVE ALGORITHMS

To validate the robustness of the proposed algorithm, we compare MOMBO with NSGA-II and the basic MOB [47] in terms of HV, D-metric, and R-metric on 18 scheduling test sets when the stop time is $30 \times n \times m$. All the compared algorithms are implemented on the premise of computational time and experimental environment. In Table 5, the best result of the comparative methods is highlighted. For all metrics, Table 5 reports that MOMBO outperforms NSGA-II and MBO for 18 scheduling test sets in terms $\alpha = 30$, suggesting that MOMBO has better convergence capability and robustness than the others.

In this subsection, the overall performance of the proposed algorithm (MOMBO) is investigated. In 2017, we developed an evolutionary multi-objective robust scheduling algorithm (REMO) to solve the multi-objective BLSFS scheduling problem, where two novel crossover operators are proposed to take advantage of non-dominated solutions, and a rescheduling strategy based on the local search is employed to reduce the negative influence resulted from uncertainty [29]. In 2017, Shen et al. designed a modified Multi-Objective Evolutionary Algorithm based on Decomposition (m-MOEA/D) for robust scheduling, in which a new subproblem update selection strategies are employed to improve the schedule robustness to uncertainties and maintaining a small variance of disrupted objective values [30]. In 2018, Fu et al. adopted a hybrid Multi-Objective Evolutionary Algorithm (h-MOEA) to address a two-agent stochastic flow shop deteriorating scheduling problem. In this algorithm, two populations and one archive are designed to enhance the global and local searches [39].

In this study, we compare MOMBO with h-MOEA, m-MOEA/D and REMO, in terms of HV, D-metric, and R-metric on 18 scheduling test sets when the stop time is $30 \times n \times m$ and $60 \times n \times m$. All the compared algorithms are implemented on the premise of computational time and experimental environment. In Tables 6-7, the best result of the comparative methods is highlighted. For HV metric, Tables 6-7 report that MOMBO outperforms h-MOEA,

**TABLE 5.** Experimental results of naga-ii, mbo and mombo in terms of HV, D-metric, and R-metric measures when $\alpha = 30$.

| instance | HV | | | D-metric | | | R-metric | | |
|---|---|---|---|---|---|---|---|---|---|
| | NSGA-II | MBO | MOMBO | NSGA-II | MBO | MOMBO | NSGA-II | MBO | MOMBO |
| 30×5 | 0.325711 | 0.340454 | **0.691396** | 0.61 | 0.59 | **0.23** | 0.11 | 0.12 | **0.69** |
| 30×10 | 0.247454 | 0.224603 | **0.378081** | 0.41 | 0.39 | **0.28** | 0.21 | 0.21 | **0.48** |
| 30×20 | 0.314312 | 0.254223 | **0.597521** | 0.43 | 0.52 | **0.13** | 0.13 | 0.16 | **0.79** |
| 50×5 | 0.369035 | 0.401294 | **0.753579** | 0.48 | 0.31 | **0.13** | 0.17 | 0.21 | **0.82** |
| 50×10 | 0.488191 | 0.467802 | **0.801956** | 0.38 | 0.38 | **0.11** | 0.25 | 0.21 | **0.86** |
| 50×20 | 0.482404 | 0.364191 | **0.835831** | 0.36 | 0.42 | **0.09** | 0.43 | 0.33 | **0.89** |
| 70×5 | 0.673561 | 0.782101 | **0.974815** | 0.36 | 0.23 | **0.03** | 0.51 | 0.67 | **0.96** |
| 70×10 | 0.58821 | 0.592377 | **0.895101** | 0.22 | 0.18 | **0.06** | 0.45 | 0.49 | **0.73** |
| 70×20 | 0.575309 | 0.61387 | **0.824432** | 0.21 | 0.21 | **0.09** | 0.46 | 0.48 | **0.69** |
| 90×5 | 0.391517 | 0.421035 | **0.847553** | 0.55 | 0.43 | **0.10** | 0.11 | 0.19 | **0.76** |
| 90×10 | 0.546351 | 0.583476 | **0.837783** | 0.23 | 0.19 | **0.07** | 0.29 | 0.36 | **0.78** |
| 90×20 | 0.485454 | 0.56454 | **0.836981** | 0.32 | 0.28 | **0.06** | 0.16 | 0.33 | **0.85** |
| 110×5 | 0.367312 | 0.340603 | **0.768391** | 0.54 | 0.58 | **0.11** | 0.13 | 0.10 | **0.84** |
| 110×10 | 0.411035 | 0.364023 | **0.812483** | 0.38 | 0.42 | **0.16** | 0.25 | 0.31 | **0.79** |
| 110×20 | 0.541091 | 0.514294 | **0.840205** | 0.24 | 0.29 | **0.17** | 0.25 | 0.33 | **0.78** |
| 200×10 | 0.332404 | 0.421802 | **0.900773** | 0.53 | 0.38 | **0.08** | 0.17 | 0.22 | **0.91** |
| 200×20 | 0.427561 | 0.412191 | **0.931553** | 0.39 | 0.45 | **0.11** | 0.13 | 0.09 | **0.92** |
| 500×20 | 0.257821 | 0.283101 | **0.911396** | 0.69 | 0.64 | **0.10** | 0.33 | 0.43 | **0.89** |

**TABLE 6.** Experimental results obtained by all the algorithms in terms of HV, D-metric, and R-metric measures when $\alpha = 30$.

| instance | HV | | | | D-metric | | | | R-metric | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | h-MOEA | m-MOEA/D | REMO | MOMBO | h-MOEA | m-MOEA/D | REMO | MOMBO | h-MOEA | m-MOEA/D | REMO | MOMBO |
| 30×5 | 0.519794 | **0.641574** | 0.329712 | 0.581867 | 0.04 | 0.02 | 0.1 | **0.01** | 0.43 | 0.7 | 0.42 | **0.87** |
| 30×10 | 0.664187 | **0.727196** | 0.676835 | 0.667461 | 0.07 | **0.01** | 0.05 | 0.06 | 0.67 | **0.86** | 0.66 | 0.7 |
| 30×20 | 0.436301 | 0.477136 | 0.467497 | **0.929246** | 0.11 | 0.12 | 0.17 | **0.02** | 0.29 | 0.22 | 0.62 | **0.63** |
| 50×5 | 0.583632 | 0.725613 | 0.786709 | **0.909449** | 0.21 | 0.14 | 0.12 | **0.01** | 0.38 | 0.78 | 0.76 | **0.82** |
| 50×10 | 0.811362 | 0.896657 | 0.876835 | **0.958564** | 0.18 | 0.16 | 0.11 | **0.08** | 0.91 | 0.78 | 0.85 | **0.93** |
| 50×20 | 0.630894 | 0.941956 | 0.856627 | **0.954525** | 0.17 | 0.08 | 0.13 | **0.06** | 0.48 | 0.62 | 0.53 | **0.71** |
| 70×5 | 0.871483 | 0.828176 | 0.867979 | **0.90086** | 0.27 | 0.18 | 0.13 | **0.04** | 0.6 | 0.6 | 0.52 | **0.81** |
| 70×10 | **0.959525** | 0.645122 | 0.586694 | 0.803923 | **0.06** | 0.12 | 0.73 | 0.19 | 0.68 | 0.25 | 0.11 | **0.41** |
| 70×20 | 0.672337 | 0.634121 | 0.916388 | **0.926475** | 0.16 | 0.21 | 0.11 | **0.02** | 0.31 | 0.25 | 0.81 | **0.93** |
| 90×5 | 0.578689 | 0.379891 | **0.986546** | 0.927187 | 0.19 | 0.23 | **0.08** | 0.15 | 0.73 | 0.22 | 0.91 | **0.75** |
| 90×10 | 0.77402 | 0.790352 | 0.973371 | **0.989356** | 0.31 | 0.23 | 0.18 | **0.07** | 0.43 | 0.51 | 0.74 | **0.86** |
| 90×20 | 0.11334 | 0.754505 | **0.943986** | 0.75249 | 0.91 | 0.11 | **0.06** | 0.11 | 0.29 | 0.38 | **0.76** | 0.57 |
| 110×5 | **0.949793** | 0.719189 | 0.92596 | 0.919355 | **0.03** | 0.21 | 0.09 | 0.13 | 0.83 | 0.4 | **0.78** | 0.72 |
| 110×10 | 0.716684 | 0.809305 | 0.832756 | **0.852322** | 0.21 | 0.13 | 0.09 | **0.05** | 0.4 | 0.42 | 0.64 | **0.65** |
| 110×20 | 0.519794 | 0.581867 | 0.629095 | **0.716388** | 0.23 | 0.19 | 0.17 | **0.11** | 0.33 | 0.36 | 0.45 | **0.7** |
| 200×10 | 0.664187 | **0.727196** | 0.629712 | 0.631669 | 0.17 | **0.09** | 0.16 | 0.15 | 0.51 | **0.89** | 0.29 | 0.75 |
| 200×20 | 0.436301 | 0.477136 | 0.622577 | **0.894912** | 0.27 | 0.23 | 0.12 | **0.07** | 0.23 | 0.25 | 0.73 | **0.83** |
| 500×20 | 0.738259 | 0.832756 | 0.985078 | **0.987849** | 0.23 | 0.16 | 0.14 | **0.09** | 0.25 | 0.79 | 0.8 | **0.85** |

m-MOEA/D and REMO for 11 out of 18 and 16 out of 18 scheduling test sets in terms $\alpha = 30$ and $\alpha = 60$, respectively, whereas is inferior to the others for 7 out of 18 and 2 out of 18 test sets, respectively. With respect to D-metric and R-metric, MOMBO achieves a better performance in most test sets.

From the above results, the superiority of MOMBO attributes to the strategy proposed in subsection V.C and the ideal-point assisted local search strategy of subsection V.D, since they improve the capabilities of the algorithm in exploration and exploitation. The reason why MOMBO is worse than h-MOEA, m-MOEA/D and REMO for a number of scheduling instances may be that the proposed algorithm misses some opportunities to generate promising solutions because much time is spent on the local search. In the future, we will research the strategy of reducing the computational complexity of the local search. In summary, MOMBO is comparable to the other three algorithms for most scheduling instances with respect to HV, D-metric, and R-metric.

In addition, Wilcoxon rank sum test with the significance level of 0.05 is employed to determine whether the results obtained by one algorithm are statistically significantly difference from those obtained by the another algorithm. The sign of '+ - =' in A vs. B indicates that according to each metric, method A is significantly better than B, significantly worse than B, or there is no significant difference between A and B. Each value in Table 8 is $p$ value that is the probability of observing the given result by chance if the null hypothesis is true. From Table 8, for the most BLSFS scheduling test sets, MOMBO is significantly different from the other compared algorithms in terms of $30 \times n \times m$.

### E. FURTHER COMPARISONS ON FOUR INSTANCES

To further evaluate the performance of the proposed algorithm, we present the convergence profiles of all the compared algorithms on four instances, namely, 30×5, 50×5, 90×10, and 200×20. We run the algorithm with CPU time

**TABLE 7.** Experimental results obtained by all the algorithms in terms of HV, D-metric, and R-metric measures when $\alpha = 60$.

| instance | HV | | | | D-metric | | | | R-metric | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | h-MOEA | m-MOEA/D | REMO | MOMBO | h-MOEA | m-MOEA/D | REMO | MOMBO | h-MOEA | m-MOEA/D | REMO | MOMBO |
| 30×5 | 0.767953 | 0.980288 | 0.562467 | **0.988368** | 0.21 | 0.15 | 0.32 | **0.01** | 0.43 | 0.88 | 0.39 | **0.92** |
| 30×10 | 0.745499 | **0.747428** | 0.696948 | 0.733659 | 0.09 | **0.02** | 0.18 | 0.11 | 0.59 | 0.63 | 0.53 | **0.64** |
| 30×20 | 0.615041 | 0.809012 | 0.769514 | **0.951107** | 0.17 | 0.08 | 0.11 | **0.02** | 0.44 | 0.77 | 0.64 | **0.85** |
| 50×5 | 0.871114 | 0.73987 | 0.793401 | **0.918501** | 0.09 | 0.22 | 0.16 | **0.03** | 0.79 | 0.61 | 0.72 | **0.83** |
| 50×10 | 0.739438 | 0.829702 | 0.777666 | **0.940116** | 0.17 | 0.15 | 0.12 | **0.04** | 0.72 | 0.79 | 0.75 | **0.83** |
| 50×20 | 0.69869 | 0.753522 | 0.781097 | **0.880805** | 0.29 | 0.11 | 0.09 | **0.04** | 0.52 | 0.63 | 0.67 | **0.79** |
| 70×5 | 0.608176 | 0.626171 | 0.625595 | **0.694057** | 0.22 | 0.12 | 0.17 | **0.09** | 0.71 | 0.8 | 0.83 | **0.84** |
| 70×10 | 0.898644 | 0.808849 | 0.793792 | **0.902471** | 0.07 | 0.13 | 0.15 | **0.02** | 0.66 | 0.61 | 0.65 | **0.75** |
| 70×20 | 0.684205 | 0.743725 | 0.884811 | **0.926193** | 0.23 | 0.1 | 0.09 | **0.04** | 0.29 | 0.55 | 0.61 | **0.87** |
| 90×5 | 0.60921 | 0.561719 | **0.77957** | 0.75046 | 0.22 | 0.29 | 0.08 | **0.11** | 0.46 | 0.59 | 0.76 | **0.83** |
| 90×10 | 0.588938 | 0.770338 | 0.682238 | **0.859145** | 0.2 | 0.09 | 0.12 | **0.03** | 0.17 | 0.64 | 0.53 | **0.88** |
| 90×20 | 0.777323 | 0.835387 | 0.74831 | **0.861783** | 0.26 | 0.13 | 0.18 | **0.08** | 0.33 | 0.47 | 0.33 | **0.68** |
| 110×5 | 0.737983 | 0.689203 | 0.718932 | **0.954525** | 0.17 | 0.24 | 0.19 | **0.01** | 0.58 | 0.25 | 0.43 | **0.81** |
| 110×10 | 0.74313 | 0.715947 | 0.74333 | **0.778176** | 0.12 | 0.19 | 0.11 | **0.06** | 0.63 | 0.57 | 0.68 | **0.72** |
| 110×20 | 0.926193 | 0.912425 | 0.873273 | **0.933278** | 0.11 | 0.11 | 0.16 | **0.09** | 0.86 | 0.72 | 0.55 | **0.91** |
| 200×10 | 0.761719 | 0.6891 | 0.705714 | **0.823835** | 0.12 | 0.06 | 0.68 | **0.08** | 0.55 | 0.44 | 0.61 | **0.79** |
| 200×20 | 0.892484 | 0.890559 | 0.85767 | **0.923474** | 0.18 | 0.08 | 0.12 | **0.07** | 0.43 | 0.73 | 0.59 | **0.89** |
| 500×20 | 0.707603 | 0.745211 | 0.776983 | **0.880069** | 0.11 | 0.08 | 0.06 | **0.02** | 0.66 | 0.77 | 0.83 | **0.91** |

**TABLE 8.** p values of the Wilcoxon two -sided rank sum test results on all the compared algorithms w.r.t HV, D-metric, and R-metric.

| A vs. B | HV | D-metric | R -metric | HV | D-metric | R -metric |
|---|---|---|---|---|---|---|
| | 30×5 | | | 30×10 | | |
| MOMBO vs. h-MOEA | 0.0698 = | 1.2567E-5 + | 6.712E-6 + | 0.0564 = | 0.1941 = | 0.3313= |
| MOMBO vs. m-MOEA/D | 0.4569 - | 0.1147 = | 0.0647 = | 0.0365 - | 0.0112 = | 0.0497 = |
| MOMBO vs. REMO | 5.2356e-5 + | 3.3411E-3 + | 5.4761E-5 + | 2.1364e-2 = | 2.1743E-3 + | 3.1597E-6 + |
| | 30×20 | | | 50×5 | | |
| MOMBO vs. h-MOEA | 4.2651E-5 + | 3.7411E-7 + | 2.5714E-8 + | 2.3659E-10 + | 8.2147E-9 + | 6.4712E-5 + |
| MOMBO vs. m-MOEA/D | 2.5698E-6 + | 6.3171E-7 + | 5.4751E-5 + | 4.5698E-9 + | 2.4711E-3 + | 1.2843E-10 + |
| MOMBO vs. REMO | 8.2365E-10 + | 1.6197E-10 + | 6.5712E-4 + | 1.4721E-7 + | 5.9167E-8 + | 9.3714E-5 + |
| | 50×10 | | | 50×20 | | |
| MOMBO vs. h-MOEA | 2.6541E-7 + | 6.4192E-6 + | 1.8974E-6 + | 1.8457E-12 + | 1.7149E-5 + | 9.1754E-5 + |
| MOMBO vs. m-MOEA/D | 6.5347E-8 + | 7.1311E-8 + | 9.3784E-10 + | 8.2364E-11 + | 3.9417E-9 + | 6.5714E-10 + |
| MOMBO vs. REMO | 4.1897E-12 + | 3.7911E-10 + | 4.5877E-5 + | 4.6574E-9 + | 8.9374E-4 + | 4.7164E-9 + |
| | 70×5 | | | 70×10 | | |
| MOMBO vs. h-MOEA | 9.2658E-10 + | 2.9547E-2 = | 0.2714 = | 0.0148 - | 0.0921 = | 0.5971 = |
| MOMBO vs. m-MOEA/D | 2.1597E-7 + | 2.1457E-11 + | 3.6174E-4 + | 2.1479E-6 + | 2.7791E-6 + | 5.6419E-5 + |
| MOMBO vs. REMO | 3.5743E-7 + | 6.3741E-9 + | 8.1687E-5 + | 6.3741E-10 + | 3.4178E-5 + | 7.5874E-5 + |
| | 70×20 | | | 90×5 | | |
| MOMBO vs. h-MOEA | 7.1451E-9 + | 9.5621E-7 + | 4.3671E-8 + | 0.0314 + | 3.6741E-10 + | 6.2231E-4 + |
| MOMBO vs. m-MOEA/D | 5.1114E-5 + | 3.1475E-10 + | 6.3417E-10 + | 2.1457E-3 + | 8.1143E-4 + | 1.9843E-5 + |
| MOMBO vs. REMO | 0.0061 = | 6.6741E-5 + | 9.1136E-5 + | 0.0174 - | 0.0792 = | 0.6143 = |
| | 90×10 | | | 90×20 | | |
| MOMBO vs. h-MOEA | 1.2794E-10 + | 7.1547E-9 + | 5.7168E-8 + | 2.3741E-13 + | 6.2517E-10 + | 7.3321E-9 + |
| MOMBO vs. m-MOEA/D | 7.8514E-11 + | 3.9147E-7 + | 1.9574E-5 + | 3.4712E-7 + | 4.9173E-6 + | 6.9173E-5 + |
| MOMBO vs. REMO | 2.4131E-9 = | 8.1741E-4 + | 6.9247E-5 + | 0.1274 = | 1.7713E-2 + | 8.6457E-6 + |
| | 110×5 | | | 110×10 | | |
| MOMBO vs. h-MOEA | 0.0891 = | 2.1962E-5 + | 6.6411E-6 + | 5.4197E-3 + | 3.4171E-4 + | 6.7742E-6 + |
| MOMBO vs. m-MOEA/D | 0.0023 + | 0.7146 = | 0.6191 = | 0.1798 = | 0.0354 = | 0.3571 = |
| MOMBO vs. REMO | 0.1759 = | 0.0611 = | 8.6987E-5 + | 0.0867 = | 0.3141 = | 0.6811 = |
| | 110×20 | | | 200×10 | | |
| MOMBO vs. h-MOEA | 6.1745E-11 + | 3.1411E-7 + | 2.7194E-10 + | 9.1476E-12 + | 5.3746E-6 + | 6.9173E-4 + |
| MOMBO vs. m-MOEA/D | 7.9147E-10 + | 5.3798E-10 + | 3.6917E-6 + | 3.8751E-10 + | 6.4751E-7 + | 7.1952E-9 + |
| MOMBO vs. REMO | 9.4873E-5 + | 4.9982E-6 + | 2.9978E-10 + | 1.1148E-8 + | 1.1411E-10 + | 5.3674E-5 + |
| | 200×20 | | | 500×20 | | |
| MOMBO vs. h-MOEA | 6.7512E-4 + | 4.6987E-10 + | 6.9514E-9 + | 4.8197E-3 + | 2.5889E-6 + | 9.9147E-10 + |
| MOMBO vs. m-MOEA/D | 5.2714E-13 + | 9.7614E-6 + | 2.6777E-6 + | 5.6184E-8 + | 4.6741E-10 + | 2.6974E-10 + |
| MOMBO vs. REMO | 5.1376E-8 + | 6.8427E-9 + | 1.6654E-10 + | 7.9146E-5 + | 1.9871E-11 + | 6.1112E-6 + |

from 1 to 50 s with the step size of 1 s on the aforementioned PC.

From Fig. 5, we can draw the conclusion that HV the convergence performance of MOMBO reaches the best on these four instances as the run time increases, suggesting that the proposed algorithm can guide the population towards the true Pareto-optimal front as the run time increases due to the fact that the proposed algorithm explicitly takes advantage of information from non-dominated solutions and the proposed local search can lead solution close to the ideal point.

In addition, Fig. 6 lists the final non-dominated fronts obtained by all the compared algorithms, respectively, on four instances, i.e., 30×5, 50×5, 90×10, and 200×20. For MOMBO, almost all the solutions of the best found non-dominated front are dominated ones obtained by

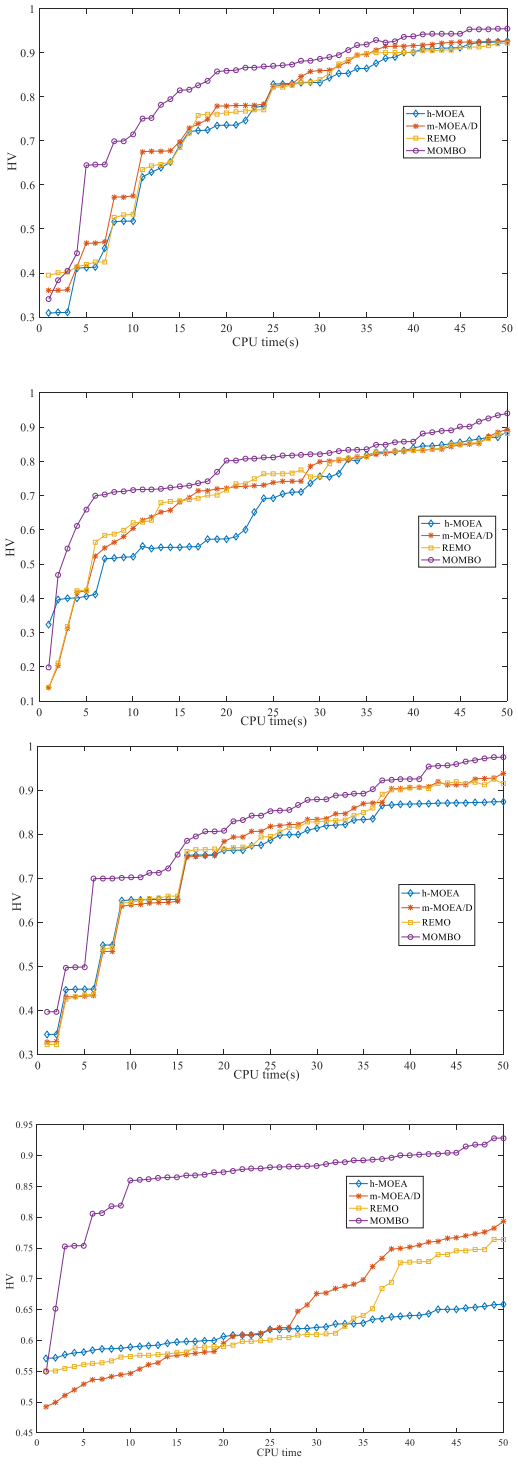



**FIGURE 5.** Changes of HV-U and HV-D over time on instances 30×5, 50×5 90×10 and 2000×20.

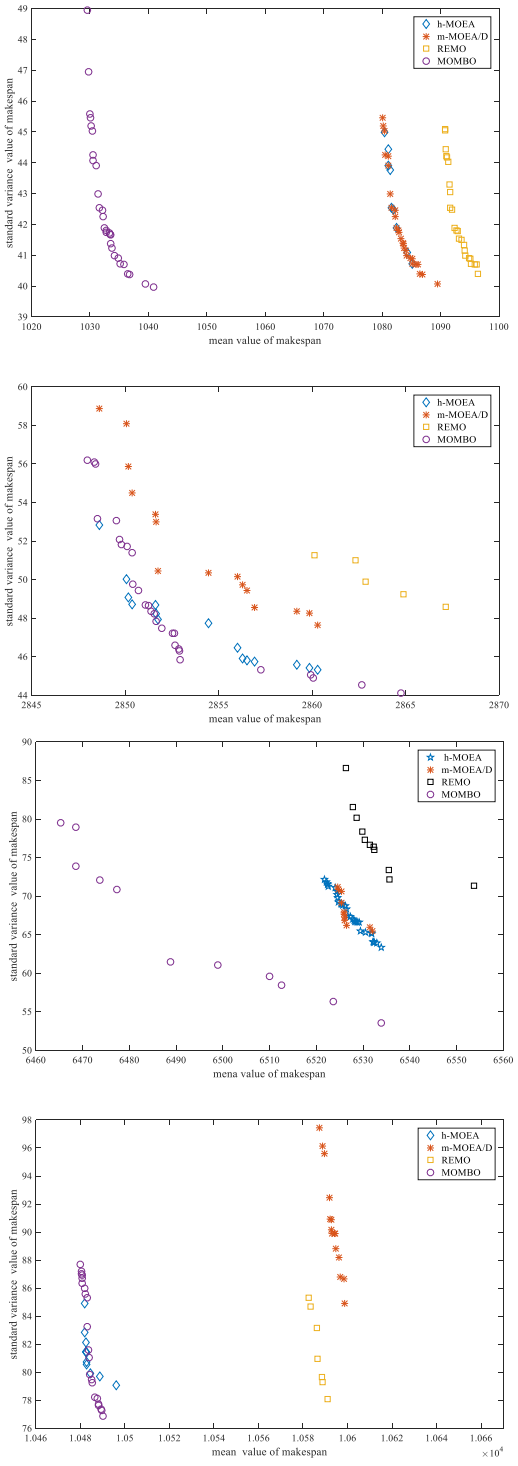



**FIGURE 6.** Pareto fronts for test instances with sizes 30×5, 50×5 90×10 and 200×20.

h-MOEA, m-MOEA/D and REMO, suggesting that the proposed algorithm has the capability to gradually guiding the population toward the true Pareto-optimal front. A possible reason is that the MOMBO takes full use of the valuable information provided by the non-dominated solutions.

## VII. CONCLUSION

This paper proposed a multi-objective migrating birds optimization algorithm to solve a stochastic blocking lot-streaming flow shop scheduling problem that have two important conflicting objective functions including variance and mean values of makespan. In order to perform exploration for

promising solutions within the entire solution space, the proposed algorithm incorporate an effective population initialization approach, a simple but effective an ideal-point assisted local search strategy and estimation of distribution ideal. Computational experiments are given and compared with the results yielded by the existing h-MOEA, m-MOEA/D and REMO algorithms.

There are several opportunities for future research on BLSFS scheduling problems with stochastic processing time, such as reducing the computational complexity of local search, designing an adaptive mechanism for selecting to improve the capability in exploitation. In addition, other types of uncertainties, such as the machine breakdowns, nondeterministic processing time, the operator illness and the change of the due date can also be considered.

## REFERENCES


[1] I. Ribas, R. Companys, and X. Tort-Martorell, "Efficient heuristics for the parallel blocking flow shop scheduling problem," *Expert Syst. Appl.*, vol. 74, pp. 41–54, May 2017.

[2] J.-Q. Li, P. Duan, J. Cao, X.-P. Lin, and Y.-Y. Han, "A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria," *IEEE Access*, vol. 6, pp. 58883–58897, 2018.

[3] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-G. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Cleaner Prod.*, vol. 181, pp. 584–598, Apr. 2018.

[4] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl.-Based Syst.*, vol. 148, pp. 115–130, May 2018.

[5] Y.-Y. Han, D. Gong, and X. Sun, "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking," *Eng. Optim.*, vol. 47, no. 7, pp. 927–946, 2015.

[6] Y.-Y. Han, J. J. Liang, Q.-K. Pan, J.-Q. Li, H.-Y. Sang, and N. N. Cao, "Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem," *Int. J. Adv. Manuf. Technol.*, vol. 67, nos. 1–4, pp. 397–414, 2013.

[7] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018, dio: 10.1109/ACCESS.2018.2840512.

[8] Y. N. Guo, J. J. Jiao, J. H. Ji, D. W. Gong, and J. Cheng, "Firework-based software project scheduling method considering the learning and forgetting effect," *Soft Comput.*, to be published, doi: 10.1007/s00500-018-3165-2.

[9] R. S. Patwal, N. Narang, and H. Garg, "A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units," *Energy*, vol. 142, pp. 822–837, Jan. 2018.

[10] H. Garg, "Performance analysis of an industrial system using soft computing based hybridized technique," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 39, no. 4, pp. 1441–1451, 2017.

[11] H. Garg, "A hybrid PSO-GA algorithm for constrained optimization problems," *Appl. Math. Comput.*, vol. 274, pp. 292–305, Feb. 2016.

[12] P.-Y. Duan, J.-Q. Li, Y. Wang, H.-Y. Sang, and B.-X. Jia, "Solving chiller loading optimization problems using an improved teaching-learning-based optimization algorithm," *Optim. Control Appl. Methods*, vol. 39, no. 1, pp. 65–77, 2018.

[13] H. Garg and S. P. Sharma, "Multi-objective reliability-redundancy allocation problem using particle swarm optimization," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 247–255, 2013.

[14] H. Garg, "An efficient biogeography based optimization algorithm for solving reliability optimization problems," *Swarm Evol. Comput.*, vol. 24, pp. 1–10, Oct. 2015.

[15] H. Garg, "Solving structural engineering design optimization problems using an artificial bee colony algorithm," *J. Ind. Manage. Optim.*, vol. 10, no. 3, pp. 777–794, 2014.

[16] Z.-X. Zheng, J.-Q. Li, and P.-Y. Duan, "Optimal chiller loading by improved artificial fish swarm algorithm for energy saving," *Math. Comput. Simul.*, vol. 155, pp. 227–243, Jan. 2019.

[17] Y.-Y. Han, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved artificial bee colony algorithm for the blocking flowshop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 60, nos. 9–12, pp. 1149–1159, 2012.

[18] H.-Y. Sang, P.-Y. Duan, and J.-Q. Li, "An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem," *Swarm Evol. Comput.*, vol. 38, pp. 42–53, Feb. 2018.

[19] T. Meng, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 38, pp. 64–78, Feb. 2018.

[20] B. Zhang, Q.-K. Pan, L. Gao, X.-L. Zhang, H.-Y. Sang, and J.-Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," *Appl. Soft Comput.*, vol. 52, pp. 14–27, Mar. 2017.

[21] Y. Han, D. Gong, Y. Jin, and Q.-K. Pan, "Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time," *Appl. Soft Comput.*, vol. 42, pp. 229–245, May 2016.

[22] D. Davendra, R. Senkerik, I. Zelinka, M. Pluhacek, and M. Bialic-Davendra, "Utilising the chaos-induced discrete self organising migrating algorithm to solve the lot-streaming flowshop scheduling problem with setup time," *Soft Comput.*, vol. 18, no. 4, pp. 669–681, 2014.

[23] H.-Y. Sang, Q.-K. Pan, P.-Y. Duan, and J.-Q. Li, "An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 6, pp. 1337–1349, 2018.

[24] G. V. Chakaravarthy *et al.*, "Improved sheep flock heredity algorithm and artificial bee colony algorithm for scheduling m-machine flow shops lot streaming with equal size sub-lot problems," *Int. J. Prod. Res.*, vol. 52, no. 5, pp. 1509–1527, 2014.

[25] F. M. Defersha and M. Chen, "A genetic algorithm for one -job *m*–machine flowshop lot streaming with variable sublots," *Int. J. Oper. Res.*, vol. 10, no. 4, pp. 458–468, 2011.

[26] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3252–3259, 2011.

[27] Y.-Y. Han, D.-W. Gong, X.-Y. Sun, and Q.-K. Pan, "An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 52, no. 8, pp. 2211–2231, 2014.

[28] Q.-K. Pan and R. Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times," *Omega*, vol. 40, no. 2, pp. 166–180, 2012.

[29] Y. Han *et al.*, "Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 184–197, 2019.

[30] X.-N. Shen, Y. Han, and J.-Z. Fu, "Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems," *Soft Comput.*, vol. 21, no. 21, pp. 6531–6554, 2017.

[31] J. Xiong, L.-N. Xing, and Y.-W. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 112–126, 2013.

[32] S. Shoval and M. Efatmaneshnik, "A probabilistic approach to the stochastic job-shop scheduling problem," *Procedia Manuf.*, vol. 21, pp. 533–540, Mar. 2018.

[33] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *J. Math. Anal. Appl.*, vol. 355, no. 1, pp. 63–81, 2009.

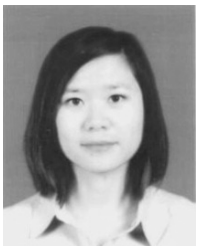
[34] Y. Fu, J. Ding, H. Wang, and J. Wang, "Two-objective stochastic flowshop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system," *Appl. Soft Comput.*, vol. 68, pp. 847–855, Jul. 2018.

[35] S. H. Choi and K. Wang, "Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach," *Comput. Ind. Eng.*, vol. 63, no. 2, pp. 362–373, 2012.

[36] Y.-K. Lin, D.-H. Huang, and C.-F. Huang, "Estimated network reliability evaluation for a stochastic flexible flow shop network with different types of jobs," *Comput. Ind. Eng.*, vol. 98, pp. 401–412, Aug. 2016.

[37] K. Nagasawa, Y. Ikeda, and T. Irohara, "Robust flow shop scheduling with random processing times for reduction of peak power consumption," *Simul. Model. Pract. Theory*, vol. 59, pp. 102–113, Dec. 2015.

[38] D.-M. Lei, "Minimizing makespan for scheduling stochastic job shop with random breakdown," *Appl. Math. Comput.*, vol. 218, no. 24, pp. 11851–11858, 2012.
[39] Y. Fu, H. Wang, G. Tian, Z. Li, and H. Hu, "Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm," *J. Intell. Manuf.*, to be published, doi: 10.1007/s1084.
[40] C. Almeder and R. F. Hartl, "A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 88–95, 2013.
[41] M. Ćwik and J. Józefczyk, "Heuristic algorithms for the minmax regret flow-shop problem with interval processing times," *Central Eur. J. Oper. Res.*, vol. 26, no. 1, pp. 215–238, 2018.
[42] A. Allahverdi, H. Aydilek, and A. Aydilek, "Single machine scheduling problem with interval processing times to minimize mean weighted completion time," *Comput. Oper. Res.*, vol. 51, pp. 200–207, Nov. 2014.
[43] X. Feng, F. Zheng, and Y. Xu, "Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times," *Int. J. Prod. Res.*, vol. 54, no. 12, pp. 3706–3717, 2016.
[44] H. Mokhtari and A. Salmasnia, "A Monte Carlo simulation based chaotic differential evolution algorithm for scheduling a stochastic parallel processor system," *Expert Syst. Appl.*, vol. 42, no. 20, pp. 7132–7147, 2015.
[45] A. A. Juan, B. B. Barrios, E. Vallada, D. Riera, and J. Jorba, "A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times," *Simul. Model. Pract. Theory*, vol. 46, no. 4, pp. 101–117, 2014.
[46] S. Asta, D. Karapetyan, A. Kheiri, E. Özcan, and A. J. Parkes, "Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem," *Inf. Sci.*, vol. 373, pp. 476–498, Dec. 2016.
[47] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: A new metaheuristic approach and its application to the quadratic assignment problem," *Inf. Sci.*, vol. 217, pp. 254–263, Apr. 2011.
[48] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem," *Inf. Sci.*, vol. 217, no. 24, pp. 65–77, Dec. 2012.
[49] L. W. Shen, H. Asmuni, and F. C. Weng, "A Modified migrating bird optimization for University course timetabling problem," *J. Teknologi*, vol. 72, no. 1, pp. 89–96, 2015.
[50] A. F. Alkaya, R. Algin, Y. Sahin, M. Agaoglu, and V. Aksakalli, "Performance of migrating birds optimization algorithm on continuous functions," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2014, pp. 452–459.
[51] V. Tongur and E. Ülker, "Migrating birds optimization for flow shop sequencing problem," *J. Comput. Commun.*, vol. 2, no. 4, pp. 142–147, 2014.
[52] Q.-K. Pan and Y. Dong, "An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation," *Inf. Sci.*, vol. 277, no. 2, pp. 643–655, Sep. 2014.
[53] S. Niroomand, A. Hadi-Vencheh, R. Şahine, and B. Vizvári, "Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems," *Expert Syst. Appl.*, vol. 42, no. 19, pp. 6586–6597, 2015.
[54] D. P. Ronconi, "A note on constructive heuristics for the flowshop problem with blocking," *Int. J. Prod. Econ.*, vol. 87, no. 1, pp. 39–48, 2004.
[55] Q.-K. Pan and L. Wang, "Effective heuristics for the blocking flowshop scheduling problem with makespan minimization," *Omega*, vol. 40, no. 2, pp. 218–229, 2012.
[56] K. Sindhya, A. Sinha, K. Deb, and K. Miettinen, "Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems," in *Proc. 11th IEEE Congr. Evol. Comput.*, Piscataway, NJ, USA, May 2009, pp. 2919–2926.
[57] Y. P. Liu, G. G. Yen, and D. W. Gong, "A multi-modal multi-objective evolutionary algorithm using two-archive and recombination strategies," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2018.2879406.
[58] S.-H. Yoon and J. A. Ventura, "An application of genetic algorithms to lot-streaming flow shop scheduling," *IIE Trans.*, vol. 34, no. 9, pp. 779–787, 2002.
[59] C.-T. Tseng and C.-J. Liao, "A discrete particle swarm optimization for lot-streaming flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 191, no. 2, pp. 360–373, 2008.



**YUYAN HAN** received the M.S. degree from the School of Computer Science, Liaocheng University, China, in 2012, and the Ph.D. in control theory and control engineering from the China University of Mining and Technology, China, in 2016. Since 2016, she has been a Lecturer with the School of Computer Science, Liaocheng University. She has authored more than 20 refereed papers. Her current research interests include evolutionary computation, multi-objective optimization, and flow shop scheduling.

**JUN-QING LI** received the master's degree in computer science and technology from Shandong Economic University, Shandong, China, in 2004, and the Ph.D. degree from Northeastern University, Shenyang, China, in 2016. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, where he became a Professor, in 2017. He has authored more than 40 refereed papers. His current research interests include intelligent optimization and scheduling.

**DUNWEI GONG** received the B.S. degree in applied mathematics from the China University of Mining and Technology, China, in 1992, the M.S. degree in control theory and its applications from Beihang University, China, in 1995, and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, in 1999. Since 2004, he has been a Professor with the China University of Mining and Technology. He is also a Professor with the Qingdao University of Science and Technology, China. He has authored more than 100 refereed papers. His main research interests include evolutionary computation and search-based software engineering.

**HONGYAN SANG** received the M.S. degree from the School of Computer Science, Liaocheng University, China, in 2009, and the Ph.D. degree from the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, in 2013.

She has been an Associate Professor with the School of Computer Science, Liaocheng University. She has authored more than 20 refereed papers. Her current research interests include evolutionary computation, multi-objective optimization, and flow shop scheduling.