Engineering Applications of Artificial Intelligence 116 (2022) 105443

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A heuristic and meta-heuristic based on problem-specific knowledge for distributed blocking flow-shop scheduling problem with sequence-dependent setup times

Fuqing Zhao [a,*], Haizhu Bao [a], Ling Wang [b], Tianpeng Xu [a], Ningning Zhu [a], Jonrinaldi [c]

[a] *School of Computer and Communication, Lanzhou University of Technology, Lanzhou, 730050, China*
[b] *Department of Automation, Tsinghua University, Beijing, 10084, China*
[c] *Department of Industrial Engineering, Universitas Andalas, Padang, 25163, Indonesia*

ARTICLE INFO

Dataset link: https://github.com/banian2314/MBIST-TSM

Keywords:
Distributed scheduling
Blocking flow-shop
Sequence-dependent setup times
Meta-heuristic
Heuristic

ABSTRACT

The distributed production scenario with the sequence-dependent setup times (SDST) widely exists in the modern manufacturing system. This paper investigates the distributed blocking flow-shop scheduling problem with sequence-dependent setup times (SDST/DBFSP). Considering the complexity of the distributed scenario and SDSTs, a discrete heuristic and meta-heuristic is proposed by exploring the problem-specific knowledge. First, a knowledge-incorporated construction heuristic is proposed to reduce the blocking times and idle times generated by SDSTs. In the first stage of the meta-heuristic, an insertion-based neighborhood operator of different factories is developed to explore promising regions in the decision space. In the second stage, a local search operator is embedded to enhance the exploitation ability. Additionally, a simulated annealing-like acceptance criterion of the iterated greedy algorithm is employed to keep the diversity of the population. Finally, an insertion operation for critical factories is introduced to further improve the accuracy of the solutions. Moreover, a speedup method for the insertion neighborhood is expanded to reduce the computational complexity of SDST/DBFSP. In the part of the experiment, a deconstruction process is designed to gain insight into the contribution of each component in the proposed meta-heuristic. The proposed meta-heuristic is assessed through comparing with five state-of-the-art algorithms to demonstrate its effectiveness. The experimental results testified that the proposed meta-heuristic outperforms other algorithms regarding the significance of the SDST/DBFSP.

1. Introduction

Production scheduling plays a significant role in the decision-making and planning of the intelligent manufacturing system (He et al., 2021; Wang et al., 2022; Zhen et al., 2021). The mode of production scheduling has gradually turned to the distributed manufacturing model of the multi-factory and multi-supply chains with the increasing trend of economic globalization and world economic integration (Cheng et al., 2021). The blocking flow-shop scheduling problem (BFSP) (Han et al., 2019) widely exists in the manufacturing and production systems, such as the non-ferrous metallurgical industry (Chen et al., 2019), chemical industry (Fu et al., 2020), and serial manufacturing processes (Jing et al., 2021). In recent years, the distributed flow-shop scheduling problem (DFSP) is considerably popular to effectively reduce costs and risks via the cooperation of factories, which shortens the production time and improved production efficiency (Zhao et al., 2020a). Compared with BFSP which scheduled the jobs in a single factory, the excess difficulty of the distributed blocking flow-shop scheduling problem (DBFSP) is to assign the jobs in certain isomorphic factories. Existing experiments and literature have shown that the BFSP with more than two machines is a typical NP-hard problem (Zhao et al., 2020c).

In the actual production process, certain unproductive operations are often added between two continuous operations, such as machine cleaning, tool replacement, and job transportation (Huang et al., 2020). These unproductive operations are important conditions for the smooth processing of the jobs. The duration of these operations is closely related to the current job being processed and the last job that has been processed. These durations are called the sequence-dependent setup times (SDSTs) since different processing sequences produce different durations (Shao et al., 2018). The setup times are included in processing time in the majority of the existing DFSPs. However, substantial setup

* Corresponding author.
E-mail addresses: Fzhao2000@hotmail.com (F. Zhao), banian2314@outlook.com (H. Bao), wangling@tsinghua.edu.cn (L. Wang), xutp@lut.edu.cn (T. Xu), 307516638@qq.com (N. Zhu), jonrinaldi@eng.unand.ac.id (Jonrinaldi).

https://doi.org/10.1016/j.engappai.2022.105443
Received 6 April 2022; Received in revised form 17 August 2022; Accepted 6 September 2022
Available online xxxx

times are required between the processing of two jobs in various industrial production systems. The effective processing of setup operation is critical to improving the efficiency of the manufacturing system. Hence, the setup times in the shop schedule are considered independently of processing times in this paper. The SDSTs are contained in 50% of the manufacturing plants. Further, 92% of the order deadlines are affected by these SDSTs (Huang et al., 2020). Hence, it is of great significance to investigate this kind of scheduling problem for actual manufacturing. In this paper, the distributed blocking flow-shop scheduling problem with sequence-dependent setup times (SDST/DBFSP) is considered to minimize the maximum completion time of all the jobs.

The SDST/DBFSP is denoted as $DF|blocking, s_{ijk}|C_{max}$ according to the well-known three-field notation $\alpha|\beta|\gamma$ proposed by Graham et al. (1979), where DF denotes the DBFSP, $blocking$ and s_{ijk} represent the blocking constraint and the SDST, C_{max} represents the maximum completion time of all the jobs (makespan). As the DBFSP is an extension of the SDST/DBFSP, the SDST/DBFSP is an NP-hard problem as well apparently. Traditional mathematical methods, for instance, integer programming (Bastos et al., 2019), branch and bound (Tomazella and Nagano, 2020), and dynamic programming (Liu et al., 2021) are efficient methods to solve certain small-scale scheduling problems. However, the methods have inferior performance to address the large-scale distributed scheduling problems with multiple machines, multiple jobs, and the SDSTs. The heuristic and metaheuristic approaches are among the best alternatives to discover optimal or near-optimal solutions for scheduling problems with the makespan criterion in reasonable computational time (Li et al., 2020). In particular, the combination of swarm intelligence and heuristics to solve this kind of scheduling problem is a hot topic in both domestic and foreign research fields in recent years (Pan et al., 2020; Xin et al., 2021). Nevertheless, to the author's knowledge, the SDST/DBFSP has not been considered so far. In this paper, a heuristic and three-stage meta-heuristic (MBIST-TSM) (MBIST represents a constructive heuristic and TSM represents a three-stage meta-heuristic), is proposed to solve the SDST/DBFSP. The contributions of this paper are summarized as follows:

(1) The problem-specific knowledge of the SDST/DBFSP is explored.
(2) A heuristic that considers the problem-specific knowledge is proposed, namely, minimizing the blocking times and idle times created by SDSTs (MBIST).
(3) A three-stage meta-heuristic that incorporating the knowledge with the search process of the algorithm is investigated, which can enhance the searching ability.
(4) The calculation methods for the makespan of SDST/DBFSP are presented. Meanwhile, a speedup method for evaluating the insert neighborhood is proposed to reduce the complexity of evaluating the entire insertion neighborhood.

The remainder of this paper is structured as follows. In Section 2, the closely related literature is reviewed. The considered SDST/DBFSP is described in Section 3. The speedup method for the insert neighborhood structure is proposed in Section 4. In Section 5, the proposed MBIST-TSM incorporated problem-specific knowledge is elaborated. The experimental results are implemented in Section 6. Finally, the conclusions and outlook for the future study are summarized in Section 7.

2. Literature review

The literature about the closely related problems including the standard DFSP, the DBFSP, and the DFSP with SDST (SDST/DFSP) are reviewed in this section. Naderi and Ruiz (2010) first studied the DFSP in 2010, in which the makespan was to be minimized and developed six types of the mixed-integer linear programming (MILP) models. Hatami et al. (2013) investigated the distributed assembly flow-shop scheduling problems in supplychain systems and solved them using a variable neighborhood descent (VND) algorithm. In recent years, many metaheuristics have been used to solve DFSPs, which can be classified into several categories according to the problem features. In DFSPs of the first type, the makespan is minimized. To solve problems of this type, researchers have utilized many metaheuristics, such as the iterated greedy (IG) algorithm (Fernandez-Viagas and Framinan, 2015; Lin et al., 2013; Ruiz et al., 2019); the scatter search (SS) method by Naderi and Ruiz (2014); the chemical reaction optimization (CRO) algorithm by Bargaoui et al. (2017); the tabu search algorithm (TS) by Gao et al. (2013); the improved artificial bee colony algorithm (IABC) by Li et al. (2019); the revised artificial immune system (RAIS) (Lin and Ying, 2013) and the estimation of the distribution algorithm (EDA) by Wang et al. (2013). In DFSPs of the second type, additional constraints are imposed. Li et al. (2020) proposed a hybrid artificial bee colony (ABC) algorithm to solve a parallel batching distributed flow-shop problem (DFSP) with deteriorating jobs. Wang et al. (2016) considered a DFSP with machine breakdown constraints. Rifai et al. (2016) solved distributed re-entrant permutation flow-shop scheduling. Lin and Ying (2016) considered a distributed no-wait flow-shop scheduling problem. Ying et al. (2017) studied a no-idle DFSP. Deng and Wang (2017) applied a competitive memetic algorithm to a multi-objective DFSP. Wang and Wang (2020) studied an energy-efficient scheduling of DFSP by a knowledge-based cooperative algorithm (KCA). In addition, there are some excellent review articles, such as Framinan et al. (2019) etc.

The DBFSP has been widely investigated in recent years. Many types of metaheuristics have been proposed for and applied to problems of this type. Ribas et al. (2017) presented a mathematical model for DBFSP along with some constructive heuristics and meta-heuristics, which is also the first report about the DBFSP. Following this work, Ying and Lin (2017) presented three hybrid iterated greedy algorithms. Zhang et al. (2018) proposed a discrete differential evolution (DDE) algorithm as well as four constructive heuristics. Ribas et al. (2019) also investigated DBFSP with total tardiness criterion and proposed an IG algorithm. A hybrid enhanced discrete fruit fly optimization algorithm (DFOA) was proposed (Shao et al., 2020a). The authors developed an extended HPF2 (EHPF2) heuristic, which includes an improved assignment regulation and an insertion-based improvement procedure. The same authors proposed two IG methods to solve a special case of DBFSP that contained an uncertain processing time (Shao et al., 2020b). Zhao et al. (2020c) developed an ensemble discrete differential evolution (EDE) algorithm, which integrated two heuristics methods and a random strategy. Zhao et al. (2022a) proposed a water wave optimization algorithm with problem-specific knowledge (KWWO) for the distributed assembly blocking flow-shop scheduling problem.

For the DFSPs, the SDSTs have been investigated for practical applications (Huang et al., 2021; Rossi and Nagano, 2021). Shao et al. (2018) proposed a discrete water wave optimization algorithm (DWWO). A path relinking technique was cleverly applied in the refraction operation of WWO, which realizes the mapping from continuous space to discrete space. Takano and Nagano (2019) proposed 14 best-known heuristics for the permutation flow-shop problem with blocking and no setup times and adapted them to the SDST/BFSP in two different ways. Newton et al. (2019) developed a constructive heuristic taking both blocking constraints and setup times into account in PFSP. The authors proposed the reinforcement methods of constraint guidance and random path guidance to improve the diversity of the algorithm. Han et al. (2020) constructed a multi-objective optimization model of the SDST/BFSP with the criterion of makespan and energy. The authors introduced a penalty-based boundary interstation to improve the exploitation. An extended IG (EIG) algorithm was proposed to solve the SDST/BFSP by Cheng et al. (2021). In the EIG algorithm, a local search mechanism based on the features of blocking and setup time was developed to enhance the exploitation. Ribas and Companys (2021) developed a combined heuristic to address the SDST/BFSP. Riahi et al. (2021) proposed a constraint based local search (CBLS) algorithm to transform the SDST constraints into an auxiliary objective function. The heuristic showed promising performance in a short CPU time. Rossi and

Table 1
Review of representative work for the distributed production scheduling.

Problem	Objectives	Methods
DFSPs	C_{max}	14 heuristics and VND (Naderi and Ruiz, 2010), VND (Hatami et al., 2013), IG (Fernandez-Viagas and Framinan, 2015; Lin and Ying, 2013; Ruiz et al., 2019), SS (Naderi and Ruiz, 2014), CRO (Bargaoui et al., 2017), TS (Gao et al., 2013), IABC (Li et al., 2019), RAIS (Lin and Ying, 2013), EDA (Wang et al., 2013), Fuzzy logic-based hybrid estimation of distribution algorithm (Wang et al., 2016), Iterated cocktail greedy (Lin and Ying, 2016), Iterated reference greedy (Ying et al., 2017), Hybrid artificial bee colony algorithm (Li et al., 2020)
	C_{max}, TUC, $\overline{T}$	Multi-objective adaptive large neighborhood search (Rifai et al., 2016)
	C_{max}, $\sum T_i$	Competitive memetic algorithm (Deng and Wang, 2017)
	C_{max}, TEC	KCA (Wang and Wang, 2020)
DBFSP	C_{max}	Iterated local search (ILS) and IG algorithm (Ribas et al., 2017), DDE (Zhang et al., 2018), Hybrid iterated greedy algorithm (HIG) (Ying and Lin, 2017), DFOA and EHPF2 (Shao et al., 2020a), IG (Shao et al., 2020b), EDE (Zhao et al., 2020c)
	$\sum T_i$	IG (Ribas et al., 2019), KWWO (Zhao et al., 2022a,b)
SDST/DFSP	C_{max}	DWWO (Shao et al., 2018), 14 heuristics (Takano and Nagano, 2019), Constraint guided local search (Newton et al., 2019), CBLS (Riahi et al., 2021), Constructive heuristics (Ribas and Companys, 2021), IG-RN (Rossi and Nagano, 2021), DABC (Huang et al., 2021), IG_NM (Miyata and Nagano, 2022), IG_RCM (Ribas and Companys, 2021)
	C_{max}, TEC	Discrete evolutionary multi-objective optimization algorithm (Han et al., 2020)
	$\sum C_i$	EIG (Cheng et al., 2021)

Note: TUC is the total production cost. $\overline{T}$ is average tardiness. $\sum T_i$ is the total tardiness. TEC is total energy consumption. $\sum C_i$ is the total completion time.

Nagano (2021) proposed a new IG based on IG2S, denoted as IG-RN, to solve the mixed no-idle distributed permutation flow-shop scheduling problem with SDSTs. Huang et al. (2021) developed a discrete artificial bee colony (DABC) algorithm for the distributed permutation flowshop scheduling problem with SDSTs. Miyata and Nagano (2022) investigated a DBFSP with SDSTs and maintenance operations. The authors proposed an IG with variable search neighborhood (IG_NM). Ribas et al. (2021) developed an efficient IG algorithm (IG_RCM) for the parallel flow-shop scheduling problem with SDSTs. In addition, there is also a very important the customer order scheduling problem with sequence-dependent setup times, which has been widely studied in recent years (Prata et al., 2022, 2021).

The literature about the DBFSP and the related problem with different constraints have been described above, a summary of which is provided in Table 1. In the table, we list all the proposed algorithms. It can be seen from the above works that the advantages of these state-of-the-art methods for solving the standard DFSP, the DBFSP, and the SDST/DFSP can be attributed to a combination of constructive heuristics and local search. The construction heuristic methods were employed to quickly produce certain solutions with good performance as the initial sequence. After that, the metaheuristics were used to further optimize the solution. In this process, the idea of swarm intelligence was adopted to improve the ability of exploitation. The search direction was enriched through multiple individuals effectively. However, it costs a lot of time to evolve the whole population through different evolutionary operators. On the other hand, the incorporation of problem-specific knowledge was proved to be an effective way to guide the search of metaheuristics, avoid useless search, and enable metaheuristics to be explored in promising regions (He et al., 2021; Naik et al., 2020).

3. Problem statement

The problem under consideration can be described as follows: A set of n jobs, $J = \{J_1, J_2, \ldots, J_n\}$, have to be processed by a set of f identical factories, $F = \{F_1, F_2, \ldots, F_f\}$, each of which is a flow-shop consisting of a set of m machines, $M = \{M_1, M_2, \ldots, M_m\}$. A job J_j ($J_j \in J$) is processed in any one of the f factories. In each factory, the jobs are processed following the same production route that starts from the first machine and ends at the last machine. The setup times are separable from the processing times. Further, the value of the setup times is sequence dependent. As with the classic BFSP, there are no intermediate buffers between consecutive machines. Meanwhile, the machine must take some time to prepare before processing the next job considering the characteristics of the problem. Therefore, the job did not leave the current machine after completion of its operation until the next machine is free and prepared for processing. The blocking times are generated by the constraint. The other common flow-shop constraints are also considered:

(a) All jobs are independent and released at time zero;
(b) each job is only processed on one machine at the same time;
(c) each machine processes only one job at the same time;
(d) the processing time of each job on any machine is determined;
(e) pre-emption is not allowed.

In this article, we consider a typical process that is carried out in many industrial processes, such as aluminum production process.

3.1. Problem formulation

The definitions of the variables and parameters mentioned are recorded before describing mathematical models as follows.

n	The number of jobs.
m	The number of machines.
f	The number of factories.
i	The index of the job, $i = 0, 1, 2, \ldots, n$. 0 indicates a dummy job.
j	The index of the machine, $j = 0, 1, 2, \ldots, m$. 0 indicates a virtual machine.
l	The index of the factory, $l = 1, 2, \ldots, f$.
k	The index of a certain job in the job sequence, $k \in \{1, 2, \ldots, n\}$.
h	The index of a certain job in the job sequence, $h \in \{1, 2, \ldots, n\}$.
n_l	The number of jobs in factory l.
π_l	The processing sequence in factory l, $\pi_l = \{\pi_l(1), \pi_l(2), \ldots, \pi_l(n_l)\}$.
$\prod$	A feasible scheduling job sequence, $\prod = \{\pi_1, \pi_2, \ldots, \pi_f\}$.
NP	The size of the population.
s	The index of the solution in POP, $s = 1, 2, \ldots, NP$.
$\prod_s$	A feasible scheduling job sequence, $\prod_s = \{\pi_1; \pi_2; \ldots; \pi_l; \ldots; \pi_f\}$.
POP	The population contains NP solutions, $POP = \{\prod_1; \prod_2; \ldots; \prod_s; \ldots; \prod_{NP}\}$

$\prod^{best}$	The best solution in POP.
$\prod^{worst}$	The worst solution in POP.
$p_{\pi_l(i),j}$	The processing time of job i on machine j.
$s_{\pi_l(h),\pi_l(i),j}$	The setup time of adjacent job h and job i on machine j.
$D_{l,\pi_l(i),0}$	The start processing time of ith job on the first machine in lth factory.
$D_{l,\pi_l(i),j}$	The departure time of ith job on machine j in lth factory.
$Q_{l,\pi_l(i),0}$	The reverse start processing time of ith job on the first machine in lth factory.
$Q_{l,\pi_l(i),j}$	The reverse departure time of ith job on machine j in lth factory.
$x_{i,l,k}$	The binary variable that takes value 1 if the kth job of π_l is i, and 0 otherwise.
$y_{h,i,l,k}$	The binary variable that takes value 1 if the kth job of π_l is i and the hth job is the precursor job of the kth job, and 0 otherwise.
C_{max}	The maximum completion time.
J	The job sequence, $J = \{J_1, J_2, \dots, J_n\}$.
M	The machine sequence, $M = \{M_1, M_2, \dots, M_m\}$.
F	The factory sequence, $F = \{F_1, F_2, \dots, F_f\}$.

The details of the mathematical model of DBFSP with SDST are described as follows.

$$x_{i,l,k} \in \{0,1\}, \forall i,k \in \{1,2,\dots,n\}, \forall l \in \{1,2,\dots,f\} \tag{1}$$

$$y_{h,i,l,k} \in \{0,1\}, \forall i,k \in \{1,2,\dots,n\}, \forall l \in \{1,2,\dots,f\}, \forall h \in 0,1,\dots,n-1 \tag{2}$$

$$\text{Min}\, C_{max} = \max_{\substack{k=1,2,\dots,n\\ l=1,2,\dots,f}} \left(D_{l,\pi_l(k),m}\right) \tag{3}$$

$$\sum_{l=1}^{f}\sum_{k=1}^{n} x_{i,l,k} = 1, \forall i \in \{1,2,\dots,n\} \tag{4}$$

$$\sum_{i=1}^{n} x_{i,l,k} \le 1, \forall k \in \{1,2,\dots,n\}, \forall l \in \{1,2,\dots,f\} \tag{5}$$

$$x_{0,l,0} = 1, \forall l \in \{1,2,\dots,f\} \tag{6}$$

$$x_{i,l,k} = \sum_{h=0}^{n} y_{h,i,l,k}, \forall i,k \in \{1,2,\dots,n\}, \forall l \in \{1,2,\dots,f\} \tag{7}$$

$$x_{i,l,k} = \sum_{h=0}^{n} y_{h,i,l,k+1}, \forall i \in \{1,2,\dots,n\}, \forall k \in \{1,2,\dots,n-1\}, \forall l \in \{1,2,\dots,f\} \tag{8}$$

$$D_{l,\pi_l(0),j} \ge 0, \forall j \in \{1,2,\dots,m\}, \forall l \in \{1,2,\dots,f\} \tag{9}$$

$$D_{l,\pi_l(k),0} \ge D_{l,\pi_l(k-1),1} + \sum_{i=1}^{n}\sum_{h=0}^{n} y_{h,i,l,k} \cdot s_{\pi_l(h),\pi_l(i),1}, \forall k \in \{1,2,\dots,n\}, \forall l \in 1,2,\dots,f \tag{10}$$

$$D_{l,\pi_l(k),j} \ge D_{l,\pi_l(k),j-1} + \sum_{i=1}^{n} x_{i,l,k} \cdot p_{\pi_l(i),\pi_l(j)}, \forall k \in \{1,2,\dots,n\}, \forall j \in \{1,2,\dots,m\}, \forall l \in \{1,2,\dots,f\} \tag{11}$$

$$D_{l,\pi_l(k),j} \ge D_{l,\pi_l(k-1),j+1} + \sum_{i=1}^{n}\sum_{h=0}^{n} y_{h,i,l,k} \cdot s_{\pi_l(h),\pi_l(i),j+1}, \forall k \in \{1,2,\dots,n\}, \forall j \in \{1,2,\dots,m-1\}, \forall l \in \{1,2,\dots,f\} \tag{12}$$

$$C_{max} \ge D_{l,\pi_l(n),m}, \forall l \in \{1,2,\dots,f\} \tag{13}$$

The decision variable (1) is a binary variable that value 1 if the job J_i is the kth job processed in the factory F_l, and 0 otherwise. The decision variable (2) is also a binary variable that value 1 if the job J_i is the kth job processed in the factory F_l and the job h is the precursor job of the kth job, and 0 otherwise. The objective function (3) is to minimize the makespan. Constraint (4) indicates each job can only be placed in one factory and only in one position in the factory. Constraint (5) ensures that at most one job can be allocated to each position of a factory. Constraint (6) shows that the virtual job of each factory appeared in position 0. Constraints (7) and (8) express that each job has and has only one precursor and subsequent job. Constraint (9) ensures that the start time of each job is not less than 0. Constraint (10) is the start time of each job on the first machine in each factory. Constraint (11) indicates the relationship between the departure time of two adjacent operations of the same job. Constraint (12) represents the relationship between the departure times of two adjacent jobs. Constraint (13) indicates the objective value is not less than the maximum value of the complete time of all jobs on the last machine. The objective of the SDST/DBFSP is to assign n jobs to f factories and determine the sequence within each factory so that the maximum completion time of all the jobs or the $C_{\max}$ is minimized.

3.2. Realistic problem example

In Western China, the development of nonferrous metallurgy is affected by raw material supply and the regional environment. Hence, factories have to be distributed in certain different places to reduce manufacturing costs and improve production efficiency. The development of the aluminum industry is a typical example. In many aluminum plants distributed in various places, there are many parallel electrolytic cells in the electrolytic workshop and parallel continuous casters in the smelter. An overall process of aluminum production is shown in Fig. 1. Pure aluminum is extracted from raw material (alumina) via the electrolysis cells and then cast into ingots by casters. Aluminum production is driven by customer orders, which means that customers first sign contracts with aluminum plants, and then convert them into production orders in the form of alloy composition, size, etc. Finally, the production order is arranged on the electrolytic cell and continuous casting machines. In the process of electrolysis and melting, each order is allocated to each processing cell according to the allocation rules due to the requirements of electrolytic cells and continuous casters. The orders at the same cell should be scheduled considering the possible setup times, which are mainly caused by the variation of the size and alloy composition of orders. The setup times are sequence-dependent in the light of setup forms such as adjustment of equipment and cleaning of cells, which are both expensive and time-consuming. It is found that the production structure of the orders is roughly the same, and the production is continuous. Hence, furnace and caster are a stage that simplified the problem. The problem is divided into two stages, electrolysis and casting. Certain orders are assigned to electrolytic cells in different regions and scheduled considering production constraints and SDSTs. After that, the electrolyzed pure molten aluminum is transferred to the casting and rolling workshops and scheduled at casters. The pure molten aluminum is blocked in the electrolytic cell until the casting workshop completes the processing before the pure molten aluminum is processed at casters. In other words, there are no buffers on different machines, which is a typical BFSP. On the other hand, when the orders of different models are arranged in the sequence, the electrolytic cell needs to be cleaned or the equipment needs to be replaced in the processing process, which produces the SDSTs. Additionally, the electrolysis and casting are processed in multiple factories. Therefore, this problem could be simplified as a SDST/DBFSP.

4. A speedup method for the insertion neighborhood

The inserting neighborhood transformation into permutation sequence is widely utilized in flow-shop scheduling problems (Shao et al., 2018). The speedup method consists of evaluating all permutations generated by the insertion of a single job in all possible positions. Therefore, the computational complexity of this process is very high. Fortunately, Taillard (1990) designed a speedup method for the PFSP to reduce the complexity of the insertion neighborhood. Shao et al.

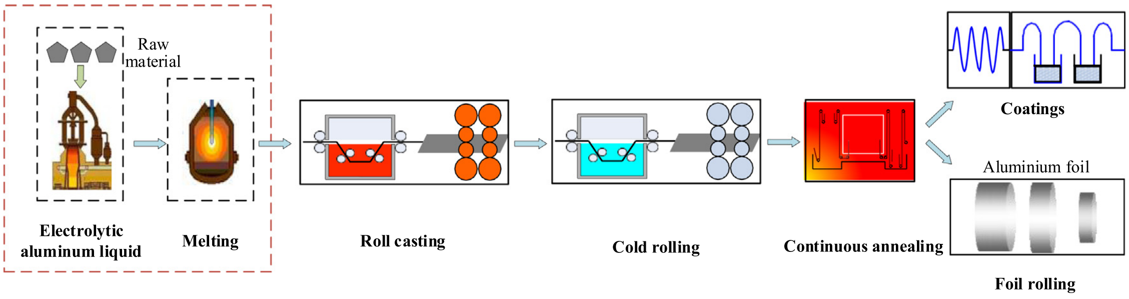


Fig. 1. The overall process of aluminum production.

(2018) extended this method to BFSP with SDST. In this section, a speedup method is proposed to reduce the computational complexity of the insertion neighborhood for $DF|blocking, s_{ijk}|C_{max}$.

Let $\prod = \{\pi_1, \pi_2, \dots, \pi_l, \dots, \pi_f\}$ presented a schedule of SDST/DBFSP, where $\pi_l = \{\pi_l(1), \pi_l(2), \dots, \pi_l(n_l)\}$ denotes the job sequence in factory l. n_l is the number of jobs in factory l. The objective of $DF|blocking, s_{ijk}|C_{max}$ is to find a permutation π_l^* in the $\prod$ such that $C_{max}(\pi_l^*) \le C_{max}(\pi_l)$. For the schedule $\prod = \{\pi_1, \pi_2, \dots, \pi_f\}$, the $D_{l,\pi_l(i),j}$ is calculated as follows.

$$D_{l,\pi_l(0),j} = 0, \forall l \in \{1,2,\dots,f\}, \forall j \in \{1,2,\dots,m\} \tag{14}$$

$$D_{l,\pi_l(i),0} = D_{l,\pi_l(i-1),1} + s_{\pi_l(i-1),\pi_l(i),1}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{1,2,\dots,n\} \tag{15}$$

$$D_{l,\pi_l(i),j} = \max\left\{D_{l,\pi_l(i-1),j+1} + s_{\pi_l(i-1),\pi_l(i),j+1}, D_{l,\pi_l(i),j-1} + p_{\pi_l(i),j}\right\}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{1,2,\dots,n\}, \forall j \in \{1,2,\dots,m-1\} \tag{16}$$

$$D_{l,\pi_l(i),m} = D_{l,\pi_l(i),m-1} + p_{\pi_l(i),m}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{1,2,\dots,n\} \tag{17}$$

$$C_{max}\left(\prod\right) = \max\left\{D_{l,\pi_l(n_l),m}\right\}, \forall l \in \{1,2,\dots,f\} \tag{18}$$

The $s_{\pi_l(0),\pi_l(1),k}$ in Eq. (16) presents the initial setup times of $\pi_l(i)$. Eq. (14) represents the departure time of the dummy job $\pi_l(0)$. Eq. (15) specifies the starting time of each job on the first machine in each factory. Eq. (16) determines the departure time of each job except the last machine. Eq. (17) computes the departure time of each job on the last machine in each factory. According to the above steps, the departure time of all jobs in each factory is calculated in turn. The makespan (C_{max}) is the maximum departure time of the last jobs in all factories by Eq. (18). The computational complexity of this recursion is $O(fmn)$.

The makespan of a permutation is calculated by traversing the permutation in the reverse order according to the reversibility of the flow-shop scheduling problem (Shao et al., 2019). Let $Q_{l,\pi_l(i),j}$ represents the reverse departure time of the ith job on machine j in the lth factory, which is calculated as follows:

$$Q_{l,\pi_l(n_l),m+1} = 0, \forall l \in \{1,2,\dots,f\} \tag{19}$$

$$Q_{l,\pi_l(n_l),j} = Q_{l,\pi_l(n_l),j+1} + p_{\pi_l(n_l),j}, \forall l \in \{1,2,\dots,f\}, \forall j \in \{m, m-1, \dots, 2\} \tag{20}$$

$$Q_{l,\pi_l(i),m+1} = Q_{l,\pi_l(i+1),m} + s_{\pi_l(i),\pi_l(i+1),m}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{n-1, n-2, \dots, 0\} \tag{21}$$

$$Q_{l,\pi_l(i),j} = \max\left\{Q_{l,\pi_l(i+1),j-1} + s_{\pi_l(i),\pi_l(i+1),j-1}, Q_{l,\pi_l(i),j+1} + p_{\pi_l(i),j}\right\}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{n-1, n-2, \dots, 0\}, \forall j \in \{m, m-1, \dots, 2\} \tag{22}$$

$$Q_{l,\pi_l(i),1} = Q_{l,\pi_l(i),2} + p_{\pi_l(i),1}, \forall l \in \{1,2,\dots,f\}, \forall i \in \{n, n-1, \dots, 0\} \tag{23}$$

$$C_{max}\left(\prod\right) = \max\left\{Q_{l,\pi_l(0),1}\right\}, \forall l \in \{1,2,\dots,f\} \tag{24}$$

The $p_{\pi_l(0),j}$ in Eq. (22) is 0. The last job processed in each factory is considered first, and then the previous job is calculated successively until all jobs in each factory are considered. Finally, the C_{max} is the maximum reverse departure time of the first jobs in all factories by Eq. (24).

Table 2
Processing times $p_{i,j}$ of jobs on machines M_1 and M_2.

	J_1	J_2	J_3	J_4	J_5
M_1	42	98	63	94	35
M_2	54	68	83	55	12

Table 3
Sequence-dependent setup times $s_{h,i,1}$ and $s_{h,i,2}$ of jobs on machines M_1 and M_2.

$s_{h,i,1}$	J_1	J_2	J_3	J_4	J_5	$s_{h,i,2}$	J_1	J_2	J_3	J_4	J_5
	63	30	17	97	62		14	27	65	8	94
J_1	0	26	63	76	91		0	52	39	40	52
J_2	96	0	42	55	95		93	0	89	39	38
J_3	19	52	0	18	96		12	84	0	88	40
J_4	8	97	6	0	88		6	24	82	0	86
J_5	3	46	21	55	0		89	12	40	8	0

There is an example to show how the decision variables reflect the solution by considering a SDST/DBFSP with five jobs ($n = 5$) and two factories ($f = 2$), each of which has two machines ($m = 2$). The Gantt chart of this example is shown in Fig. 2. In the F_1, after J_1 is processed on the first machine (M_1), the J_1 is blocked on the M_1 due to the SDST of the J_1 on the second machine (M_2) and no intermediate buffers between the machines M_1 and M_2. Similarly, J_3 is also blocked on the M_1 in F_2. As observed in Fig. 2, the SDST has an impact on the scheduling sequence and optimization objectives of DBFSP. The processing times and the SDSTs are given in Tables 2 and 3. The processing sequences of both factories are $\pi_1 = 4, 1, 5$ and $\pi_2 = \{2, 3\}$ via the CPLEX solver. The makespan of the whole system is 390.

For the first calculation method, according to Eqs. (14)–(17), the departure time of each job in each factory on each machine is calculated in the Appendix A.1.1. The corresponding makespan is also $C_{max}\left(\prod\right) = \max\left\{D_{1,\pi_1(3),2}, D_{2,\pi_2(2),2}\right\} = 390$. For the second calculation method, according to Eqs. (19)–(23), the reverse departure time of each job in each factory on each machine is calculated in the Appendix A.1.2. The corresponding makespan is also $C_{max}\left(\prod\right) = \max\left\{Q_{1,\pi_1(0),1}, Q_{2,\pi_2(0),1}\right\} = 390$. The time complexity of the two methods for an insertion operation is $O(fmn)$. Since each job is inserted into $(n-1+f)$ neighborhoods, the computational complexity of examining the entire insertion neighborhood is $O(fmn \times n(n-1+f)) \approx O(fmn^3)$. The computational cost of these methods becomes unacceptable with the increase of the problem scale.

The permutation $\pi_1 = \{\pi_1(1), \pi_1(2), \dots, \pi_1(n_1)\}$ in F_1 is divided into two partial sequences ($\pi_1^1 = \{\pi_1(1), \pi_1(2), \dots, \pi_1(q_1)\}$ and $\pi_1^2 = \{\pi_1(q_1+1), \pi_1(q_1+2), \dots, \pi_1(n_1)\}$) via the reversibility of flow-shop problems. The departure time $D_{1,\pi_1(i),j}$ of each job in π_1^1 is computed by utilizing Eqs. (14)–(17), where $i = 1, 2, \dots, q_1$ and $j = 1, 2, \dots, m$. Similarly, the reverse departure time $Q_{1,\pi_1(i),j}$ of each job in π_1^2 is

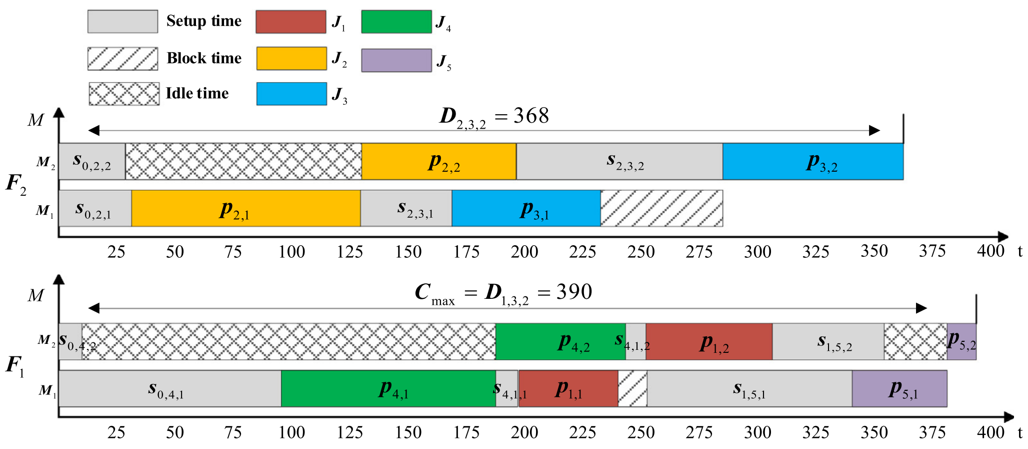


Fig. 2. Gantt chart of the SDST/DBFSP.

computed by utilizing Eqs. (19)–(23), where $i = q_1 + 1, q_1 + 2, \dots, n_1$ and $j = 1, 2, \dots, m$. The same method is applied to get the $D_{g,\pi_g(i),j}$ and $Q_{g,\pi_g(i),j}$ in each factory, where $g = 1, 2, \dots, f$. Finally, the makespan $C_{max}\left(\prod\right)$ is calculated as follows.

$$C_{max}\left(\prod\right) = \max_{g=1,2,\dots,f}\left\{\max_{j=1,2,\dots,m}\left(D_{g,\pi_g(q_g),j} + s_{\pi_g(q_g),\pi_g(q_g+1),2} + Q_{g,\pi_g(q_g+1),j}\right)\right\} \quad (25)$$

The job $\pi_l(p)$ is removed from π_l and reinserted into position u, where $p \in \{1, 2, \dots, n_l\}$ and $u \in \{1, 2, \dots, n_l\} \wedge u \notin \{p-1, p\}$. Hence, there are n_l permutations in total. These permutations are computed as follows:

Step 1. Calculate the departure time $D_{l,\pi_l(i),j}$ and the reverse departure time $Q_{l,\pi_l(i),j}$ of π_l, $\forall l \in \{1, 2, \dots, f\}, i = 1, 2, \dots, n_l, j = 1, 2, \dots, m$.

Step 2. Remove the job $\pi_l(p)$ to get a partial sequence $\pi_{l''} = \{\pi_{l''}(1), \pi_{l''}(2), \dots, \pi_{l''}(n_l - 1)\}$.

Step 3. Compute the departure time $D''_{l,\pi_{l''}(i),j}$, $i = 1, 2, \dots, n_l - 1, j = 1, 2, \dots, m$ as follows. If $i < p$, $D''_{l,\pi_{l''}(i),j} = D_{l,\pi_l(i),j}$; else calculate $D''_{l,\pi_{l''}(i),j}$ via Eqs. (13)–(17). Compute the reverse departure time $Q''_{l,\pi_{l''}(i),j}$, $i = n_l - 1, n_l - 2, \dots, 1, j = m, m-1, \dots, 1$ as follows. If $i > p$, $Q''_{l,\pi_{l''}(i),j} = Q_{l,\pi_l(i),j}$; else calculate $Q''_{l,\pi_{l''}(i),j}$ via Eqs. (19)–(23).

Step 4. For $u = 1, 2, \dots, n_l$ do the following steps:

Step 4.1 The job $\pi_l(p)$ is inserted into the uth position of $\pi_{l''}$ and generated a new permutation $\pi_{l'}$, $u \in \{1, 2, \dots, n_l\} \wedge u \notin \{p-1, p\}$. The departure time $D'_{l,\pi_{l'},(u),j}$ is computed via the obtaining $D''_{l,\pi_{l''}(u-1),j}$, $j = 1, 2, \dots, m$.

Step 4.2 The $C_{max}(\pi_l)$ is calculated as follows.

$$C_{max}(\pi_l) = \max_{g=1,2,\dots,f}\left\{\max_{j=1,2,\dots,m}\left(D'_{l,\pi_g(u),j} + s_{\pi_g(p),\pi_{g''}(u),2} + Q''_{g,\pi_{g''}(u),j}\right)\right\} \quad (26)$$

The computational complexity of Step 1 is $O(mn)$. Step 2 has $O(n)$ complexity. Step 3 has $O(2 \times mn)$ complexity. A loop of n steps is contained. The computational complexity of each step is $O(mn)$. Hence, the computational complexity of this speedup method is $O(fmn^2)$.

5. The discrete heuristic and three-stage meta-heuristic

In this section, a discrete heuristic and three-stage meta-heuristic (MBIST-TSM) is designed via incorporating the knowledge with the search process of the algorithm. Algorithm 1 describes the framework of the proposed MBIST-TSM algorithm and the following sections elaborate on its components. The framework of the MBIST-TSM algorithm is illustrated in Fig. 3. The proposed MBIST-TSM algorithm first generates an initial population containing NP solutions based on constructive heuristics. Then, the best solution for the population is chosen to guide the evolution of the population due to its improvement could be the finest comparing the other solutions. Then the following three-stage meta-heuristic is iteratively performed until a stopping criterion is satisfied. In each stage, a population updating procedure is utilized to determine which solutions will go to the next iteration.

```
Algorithm 1: The framework of the MBIST-TSM
Input: $NP, p_{i,j}, f$
Output: $\prod^{best}$
  Generate initial population $POP$ via Algorithm 2.
  Find the best solution $\prod^{best}$ in the population.
  while stopping criterion not satisfied do
      Conduct the first stage of the MBIST-TSM via Algorithm 3.
      Conduct the second stage of the MBIST-TSM via Algorithm 4.
      Conduct the third stage of the MBIST-TSM via Algorithm 5.
  end while
Return $\prod^{best}$
```

5.1. Constructive heuristic for SDST/DBFSP

The blocking times and idle times are caused by both setting times and processing times in SDST/DBFSP. The setting times are constantly changing for different scheduling sequences. Through a reasonable scheduling sequence, the blocking times and idle times caused by the setting times are minimized to reduce the maximum completion time of the jobs. In the proposed heuristic, a heuristic solution to the problem is generated by minimizing the blocking times and idle times created by SDSTs (MBIST) as much as possible. The proposed heuristic includes a new assignment rule and an insertion-based improvement procedure.

The MBIST has three procedures. The first step is to choose the job in the first position for each factory. Let $U = \{\pi(1), \pi(2), \dots, \pi(n)\}$ denotes the set of all unscheduled permutations. The calculation steps are as follows:

Step 1. We assume that when each job in U is the first job to be processed, it is arranged in descending order according to its setting time on the last machine.

Step 2. The first f jobs are successively assigned to the first position of factory 1 to f, i.e., $\psi = \{\pi_1(1), \pi_2(1), \dots, \pi_f(1)\}$. Let $U = U - \psi$.

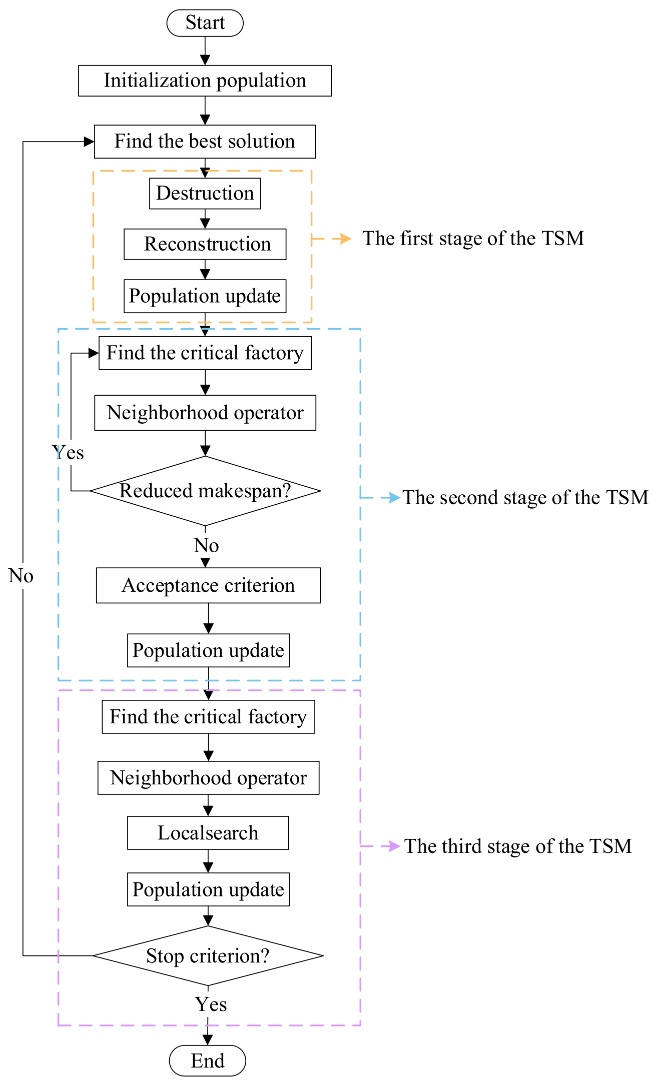


Fig. 3. The framework of the MBIST-TSM.

In the second step, the unscheduled jobs are arranged for each factory via the following steps:

Step 1. Sort the sum of processing times to each job by descending order in U.

Step 2. The first f jobs are assigned to the last positions of factory 1 to f. In this process, all factories are considered one by one. $\psi = \{\pi_1(n_1), \pi_2(n_2), \ldots, \pi_f(n_f)\}$. Let $U = U - \psi, \mu = 2, l = 1$.

Step 3. Calculate the measurement d_ρ of each job ρ in U via Eq. (27). Then, the job with the smallest value of d_ρ as $\pi_l(n_\mu)$. Let $U = U - \{\pi_l(n_\mu)\}$.

$$d_\rho = \sum_{j=2}^{m} \max\{0, (p_{\pi(\mu-1),j} - p_{\rho,j-1})\} \tag{27}$$

Step 4. Let $\mu = \mu + 1, l = l + 1$, where l is the index of factories. In this process, all jobs are considered one by one. Repeat this process until all jobs are assigned.

In the last step, the insertion procedure of NEH is performed in each factory. The NEH has carried out all jobs except for the first position and the last position in each factory. The example mentioned above is employed again to explain this part. In Table 3, the setting times of all jobs on the last machine are sorted by descending order, i.e., $U = \{\pi(5), \pi(3), \pi(2), \pi(1), \pi(4)\}$. Then, the jobs $\pi(5)$ and $\pi(3)$ are assigned to the first positions of factories. Next, the processing times of unscheduled jobs are sorted by descending order in Table 2, i.e., $U = \pi(2), \pi(4), \pi(1)$. The jobs $\pi(2)$ and $\pi(4)$ are assigned to the last positions of factories.$d_1 = \sum_{j=2}^{m} \max\left\{0, \left(p_{\pi(1),j} - p_{1,j-1}\right)\right\} = 12$. The implementation details are shown in the Appendix section.

Therefore, the job $\pi(1)$ is inserted in the second position in f_1. Finally, the sequences of both factories are $\pi_1 = 5, 1, 2$ and $\pi_2 = \{3, 4\}$. The Gantt chart is shown in Fig. 4. Although the maximum completion time is 390, the completion time in f_2 is reduced from 368 to 306. The idle times of the machine in f_2 are greatly reduced due to changing the scheduling sequence via the MBIST.

The procedure of MBIST is shown in Algorithm 2. The computational effort of MBIST is mainly spent on the assignment of jobs to the factories and the insertion procedure of NEH. In the first part, the time complexity of calculating the measurement is $O(mn)$. The speedup method presented in Section 4 is employed to evaluate the neighboring positions in the second part. Hence, the whole computational complexity of MBIST is $O(fmn^2)$.

5.2. *The proposed MBIST-TSM*

After the initialization process in Section 5.1, a three-stage meta-heuristic (TSM) is described in detail to solve the considering SDST/DBFSP. In the first stage of the TSM, an insertion-based neighborhood operator of different factories is developed to explore promising regions in the decision space. In the second stage of the TSM, a local search operator is embedded to enhance the exploitation ability. Meanwhile, a simulated annealing-like acceptance criterion is employed to maintain the diversity of the population. Finally, an insertion operation for critical factories (the factories with the largest completion times) is introduced to further improve the accuracy of the solutions. Each phase of the TSM is illustrated in the following sub-sections.

5.2.1. *Representation and initialization of solutions*

A solution in the TSM is corresponding to a location in the search space. For the SDST/DBFSP, a set of n jobs have to assign to f factories, meanwhile, the jobs in each factory must be processed on m machines in certain sequences. The f lists ($\prod = \{\pi_1; \pi_2; \ldots; \pi_f\}$) are employed to represent the sequence in each factory directly (Zhao et al., 2020b). The example in Fig. 2 is encoded as follows. Jobs 5, 1, and 4 are processed in the first factory with sequences $5 \to 1 \to 4$.

$$\Pi = \begin{pmatrix} 5 & 1 & 4 \\ 2 & 3 & \end{pmatrix}$$

The MBIST mentioned in Section 5.1 is employed to generate NP initial solutions. It is noticed that the NP initial solutions have the same position. Nevertheless, each solution is searched independently to ensure the diversity of understanding in the following three search methods. On the other hand, the initial solutions generated by the MBIST reduced the cost in the later search process. The initial population randomly generated without any problem characteristics guidance contains many unpromising solutions. At the same time, these solutions waste a lot of resources in the later search process, which affects the performance of the algorithm. As a result, the ability of the algorithm to explore other more promising regions would diminish. Therefore, the following three unique evolution methods for the TSM are designed to ensure the search diversity of the algorithm.

5.2.2. *The first stage of the TSM*

In SDST/DBFSP, the closest to the optimal solution is often the job sequence of the critical factory. Because the completion time of the critical factory determines the C_{max} of the whole scheduling sequence. Therefore, in the first stage of the TSM, a neighborhood insertion operation for the critical factory (f^c) is designed to improve the quality of the initial solutions.

```
Algorithm 2: MBIST
Input: $p_{i,j}, s_{h,i,j}, f$
Output: $\prod_s$
    Let $U = \{\pi(1), \pi(2), \dots, \pi(n)\}$, sort the jobs according to the setting time $s_{0,i,1}$, $i = 1,2,\dots,n$.
    The first $f$ jobs in the sequence are assigned to the first position of each factory in turn.
    $\psi = \{\pi_1(1), \pi_2(1),\dots, \pi_f(1)\}$. Let $U = U - \psi$.
    Sort the processing time of all jobs by descending order in $U$.
    The first $f$ jobs are assigned to the last positions of factory 1 to $f$.
    $\psi = \{\pi_1(n_1), \pi_2(n_1),\dots, \pi_f(n_f)\}$. Let $U = U - \psi$.
    $\mu \leftarrow 1; l \leftarrow 1$.
    while $U \neq \emptyset$ do
        Calculate the measurement $d_\rho$ of each job $\rho$ in $U$ via Eq. (27).
        The job with the smallest value of $d_\rho$ as $\pi_l(n_\mu)$. Let $U = U - \{\pi_l(n_\mu)\}$.
        $\mu = \mu + 1, l = l + 1$.
        if $l = f$ then $l = 1$
        end if
    end while
    for $l = 1$ to $f$ do
        for $n = 2$ to $n_l - 1$ do
            Insert the job $\pi_l(n)$ in factory $l$ resulting in the lowest completion time.
        end for
    end for
Return $\prod_s$
```

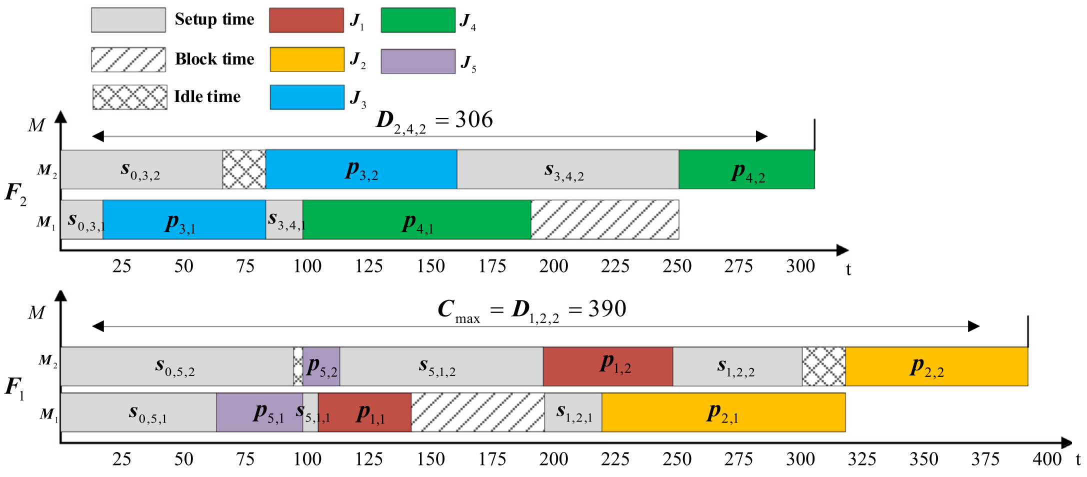


Fig. 4. Gantt chart of the example after MBIST.

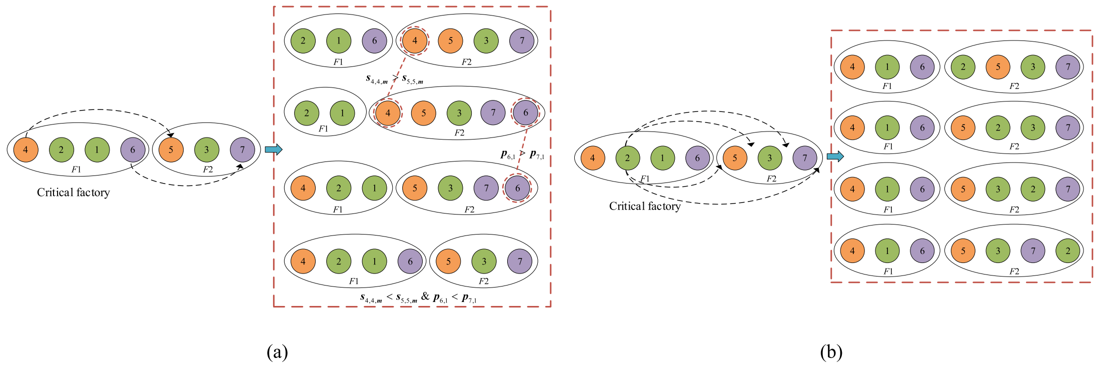


Fig. 5. The illustration of the first stage of the TSM. (a) The operations about the first and last jobs in f^c. (b) The operations of the other jobs in f^c.

The proposed insertion-based neighborhood operator consists of two stages. The main idea of this operator inspires by the destruction–reconstruction of IG (Wang et al., 2017). In the destruction phase, all the jobs in f^c are copied to form an external sequence set τ^c. Then, each job in τ^c is taken out from the f^c in order for the second stage of reconstruction to break the original permutation of f^c. In the reconstruction phase, the first job in f^c is tried to be inserted in other factories. It is worth noting that the first job can only be inserted into the first position of each factory to retain the problem characteristics extracted in the MBIST because the setting time of this job is the longest. The second step is to consider the last job of the f^c. Similarly, the job can only be inserted into the last position of the factory because it has the longest processing time on the last machine. The illustration of the operations of the first and last jobs in f^c are shown in Fig. 5(a). Finally, other jobs in the τ^c are taken out in turn and tried to insert them into all positions of all factories. This process is repeated until all

```
Algorithm 3: The first stage of the TSM
Input: $p_{i,j}$, $f$, $s_{h,i,j}$, $POP$
Output: $POP$
for $s = 1$ to $NP$ do
    Find the critical factory $f^c$ in $\prod_s$, set the number of jobs in $f^c$ as $n_c$, copy the jobs in $f^c$ to an external sequence set $\tau^c$.
    Calculate the $C_{max}$.
    for $k = 1$ to $f$ do
        if $s_{\pi_{f^c}(1),\pi_{f^c}(1),m} > s_{\pi_k(1),\pi_k(1),m}$ then
            Insert $\tau^c(1)$ job in the first position of the $f_k$, calculate the $C_{max}^{temp}$.
            if $C_{max}^{temp} < C_{max}$ then
                Update the $\prod_s$.
                $C_{max} = C_{max}^{temp}$, $n_c = n_c - 1$, $\tau^c = \tau^c - \tau^c(1)$.
            end if
        end if
        if $p_{\pi_{f^c}(n_c),1} > p_{\pi_k(n_k),1}$ then
            Insert $\tau^c(n_c)$ job in the last position of the $f_k$, calculate the $C_{max}^{temp}$.
            if $C_{max}^{temp} < C_{max}$ then
                Update the $\prod_s$.
                $C_{max} = C_{max}^{temp}$, $n_c = n_c - 1$, $\tau^c = \tau^c - \tau^c(n_c)$.
            end if
        end if
        for $\alpha = 1$ to $n_c$
            Test the $\alpha$-th job from $\tau^c$ in all possible positions of the $f_k$ according to the $\pi_k$.
            Insert $\tau^c(\alpha)$ job in the position with the lowest $C_{max}$.
            Update the $\pi_k$.
        end for
    end for
    Update the $\prod_s$.
    Conduct the Population_Update via Algorithm 6.
end for
Return $POP$
```

jobs in τ^c are considered. The illustration of the operations about the other jobs in f^c are shown in Fig. 5(b). The first stage of the TSM is shown in Algorithm 3.

As seen in Algorithm 3, both the first job and the last job in τ^c are tested for $(f-1)$ insertion positions, so the computational complexity of these two jobs is $O(2 \times (f-1) \times m)$. Each remaining job in τ^c is inserted into all positions of all factories, so each job is inserted into $(n+f-1)$ positions. Therefore, the computational complexity of one full insertion for a single job is $O(m \times (n+f-1))$. The computational complexity of other jobs is $O(n_c \times m \times (n+f-1))$. Whereas the number of jobs in the critical factory is not accurately calculated. Under extreme conditions, when the number of jobs in the critical factory is n, the maximum computational complexity is $O(n \times m \times (n+f-1))$. The speedup method presented in Section 4 is employed to evaluate the neighboring positions. Thus, the computational complexity of the first stage of the TSM is $O(NP \times mn^2)$.

5.2.3. The second stage of the TSM

In the second stage of the TSM, a neighborhood operator is designed to continuously update critical factories. Meanwhile, an acceptance criterion similar to simulated annealing is employed to prevent falling into local optima. The illustration of the second stage of the TSM is shown in Fig. 6. As seen in Algorithm 4, the first job in τ^f from the sequence is removed. Afterward, the job is inserted into all possible positions in all factories. If the C'_{max} found via the reinsertion makes the makespan improved, the job is inserted into this position and the search process is restarted. Otherwise, a simulated annealing-like acceptance criterion of IG of Ruiz and Stützle (2007) is employed to keep the diversity of the population. To be specific, if λ is less than the accepted probability η, the new scheduling solution created by the current candidate solution is accepted, where λ is a random number in the interval [0,1]. The probability η is defined as follows.

$$\eta = e^{-\frac{C'_{max}-C_{max}}{Temp}} \tag{28}$$

$$Temp = T \times \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{p_{i,j}}{10 \times m \times n} \tag{29}$$

where C'_{max} represents the makespan of the scheduling solution created by the current candidate solution, C_{max} is the makespan of the best solution currently found. $Temp$ is a constant temperature, which is calculated by (29) following the recommendation of Osman and Potts (1989). T is a control parameter. If C'_{max} is close to C_{max}, then the value of η is close to 1. Therefore, the corresponding solution of C'_{max} is accepted with a large probability. By contrast, if C'_{max} is largely worse than C_{max}, the probability of being accepted is considerably small. Afterward, the second job of τ^f is reinserted in the same way until all the jobs in τ^f are considered.

Each candidate solution generates $f \times (n+f-1)$ neighboring solutions. Therefore, a total of $NP \times (n+f-1)$ neighboring solutions is generated via the searching for prey, where NP is the population size. The creation of a new neighboring solution at least requires $O(m)$. Thus, the computational complexity of the second stage of the TSM is $O(NP \times f \times (n+f-1) \times m)$.

5.2.4. The third stage of the TSM

In many meta-heuristics, the individuals uninterruptedly approach the presumptive global best solution in the evolution. However, in certain worse cases, it will lead the population to evolve toward the local optimal because the selected leader is not the global optimal but the local optimal. As a result, many offspring will be crowded around the bad leader. At the same time, a lot of computational costs are also spent in a position that should not be searched deeply. This is one of the reasons that the kind of meta-heuristic easily converges prematurely (Zhang et al., 2021). Hence, there is an urgent need for an effective way to utilize the information of leaders with different characteristics to construct new solutions. The neighborhood structure can greatly affect the performance of the local search procedure (Deng et al., 2020). Therefore, a local search method that utilizes different neighborhood structure is presented in the third stage of the TSM.

Any strategy that does not improve critical factories cannot effectively reduce C_{max} (Shao et al., 2020b; Zhao et al., 2022a,b). Specifically, δ jobs are randomly selected from the critical factory to construct an external sequence set τ^δ, where δ is a control parameter. The

Algorithm 4: The second stage of the TSM

```
Input: $p_{i,j}$, $f$, $s_{h,i,j}$, $POP$
Output: $POP$
  for $s = 1$ to $NP$ do
    $\tau^f$ ←Randomly select a job from each factory.
    $\Pi_{s'}$ ← Remove all jobs in $\tau^f$ from $\prod_s$.
    for $j = 1$ to $f$
      Test the $j$th job from $\tau^f$ in all possible positions of all factories according to the $\Pi_{s'}$.
      Insert $\tau^f(j)$ job in the position with the lowest $C'_{max}$.
      Update the $\Pi_{s'}$.
      if $C'_{max} < C_{max}$ then
        Insert the job $\tau^f(j)$ at position $p_{min}$, update the $\prod_s$.
        $C_{max} = C'_{max}$;
      else
        if $\lambda < \eta$ then
          Insert the job $\tau^f(j)$ at position $p$ with $C'_{max}$, update the $\prod_s$.
          $C_{max} = C'_{max}$.
        end if
        $j = j + 1$.
      end if
    end for
    Conduct the Population_Update via Algorithm 6.
  end for
Return $POP$
```

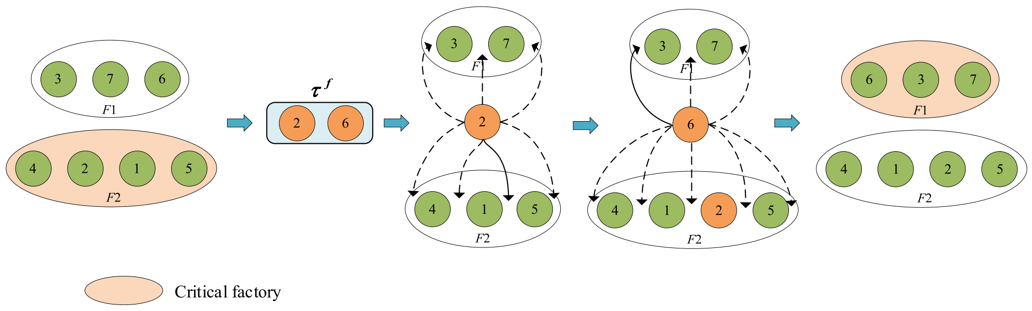


Fig. 6. The illustration of the second stage of the TSM.

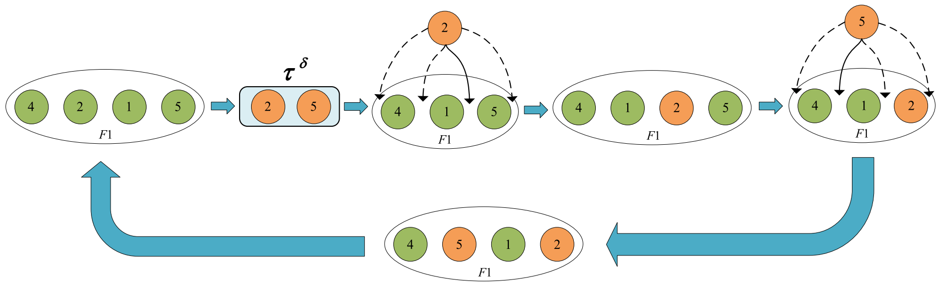


Fig. 7. The illustration of the third stage of the TSM.

value of δ will be investigated in Section 6.1. Afterward, the jobs in τ^δ are taken out from f^c in turn and reinserted into all possible positions of the critical factory. If the makespan of the current factory is improved, the critical factory is reselected and the above operations are performed. Otherwise, the operation is terminated. The illustration of the phase of the third stage of the TSM is shown in Fig. 7.

As seen in Algorithm 5, the first job in τ^δ is inserted into $(n_c - \delta + 1)$ positions. All jobs are inserted into $\left(n_c - \delta\right) \times \delta + \frac{\delta}{2} \times (\delta + 1)$ positions. Therefore, the computational complexity of one full insertion for τ^e is $O\left(NP \times m \times \left(\left(n_c - \delta\right) \times \delta + \frac{\delta}{2} \times (\delta + 1)\right)\right)$. Whereas n_c is not be accurately calculated, and the critical factory is constantly updated in the iterative process. Only in extreme circumstances, the number of jobs in the critical factory is n, and the size of τ^δ is n, the maximum computational complexity is $O(NP \times m \times \frac{n\times(n+1)}{2})$.

5.2.5. *Population update procedure*

In this phase, the current population straightway goes into the next iteration if the offspring solution is already in the population; otherwise, it will be updated by the following method. If the offspring solution is better than the worst solution found so far, the worst solution in the current population will be replaced with the offspring solution. Otherwise, the current population straightway goes into the next iteration. This strategy can speed up the convergence rate of the algorithm without deteriorating the utility of the population mechanism, due to that the population will preferentially turn to the best so far solution and then be filled with high-quality solutions over several iterations. The population update procedure is shown in Algorithm 6.

```
Algorithm 5: The third stage of the TSM
Input: $NP, p_{i,j}, s_{h,i,j}, f, POP$
Output: $POP$
  for $s = 1$ to $NP$ do
    $flag \leftarrow true.$
    while $flag$ then
      $flag \leftarrow false.$
      Find the critical factory $f^c$ in $\prod_s$.
      $\tau^\delta \leftarrow \delta$ jobs are randomly selected from $\pi_c$ in $f^c$.
      $\pi_c' \leftarrow$ Remove all jobs in $\tau^\delta$ from $\pi_c$.
      for $k = 1$ to $\delta$
        Test the job $\tau^\delta(k)$ in all possible positions of $\pi_c'$.
        Find the position $p_{min}$ that results in the lowest makespan.
        Insert the job $\tau^\delta(k)$ at position $p_{min}$, update the $\pi_c'$ and makespan $C$ of the $\pi_c'$.
      end for
      if $C < C_{max}$ then
        $\pi_c \leftarrow \pi_c'$.
        for $i = 1$ to $f$
          if $C < C_i$ then
            $flag \leftarrow true.$
          end if
        end for
      end if
    end while
    Update the $\prod_s$.
    Conduct the Population_Update via Algorithm 6.
  end for
Return $POP$
```

```
Algorithm 6: Population_Update
Input: $\prod_s$
Output: $POP$ and $\prod^{best}$
  if $\prod_s$ is not in $POP$ then
    Find the worst solution $\prod^{worst}$ in $POP$.
    if $\prod_s$ is better than $\prod^{worst}$ then
      Remove the $\prod^{worst}$ in $POP$ and $POP = \{POP; \prod_s\}$.
    end if
  end if
Return $POP$ and $\prod^{best}$.
```

6. Experiment and analysis

There are two types of data that are tested to evaluate the performance of the MBIST-TSM. One is randomly generated according to the scope of the practical production data in a Western Chinese aluminum fabrication company. The MBIST-TSM is compared with CPLEX owing to the small production scale of the actual plant in the company. Additionally, CPLEX provides effective lower bounds as the criterion for comparison, therefore, the MBIST-TSM is not necessary to compare with other approaches on small-scale issues. Another data is the well-known benchmark set proposed by Ruiz et al. (2005) for SDST/PFSP. The benchmark set is divided into four subsets, which are referred to as SSD10, SSD50, SSD100, and SSD125. The SSD10 represents that the setup times correspond to 10% of the average processing times and so on. Each set contains certain instances with 12 different combinations for the number of jobs (n) and the number of machines (m). The $n \times m$ combinations are: {20, 50, 100} × {5, 10, 20}, {200} × {10, 20}, and {500} × {20}. Each combination contains 10 instances. The number of factories is set to $f = \{2, 3, 4, 5, 6, 7\}$. Hence, a total of $4 \times 12 \times 10 \times 6 = 2880$ instances are tested. The processing times of each job are uniformly generated and distributed in the range [1, 99]. Whereas, the benchmark set is incomplete since the initial setup times of each job are not considered. The initial setup times are generated via the method of Ruiz et al. (2005). In Section 6.1, a new benchmark set is generated by the same method for calibration, since the over-fitting results via the same benchmark set (Ruiz et al., 2005). The termination criterion of algorithms to a maximum elapsed CPU time ($c \times n \times m \times f$) milliseconds (Naderi and Ruiz, 2014), where the c is set to 5, 20, and 60. The average relative percentage deviation (ARPD) index is utilized

Table 4
Results of ANOVA for parameters calibration of the MBIST-TSM.

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	p-value
NP	23.122	2	11.562	690.34	**0.0000**
δ	2.047	2	1.023	46.89	**0.0000**
T	1.960	3	0.563	26.72	**0.0000**
$NP * \delta$	0.282	4	0.071	0.73	**0.4908**
$NP * T$	0.330	6	0.056	0.60	**0.0007**
$\delta * T$	0.406	6	0.068	0.74	**0.0428**
Error	2413.8	34 520	0.091		
Total	2458.4	34 560			

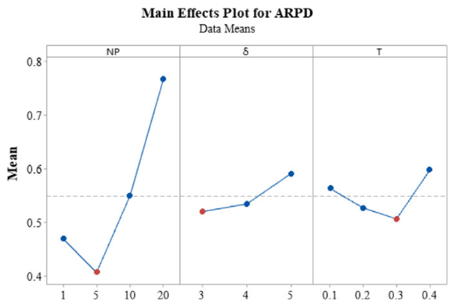


Fig. 8. Main effects plot of all parameters.

to measure the results. The ARPD is calculated by (30), where R is the number of runs, C_i is the solution generated by a specific algorithm in the ith experiment for a given instance. C_{opt} is the minimum makespan found by all algorithms. Obviously, the algorithm with the smallest ARPD has the best performance.

$$ARPD = \frac{1}{R} \cdot \sum_{i=1}^{R} \frac{C_i - C_{opt}}{C_{opt}} \cdot 100\% \tag{30}$$

The experiments are run in the Windows Server 2016 Datacenter under the hardware environment of Intel (R) Xeon (R) Gold 5115 CPU @ 2.40 GHz processor and 16.0 GB of RAM. All implemented algorithms are coded by Java. The version of the Java development kit (JDK) is jdk1.8.0_121. For communicating with other interested

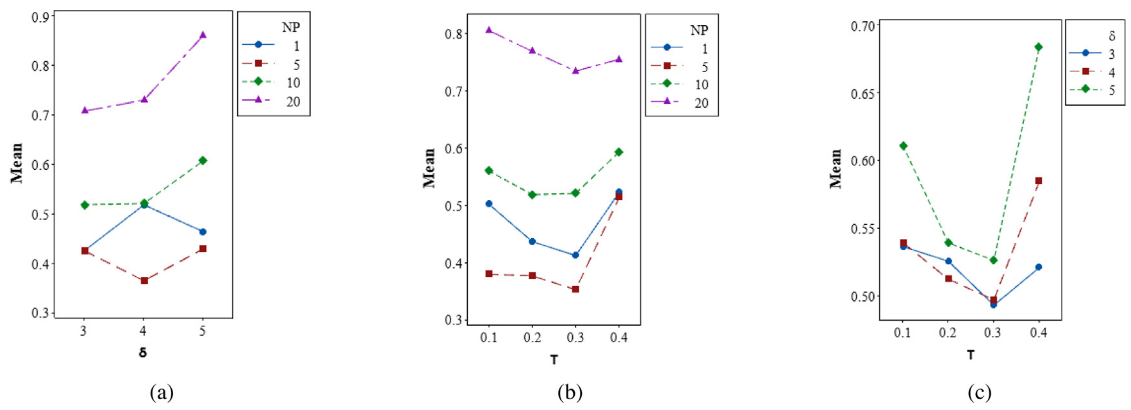


Fig. 9. Interaction plots of three parameters. (a) $NP * \delta$. (b) $NP * T$. (c) $\delta * T$.

Table 5
The computational results of different instances.

Instance			CPLEX				MBIST-TSM	
nos	Number of jobs	LB	C_{max}	CPU time (s)	C_{max}	CPU time (day)	C_{max}	CPU time (s)
1	12	30	**30**	3600	**30**	1	32	24.68
2	18	43	**43**	3600	**43**	1	47	39.08
3	24	60	62	3600	**60**	1	**60**	48.22
4	27	69	73	3600	70	1	70	60.90
5	36	92	95	3600	95	1	95	89.27
6	45	114	120	3600	120	1	**114**	129.17
7	48	120	128	3600	128	1	**120**	153.61
8	54	131	140	3600	140	1	**131**	178.74
9	60	150	159	3600	159	1	**150**	214.28
10	72	182	200	3600	196	1	**182**	297.60

Table 6
ARPDs and average CPU time for all heuristics (SSD10 and SSD50).

f	SSD10					SSD50				
	DSPT	DLPT	DLS	DNEH	MBIST	DSPT	DLPT	DLS	DNEH	MBIST
ARPD										
2	12.435	11.653	15.105	16.147	**0.003**	9.104	8.112	11.009	11.874	**0.023**
3	11.880	10.601	13.400	15.832	**0.082**	8.490	7.489	9.582	11.820	**0.088**
4	11.151	9.172	11.871	15.221	**0.158**	8.560	6.856	8.765	11.810	**0.141**
5	9.777	7.327	9.588	13.837	**0.534**	7.718	5.581	7.700	11.018	**0.743**
6	10.528	7.434	9.318	13.961	**0.838**	8.409	5.373	7.463	10.584	**0.843**
7	9.930	6.367	8.780	13.356	**0.864**	8.179	4.952	6.898	10.682	**0.826**
Average	10.950	8.759	11.344	14.726	**0.413**	8.410	6.394	8.569	11.298	**0.444**
Average CPU time										
2	6.889	6.650	**6.095**	8.574	8.122	15.703	13.282	**13.056**	16.999	15.963
3	**5.717**	6.659	5.828	7.039	7.962	**11.490**	12.640	11.703	14.618	14.844
4	5.403	5.502	**5.027**	7.707	6.771	11.111	10.736	**10.346**	14.924	13.131
5	5.277	**5.095**	5.247	6.680	6.011	10.444	**10.070**	10.432	13.693	11.892
6	5.129	5.403	**5.071**	6.588	6.407	10.268	10.723	**10.247**	13.321	12.267
7	5.271	5.156	**5.026**	6.751	5.657	10.754	10.048	**9.938**	13.398	11.612
Average	5.614	5.744	**5.382**	7.223	6.821	11.628	11.250	**10.954**	14.492	13.285

researchers, the code of all algorithms are published on GitHub (https://github.com/banian2314/MBIST-TSM).

6.1. Calibration of the parameters

The proposed MBIST-TSM contains three critical parameters: the population size NP, the size of the external sequence set δ, and the temperature T. The design of experiments (DOE) approach (Montgomery, 2009) is employed to calibrate these parameters. The potential values (levels) of each parameter (factor) are determined according to certain preliminary experiments and the relevant literature (Huang et al., 2020; Shao et al., 2018). The final levels chosen for the factors are: $NP = \{1, 5, 10, 20\}$, $\delta = \{3, 4, 5\}$, and $T = \{0.1, 0.2, 0.3, 0.4\}$. There are $4 \times 3 \times 4 = 48$ different parameter combinations in total. A full factorial design is considered to calibrate parameters and analyze the influence of different parameters on the performance of the MBIST-TSM. All of the 48 combinations above have the same termination criterion with the maximum elapsed CPU time ($t = 10 \times n \times m \times f$) milliseconds. The new benchmark instances with $n \times m = \{40, 80, 120, 200\} \times \{5, 10, 20\}$ are employed as the test bed. Each combination contains one instances and four setting time levels. The number of factories is set to $f = \{3, 5, 7\}$. Hence, a total of $48 \times 12 \times 1 \times 4 \times 3 = 6912$ instances are tested to calibrate the parameters. Each instance is independently tested 5 times. To sum up, there are $6912 \times 5 = 34\,560$ results in this section.

The experimental results are investigated by utilizing the analysis of variance (ANOVA) technique to demonstrate interactions between the parameters. The technique has been frequently employed in the scheduling literature to calibrate methods (Ribas et al., 2019; Zhao et al., 2021). The three main hypotheses (i.e., independence, normality, and heteroscedasticity) of ANOVA are checked beforehand and can be

Table 7
ARPDs and average CPU time for all heuristics (SSD100 and SSD125).

f	SSD100					SSD125				
	DSPT	DLPT	DLS	DNEH	MBIST	DSPT	DLPT	DLS	DNEH	MBIST
ARPD										
2	5.849	5.467	7.381	7.914	**0.093**	5.306	4.856	6.551	6.989	**0.173**
3	5.729	4.651	6.158	7.910	**0.307**	5.056	4.104	5.077	7.144	**0.372**
4	5.633	4.430	5.747	8.139	**0.363**	4.944	3.814	5.116	6.819	**0.477**
5	5.566	3.852	5.558	7.863	**1.050**	4.884	3.309	4.609	7.192	**1.139**
6	5.826	3.521	4.874	7.834	**1.300**	5.389	2.955	4.053	7.307	**1.568**
7	5.536	3.111	4.710	7.640	**1.109**	5.325	2.698	4.589	7.486	**1.317**
Average	5.690	4.172	5.738	7.883	**0.704**	5.151	3.623	4.999	7.156	**0.841**
Average CPU time										
2	22.116	**19.310**	19.887	24.497	22.632	29.414	27.428	**26.273**	33.113	29.657
3	**17.419**	18.751	17.502	22.052	21.501	**23.636**	25.834	24.682	30.061	28.534
4	16.350	15.848	**15.527**	21.931	19.336	21.869	21.540	**21.088**	29.163	26.102
5	15.618	**15.243**	15.737	20.536	18.220	21.261	**20.868**	21.519	27.798	25.024
6	15.456	15.888	**15.317**	20.020	18.190	**20.940**	21.534	20.962	27.091	24.132
7	16.233	15.165	**14.846**	20.200	17.124	22.008	20.832	**20.648**	27.366	23.121
Average	17.199	16.701	**16.469**	21.539	19.500	23.188	23.006	**22.529**	29.098	26.095

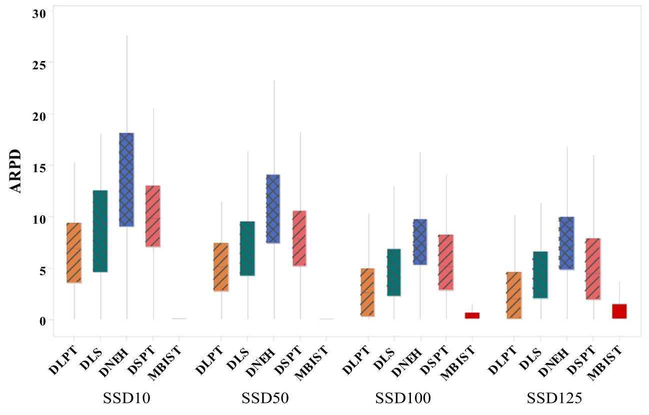


Fig. 10. The box plot of heuristics.

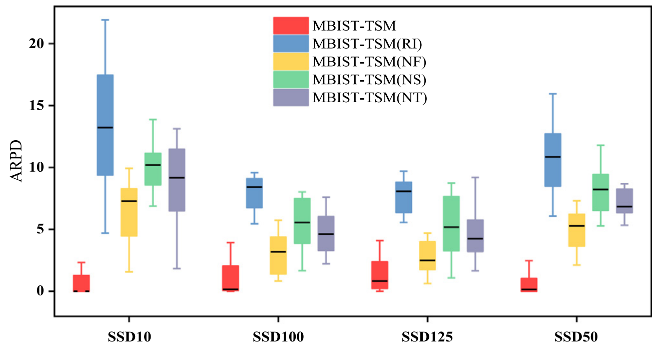


Fig. 12. The box plots of all variants of the MBIST-TSM.

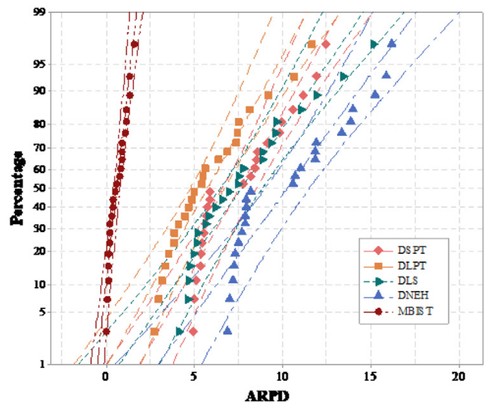


Fig. 11. The probability plot of heuristics.

satisfied. The results of ANOVA are shown in Table 4. In ANOVA, the F-ratio is an indicator to reflect the importance of each parameter when p-value is close to zero. A large F-ratio indicates that the analyzed factor has a considerable effect on the response variable.

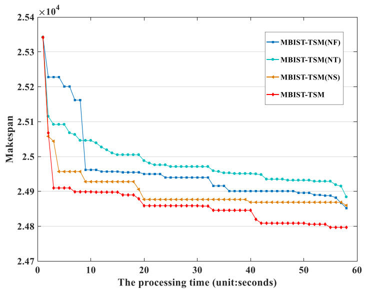
As observed in Table 4, all of the parameters are significantly affected the performance of the proposed algorithm owing to the three p-values being smaller than 0.05. Among these three parameters, the parameter NP yields the greatest F-ratio, which indicates that it has an important effect on the response variable. The main effects plots of all parameters are shown in Fig. 8. The performance of the MBIST-TSM is decreased with increasing of NP. It is precise because that a large value of NP leads to a great similarity between individuals. Therefore, a lot of computing time is wasted exploring similar individual neighborhoods, which makes the algorithm easily trap into the local optimum and decreases its ability of exploration. However, a small value of NP leads to the extremely limited number of neighborhood solutions exploited by the MBIST-TSM, which will also make the algorithm fall into the local optimal value. In particular, this phenomenon is most prominent when the population size is 1. As also seen in Table 4, the parameter δ ranks second place. δ is the disturbance intensity in the stage of the second stage of the TSM. If δ is too small, the search space should be limited, which causes the new solutions to be very close to the parent individuals. On the contrary, a large value of δ causes the process to take most of the time of the entire algorithm evolution and does a lot of the same search, which can lead to low efficiency of the algorithm. As observed in Table 4, the parameter T ranks third place. A small value of T causes that the value of η to be small according to Eq. (28). Thereby, the diversity of the population will decrease accordingly in Algorithm 3. Conversely, the value of T is too large, which increases the number of inferior solutions in the population and restricts the exploration ability of the algorithm. The best performance of the proposed algorithm is achieved when T is set to 0.3 according to Fig. 8.

Additionally, the $NP{*}\delta$, $\delta{*}T$, and $NP{*}T$ interactions are significant since their p-values are less than 0.05 in Table 4. If there are significant interactions between the parameters, the main effects plot is not meaningful (Tasgetiren et al., 2017). Therefore, the 2-level interactions between the factors are investigated to examine the interaction plot and whether it is satisfied with the judgments in the main effects plot. The interaction plots for these interactions are depicted in Fig. 9. The interactions of these parameters are weak and do not deviate from the

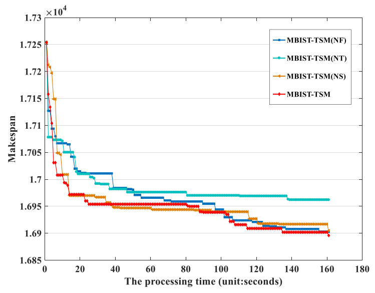
Table 8
Computational results of all variants of the MBIST-TSM.

Instance	MBIST-TSM				MBIST-TSM(RI)				MBIST-TSM(NF)				MBIST-TSM(NS)				MBIST-TSM(NT)			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
20 × 5	5.498	**4.616**	**3.672**	**2.685**	12.729	10.579	8.455	9.704	**4.546**	5.347	5.388	4.392	9.449	9.544	7.015	8.181	6.038	6.391	7.59	9.196
20 × 10	2.328	2.476	2.600	2.815	4.699	6.083	7.050	7.919	**2.159**	**2.348**	**1.628**	**2.084**	7.856	9.314	8.032	7.937	4.924	5.345	6.542	6.136
20 × 20	2.327	**1.535**	**1.023**	**0.915**	5.294	6.539	6.920	8.298	**1.581**	2.128	2.507	2.065	6.887	7.394	8.003	8.749	1.843	3.131	4.37	5.451
50 × 5	**0.000**	**0.412**	**1.491**	**2.072**	20.958	15.964	9.559	9.325	7.817	5.214	2.261	3.245	13.870	11.794	5.207	4.972	10.589	8.641	2.382	2.882
50 × 10	**0.000**	**0.307**	**0.000**	**0.426**	14.539	13.425	9.580	8.101	6.772	7.127	4.145	4.124	10.830	10.469	5.893	6.388	7.769	6.939	3.557	4.383
50 × 20	**0.209**	**0.000**	**0.066**	**0.009**	8.320	9.625	9.422	9.689	4.411	5.425	5.742	4.699	7.021	7.093	7.958	7.363	6.983	7.878	6.942	6.046
100 × 5	**0.000**	**0.562**	3.943	4.106	21.912	12.000	8.573	8.201	7.787	2.772	**0.835**	**0.804**	11.217	5.279	3.316	2.775	13.129	6.391	3.98	3.58
100 × 10	**0.000**	**0.000**	**0.132**	**0.695**	16.906	13.862	8.751	8.065	8.042	6.661	3.910	2.068	10.008	8.164	5.146	4.692	11.426	8.643	5.488	5.106
100 × 20	**0.004**	**0.000**	**0.082**	**0.056**	10.471	10.844	8.418	7.117	5.537	5.776	4.618	3.897	9.355	9.165	6.987	5.423	7.745	7.629	5.509	4.139
200 × 10	**0.000**	**0.000**	**0.119**	**1.274**	17.983	10.995	5.457	5.556	8.544	4.559	1.186	1.475	11.070	5.999	2.729	2.547	11.53	6.755	3.065	2.661
200 × 20	**0.000**	**0.000**	**0.000**	**0.000**	13.009	10.887	6.610	5.640	9.277	7.306	3.971	2.895	10.363	8.288	4.475	3.788	10.775	8.698	4.87	3.825
500 × 20	**0.000**	**0.000**	**0.179**	0.756	13.458	7.378	2.885	2.213	9.930	4.761	1.149	**0.625**	11.236	5.642	1.671	1.085	12.608	6.337	2.221	1.661
Average	**0.864**	**0.826**	**1.109**	**1.317**	13.356	10.682	7.640	7.486	6.367	4.952	3.111	2.698	9.930	8.179	5.536	5.325	8.78	6.898	4.71	4.589

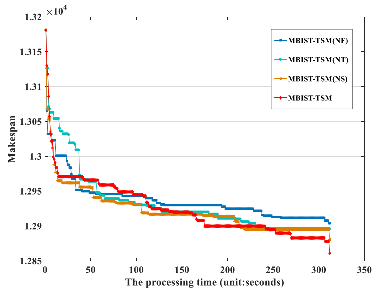
Note: 1: SSD10, 2: SSD50, 3: SSD100, 4: SSD125.



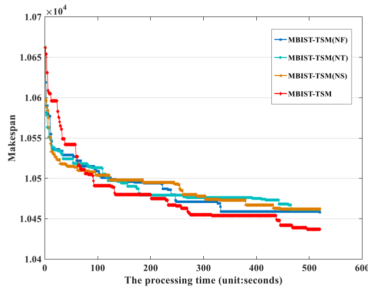
(a) $f = 2$



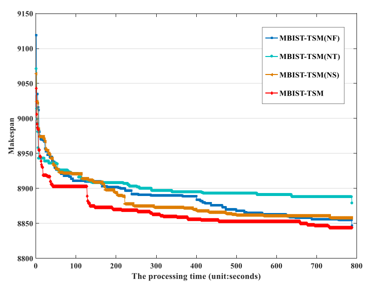
(b) $f = 3$



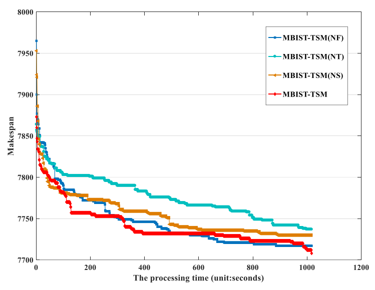
(c) $f = 4$



(d) $f = 5$



(e) $f = 6$



(f) $f = 7$

Fig. 13. The convergence curves of the strategies at different factories.

above rules. Hence, according to the experiment and analyses above, the suggested parameters are summarized as follows: $NP = 5$, $\delta = 4$, and $T = 0.3$.

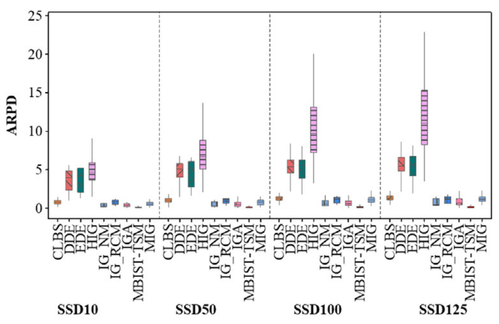
6.2. Experimental comparison of actual plant data

In an aluminum plant in Western China, the production duration is 24 h. The processing times and setting times are generated as integers in the experiments for ease of calculation. The processing times of jobs are between 3 h to 8 h. The setting times are between 1 h to 4 h. There are three parameters of the instances: the number of factories (f), the number of machines (m), and the number of jobs (n), where $f = \{3\}, m = \{4\}, n = \{12, 18, 24, 27, 36, 45, 48, 54, 60, 70\}$. The combinations of these parameters lead to 10 test instances. The version of CPLEX is cplex 20.1.0. The upper limits of CPU times for CPLEX are set as 3600 s and 1 day (86 400 s) for each instance. Among them, 3600 s is the acceptable time in practice, and 1 day is for a better test of model performance. When solving with CPLEX, the proposed lower bound is used to determine the critical value of completion time. This method cannot guarantee the optimal result, but it gets satisfactory results in actual production.

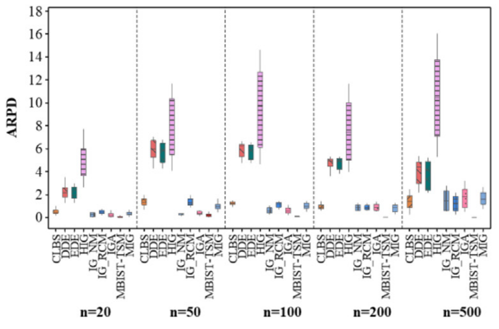
Optimal solutions of small-scale instances are obtained via CPLEX, which could verify the correctness of the model. The computational results are presented in Table 5. The LB represents the lower bound. As observed in Table 5, 5 instances (50%) out of 10 are improved via the MBIST-TSM, 3 instances (30%) out of 10 are even, and 2 instance (20%) out of 10 is inferior. Compared with the experimental results, it is equally important that the solution time of the MBIST-TSM is much shorter than CPLEX. For instances 3, 6–10, the results of the MBIST-TSM are equal to the LB of the problems, which means that the MBIST-TSM has obtained the optimal solutions for these instances. For other instances, the MBIST-TSM is still demonstrated to be effective, since the MBIST-TSM solution is much close to the LB of the problems. When the maximum CPU time of CPLEX is set to one day, certain experimental results are not improved, e.g., instances 5–9.

6.3. Comparison of the presented heuristic

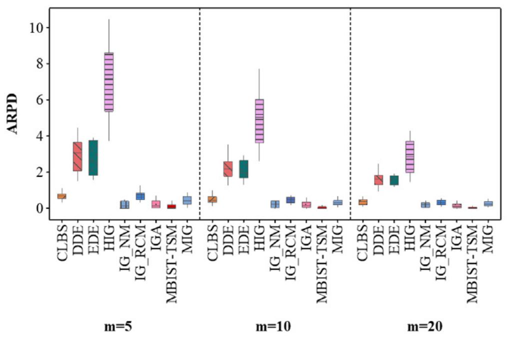
Certain heuristics for the DFSPs and the SDSTs are employed to compare with the MBIST to verify its performance since the SDST/DBFSP has not been considered. These heuristics contain DSPT, DLPT, DLS, and DNEH (Zhang et al., 2018). The implementation details are shown in the Appendix section. All of the compared heuristics are utilized



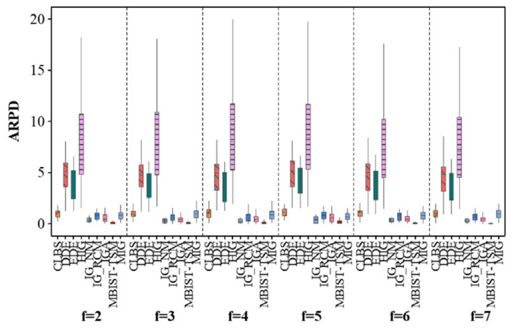
(a)



(b)



(c)



(d)

Fig. 14. The box plots of all algorithms. (a) at different scenarios. (b) at different jobs. (c) at different machines. (d) at different factories.

to address the SDST/DBFSP. The 2880 instances are employed for this comparison. The ARPDs and average CPU time for all compared heuristics are shown in Tables 6 and 7. These instances are grouped according to the different number of factories. Each cell contains an average of 120 instances per value of f. The best value of each group is highlighted with boldface in Tables 6 and 7.

As observed in Tables 6 and 7, MBIST yields a relatively lower ARPD than other heuristics. The ARPDs value of the MBIST gradually increases with the increase of the number of factories, but it is still less than all other comparison heuristics, which means that the performance of the MBIST is better than other comparisons heuristics. On the other hand, it also shows that the significance of the MBIST compared with other heuristics decreases with the increase in the number of factories. Whereas the average CPU time of the first three heuristics, i.e., DSPT, DLPT, and DLS, is lower than other heuristics in Tables 6 and 7. This is because these heuristics only allocate jobs to factories and lack the improvement process based on the characteristics of the problem. Although the MBIST takes slightly more computational cost, it obtains the initial solution with significantly improved performance, which lays a good foundation for the later iterative process. Additionally, the box plot of all heuristics with different setting time levels is illustrated in Fig. 10. The MBIST has the best performance under different setting time levels. As the increment of setting time levels, the setting time exceeds the processing time of the jobs (SSD125). According to the problem characteristics mentioned in Section 5.1 of the SDST/DBFSP, the idle times and blocking times are not effectively reduced via placing the job with the longest setting times in the first position, under the circumstances. At this time, the key to improving the heuristic performance is the second step of the MBIST.

The probability plot of different heuristics with a 95% confidence interval is shown in Fig. 11. In the probability plot, each set of data contains three lines, the middle line represents the fitting distribution line of the data, and the dashed lines on both sides represent the confidence interval. If a set of data is closely distributed along the middle line and close to the left area of the graph, it indicates that the heuristic outperforms the other methods. As observed in Fig. 11, the MBIST is correspondingly stable and compact. The proposed heuristic is the best method among all the comparison methods.

6.4. Performance analysis of each component of the MBIST-TSM

In this section, the performance of each component of the proposed MBIST-TSM is investigated to demonstrate their contributions. The MBIST-TSM contains four main improvement components: (1) the initialization method (MBIST); (2) the first stage of the TSM; (3) the second stage of the TSM; (4) the third stage of the TSM. Hence, four variants of the MBIST-TSM are proposed to validate the effectiveness: the MBIST-TSM with random initialization (MBIST-TSM (RI)), the MBIST-TSM without the first stage of the TSM (MBIST-TSM(NF)), the MBIST-TSM without the second stage of the TSM (MBIST-TSM(NS)), and the MBIST-TSM without the third stage of the TSM (MBIST-TSM(NT)). Note that, each variant only changes one component of the complete MBIST-TSM. Moreover, the benchmark instances proposed by Naderi and Ruiz (2010) are tested to prove the effectiveness of each component. The maximum elapsed CPU time of $t = 10 \times n \times m \times f$ milliseconds is chosen as the termination criterion. Each variant is run 10 times on each instance independently. The computational results are reposted in Table 8. The groups with several factories of 3, 5, and 7 are shown. The best value for each group is highlighted in boldface.

As observed in Table 8, among all four scenarios, the overall average ARPD of the MBIST-TSM outperforms the variants. Specifically, the performance of the MBIST-TSM is significant in all instances compared with the MBIST-TSM(RI), which indicates that the proposed MBIST plays an important role in the MBIST-TSM. The introduction of a construction heuristic based on SDST/DBFSP characteristics has a significant impact on the performance of the MBIST-TSM. The MBIST-TSM outperforms the MBIST-TSM(NF), which reveals that constructing an external sequence set in SDST/DBFSP to guide the search direction of other individuals is meaningful. Additionally, the search scheme constructed by employing the characteristics of the SDST/DBFSP is effective in the first stage of the MBIST-TSM. Moreover, the MBIST-TSM(NT) is inferior to the MBIST-TSM, which demonstrates that the information of different leaders to construct candidate solutions is meaningful. To be specific, the improvement of critical factories in SDST/DBFSP directly affects the improvement of the final optimization

Table 9
ARPD values for $f = 2, 3, 4$ (SSD10 and SSD50).

f	$n \times m$	SSD10									SSD50								
		IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM	IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM
	20 × 5	0.013	4.695	0.103	0.457	1.917	2.104	**0.011**	0.371	0.135	0.018	5.995	0.365	0.622	2.226	1.747	**0.016**	0.308	0.303
	20 × 10	0.002	3.208	0.099	0.253	1.570	1.855	**0.002**	0.152	0.085	0.010	4.579	0.264	0.396	1.898	2.341	**0.009**	0.237	0.264
	20 × 20	0.005	1.593	0.162	0.265	1.285	1.761	**0.004**	0.159	0.138	0.039	2.420	0.183	0.309	1.627	1.845	**0.035**	0.186	0.208
	50 × 5	0.170	7.048	0.636	0.975	5.249	6.030	**0.153**	0.585	0.198	0.279	10.160	0.909	1.217	6.536	7.078	**0.251**	0.730	0.320
	50 × 10	0.332	4.307	0.514	0.904	5.227	4.623	0.299	0.542	**0.273**	0.219	6.390	0.937	1.244	5.844	5.167	**0.197**	0.746	0.326
	50 × 20	0.177	3.425	0.560	0.740	4.488	4.302	**0.159**	0.444	0.209	0.115	4.491	0.585	0.903	4.534	4.662	**0.104**	0.542	0.188
2	100 × 5	0.509	7.413	0.537	1.044	4.617	4.056	0.458	0.626	**0.071**	0.837	12.234	1.134	1.279	6.099	6.181	0.753	0.768	**0.044**
	100 × 10	0.542	5.246	0.741	1.170	5.156	4.527	0.488	0.702	**0.091**	0.780	8.050	0.926	1.223	6.207	5.300	0.702	0.734	**0.065**
	100 × 20	0.611	3.536	0.515	0.996	5.181	5.655	0.550	0.597	**0.049**	0.515	5.178	0.395	0.867	5.528	5.919	0.464	0.520	**0.104**
	200 × 10	0.698	6.540	0.704	1.169	3.848	3.595	0.628	0.701	**0.057**	1.088	9.471	1.037	1.223	5.066	5.755	0.979	0.734	**0.000**
	200 × 20	0.591	4.304	0.494	0.738	3.837	3.216	0.531	0.443	**0.024**	0.866	6.312	0.700	0.890	5.191	4.176	0.779	0.534	**0.010**
	500 × 20	0.589	6.094	0.774	0.761	2.426	2.058	0.530	0.457	**0.028**	1.487	8.769	1.310	1.234	4.117	4.380	1.338	0.740	**0.000**
	Average	0.353	4.784	0.487	0.789	3.733	1.917	0.318	0.474	**0.113**	0.521	7.004	0.729	0.951	4.573	2.226	0.469	0.570	**0.153**
	20 × 5	**0.000**	4.481	0.226	0.499	1.735	1.639	**0.000**	**0.000**	0.051	**0.000**	6.308	0.458	0.706	2.592	2.853	**0.000**	1.712	0.181
	20 × 10	**0.000**	3.459	0.300	0.478	1.804	1.638	**0.000**	0.287	0.115	**0.000**	4.252	0.266	0.430	1.968	3.016	**0.000**	0.258	0.159
	20 × 20	0.003	1.692	0.139	0.161	1.210	1.392	**0.002**	0.097	0.045	0.013	2.335	0.143	0.242	1.410	0.921	**0.011**	0.145	0.070
	50 × 5	0.208	7.850	0.901	1.191	4.708	4.337	**0.187**	0.714	0.247	0.087	10.700	1.156	1.525	5.859	6.793	**0.078**	0.915	0.547
	50 × 10	0.138	4.557	0.753	0.950	4.372	4.762	**0.124**	0.570	0.278	0.091	6.613	0.982	1.304	5.356	4.829	**0.082**	0.782	0.363
	50 × 20	0.118	3.419	0.648	0.812	3.929	4.137	**0.106**	0.487	0.275	0.111	4.669	0.742	0.919	4.109	4.465	**0.099**	0.552	0.320
3	100 × 5	0.387	8.033	0.803	0.865	4.828	5.171	0.349	0.519	**0.130**	0.487	12.414	1.302	1.148	6.121	7.786	0.438	0.689	**0.122**
	100 × 10	0.341	5.105	0.608	0.753	4.756	5.067	0.307	0.452	**0.100**	0.447	7.754	0.972	1.030	5.768	5.918	0.402	0.618	**0.110**
	100 × 20	0.334	3.838	0.448	0.868	4.795	4.293	0.300	0.521	**0.138**	0.294	5.099	0.669	0.819	5.299	4.530	0.265	0.491	**0.100**
	200 × 10	0.577	6.203	0.774	0.640	3.624	3.213	0.519	0.384	**0.054**	0.967	9.334	1.168	1.099	5.395	5.474	0.871	0.659	**0.006**
	200 × 20	0.621	4.184	0.388	0.593	4.046	2.971	0.559	0.356	**0.096**	0.602	6.061	0.609	0.851	4.917	4.570	0.542	0.510	**0.013**
	500 × 20	0.546	5.652	1.048	0.268	2.280	2.015	0.491	0.161	**0.023**	1.340	8.218	1.356	0.922	4.063	5.106	1.206	0.553	**0.000**
	Average	0.273	4.873	0.586	0.673	3.507	1.735	0.245	0.404	**0.129**	0.370	6.980	0.818	0.916	4.405	2.592	0.333	0.550	**0.166**
	20 × 5	0.012	3.746	0.226	0.357	1.580	2.153	0.011	**0.000**	0.029	**0.000**	6.266	0.381	0.749	2.806	3.310	**0.000**	1.394	0.146
	20 × 10	0.003	2.802	0.133	0.214	1.310	0.862	**0.002**	0.128	0.029	**0.000**	3.835	0.324	0.553	1.822	2.691	**0.000**	0.332	0.193
	20 × 20	**0.000**	1.956	0.096	0.126	1.321	1.885	**0.000**	0.076	0.050	0.008	2.422	0.200	0.275	1.422	2.183	**0.007**	0.165	0.092
	50 × 5	0.105	8.435	0.967	2.243	4.605	4.022	**0.094**	1.346	0.379	0.215	11.857	1.055	1.914	5.950	5.455	**0.193**	1.148	0.458
	50 × 10	0.248	5.275	0.926	1.931	4.264	3.113	**0.223**	1.159	0.240	0.209	6.939	0.775	1.801	4.962	3.946	**0.188**	1.081	0.288
	50 × 20	0.103	3.584	0.543	1.242	3.509	2.966	**0.093**	0.745	0.134	0.206	4.797	0.691	1.394	3.759	3.915	**0.185**	0.836	0.249
4	100 × 5	0.447	9.026	0.669	0.661	4.997	4.343	0.402	0.397	**0.142**	0.529	13.635	1.145	1.022	6.097	7.782	0.476	0.613	**0.142**
	100 × 10	0.242	5.740	0.489	0.700	4.932	3.575	**0.218**	0.420	0.225	0.481	8.507	0.790	0.946	6.079	5.887	0.432	0.568	**0.160**
	100 × 20	0.368	4.315	0.420	0.629	5.039	4.289	0.331	0.377	**0.187**	0.275	5.884	0.699	0.759	5.331	5.408	0.247	0.455	**0.142**
	200 × 10	0.614	6.882	0.666	0.622	3.691	2.224	0.552	0.373	**0.077**	0.855	10.050	1.066	0.788	5.321	5.536	0.769	0.473	**0.017**
	200 × 20	0.589	4.771	0.311	0.624	4.360	3.376	0.530	0.374	**0.096**	0.651	6.737	0.582	0.627	5.289	5.469	0.586	0.376	**0.041**
	500 × 20	0.642	6.104	1.138	0.412	2.184	1.837	0.578	0.247	**0.088**	1.368	8.819	1.430	1.024	4.097	4.201	1.231	0.614	**0.005**
	Average	0.281	5.220	0.549	0.813	3.483	1.580	0.253	0.488	**0.140**	0.400	7.479	0.761	0.988	4.411	2.806	0.360	0.593	**0.161**

goal. The third stage of the MBIST-TSM strengthens the information interaction between individuals and speeds up the convergence of the algorithm. Additionally, the MBIST-TSM(NS) is inferior to the MBIST-TSM. On the one hand, the deep local search strategy for critical factories is adopted to improve the quality of solutions. On the other hand, the acceptance criterion of simulated annealing is employed to maintain the diversity of the swarm and balance the exploration and exploitation of the MBIST-TSM.

The box plots of the MBIST-TSM and its variants are shown to confirm the conclusions above in Fig. 12. The length of the box for the MBIST-TSM is less than its variants, and the box for the MBIST-TSM is located below all other boxes, which indicates that the differences between the MBIST-TSM and its variants are significant. In other words, the MBIST-TSM is significantly better than its variants in all scenarios. Particularly, the confidence intervals of the MBIST-TSM are far from the MBIST-TSM(RI), which also confirms the search scheme combined with the characteristics of the problem plays a critical role in the MBIST-TSM.

The comparison results of the MBIST-TSM and its variants at different factories are plotted in Fig. 13, respectively, in terms of convergence speed. To ensure the fairness of the experiment, the initial solutions of all algorithms are generated by the MBIST. Hence, MBIST-TSM(RI) is not compared in this experiment. The horizontal axis represents the processing times (unit: seconds), and the vertical axis represents the makespan of each iteration. It is observed that the MBIST-TSM has more promising performance than its variants at different factories. The MBIST-TSM has a convergence speed compared with other variants at $f = 2, 4, 5$, and 6, which indicates the outstanding advantage of the MBIST-TSM in convergence performance. Although the advantage of the MBIST-TSM is not obvious compared with the MBIST-TSM(NF) at $f = 3$ and 7, the MBIST-TSM maintains a continuous downward trend and obtains a higher precision solution than that of any of the three comparison variants. The results verify that the first stage of the MBIST-TSM indeed can help a particle detect a more promising position via the neighborhood insertion operation for the critical factory when the swarm has been trapped into the local optimum. Further, the MBIST-TSM also achieves higher accurate solutions than the MBIST-TSM(NS) at different factories, which indicates that the neighborhood operator and the acceptance criterion in the second stage of the MBIST-TSM are effective to improve the diversity of the population. Nevertheless, the MBIST-TSM not only has a higher convergence speed but also achieves more accurate solutions than that of the MBIST-TSM(NT) at different factories. Therefore, the information-sharing capability of the different leaders is improved by the third stage of the MBIST-TSM.

In conclusion, compared with any of the incomplete algorithms of the MBIST-TSM, the MBIST-TSM has better optimization performance. Hence, the MBIST-TSM can effectively maximize the advantages of different strategies and get the best optimization performance. This also means that all the main improvements in this paper are effective and indispensable and that the MBIST-TSM can get a better balance between exploration and exploitation.

Table 10
ARPD values for $f = 2, 3, 4$. (SSD100 and SSD125).

f	$n \times m$	SSD100									SSD125								
		IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM	IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM
	20 × 5	0.037	7.814	0.507	0.623	3.566	4.538	0.034	**0.000**	0.453	0.056	8.830	0.524	0.596	3.907	2.973	**0.051**	0.235	0.365
	20 × 10	0.026	6.093	0.350	0.681	2.503	2.885	**0.023**	0.408	0.429	0.059	6.828	0.422	0.546	2.945	3.032	**0.053**	0.328	0.426
	20 × 20	0.026	3.762	0.304	0.444	1.796	2.225	**0.024**	0.266	0.221	0.039	4.057	0.537	0.549	1.927	1.834	**0.035**	0.329	0.441
	50 × 5	0.329	14.846	1.382	1.527	7.976	7.271	**0.296**	0.916	0.492	0.410	16.045	1.521	1.791	8.066	7.950	**0.369**	1.075	0.417
	50 × 10	0.331	9.109	0.905	1.347	6.563	4.826	**0.298**	0.808	0.307	0.343	10.568	1.063	1.271	6.784	6.763	**0.309**	0.762	0.309
	50 × 20	0.262	6.233	0.884	1.053	5.431	4.965	0.236	0.632	**0.214**	0.250	7.136	1.020	1.218	5.573	5.534	**0.225**	0.731	0.274
2	100 × 5	0.923	18.202	1.083	1.109	6.040	7.792	0.830	0.665	**0.110**	1.247	20.748	1.249	1.709	6.819	6.439	1.122	1.025	**0.029**
	100 × 10	0.964	11.791	1.104	1.372	6.294	7.463	0.867	0.823	**0.084**	1.076	13.609	1.137	1.262	6.543	9.645	0.968	0.757	**0.050**
	100 × 20	0.796	7.480	0.914	1.141	5.996	5.023	0.717	0.684	**0.082**	0.707	8.840	0.934	1.145	5.562	4.205	0.636	0.687	**0.039**
	200 × 10	1.619	14.449	1.338	1.564	5.570	5.576	1.457	0.939	**0.000**	1.447	16.691	1.402	1.500	5.578	4.545	1.302	0.900	**0.000**
	200 × 20	1.137	9.339	1.038	1.242	5.089	4.727	1.023	0.745	**0.000**	1.273	10.794	1.109	1.253	4.923	5.701	1.146	0.752	**0.000**
	500 × 20	2.187	12.636	1.842	1.718	4.695	3.745	1.968	1.031	**0.000**	2.739	14.764	2.315	2.176	5.200	5.424	2.465	1.305	**0.000**
	Average	0.720	10.146	0.971	1.152	5.127	3.566	0.648	0.691	**0.199**	0.804	11.576	1.103	1.251	5.319	3.907	0.723	0.751	**0.196**
	20 × 5	0.020	8.511	0.694	1.248	3.688	4.813	**0.018**	0.722	0.656	0.012	9.484	0.833	0.998	3.837	3.924	**0.010**	0.336	0.487
	20 × 10	**0.000**	5.646	0.464	0.584	2.746	1.513	**0.000**	0.350	0.343	0.003	5.800	0.680	0.722	2.565	2.611	**0.003**	0.433	0.443
	20 × 20	0.019	3.352	0.320	0.429	1.689	1.515	**0.017**	0.257	0.209	0.033	3.556	0.463	0.482	1.833	2.303	**0.030**	0.289	0.411
	50 × 5	0.244	14.843	1.616	1.751	7.845	4.504	**0.220**	1.051	0.536	0.169	16.567	1.666	1.812	8.185	8.298	**0.152**	1.087	0.664
	50 × 10	0.101	9.734	1.249	1.601	6.195	6.166	**0.091**	0.960	0.536	0.147	10.959	1.575	1.656	6.415	5.503	**0.133**	0.994	0.719
	50 × 20	0.110	6.380	1.024	1.181	4.625	4.842	**0.099**	0.709	0.412	0.125	7.126	1.128	1.328	4.951	4.168	**0.113**	0.797	0.404
3	100 × 5	0.580	18.067	1.248	1.067	6.458	8.146	0.522	0.640	**0.195**	1.052	21.216	1.556	1.624	6.874	5.974	0.947	0.974	**0.049**
	100 × 10	0.661	11.874	1.367	1.440	6.387	6.112	0.595	0.864	**0.091**	0.479	13.138	1.336	1.245	6.281	8.860	0.431	0.747	**0.156**
	100 × 20	0.433	7.524	1.035	1.010	5.783	5.041	0.390	0.606	**0.155**	0.390	8.347	1.125	1.107	5.713	5.608	0.351	0.664	**0.128**
	200 × 10	1.398	14.613	1.456	1.214	5.542	4.855	1.258	0.728	**0.008**	1.369	16.754	1.423	1.091	5.434	5.346	1.232	0.655	**0.003**
	200 × 20	0.896	9.146	1.012	0.874	5.290	4.576	0.807	0.524	**0.013**	1.089	10.198	0.960	0.904	4.782	4.491	0.980	0.542	**0.008**
	500 × 20	2.176	12.512	2.076	1.621	4.778	4.744	1.958	0.972	**0.000**	2.497	14.320	2.247	1.956	4.957	4.748	2.247	1.174	**0.000**
	Average	0.553	10.184	1.130	1.168	5.085	3.688	0.498	0.701	**0.263**	0.614	11.456	1.249	1.244	5.152	3.837	0.553	0.746	**0.289**
	20 × 5	**0.000**	8.051	0.722	0.829	3.247	2.112	**0.000**	1.659	0.443	0.054	10.456	0.875	1.112	4.441	3.441	**0.049**	2.331	0.722
	20 × 10	0.021	5.229	0.471	0.687	2.173	2.295	**0.019**	0.412	0.345	0.005	6.925	0.613	0.937	3.187	1.495	**0.005**	0.562	0.610
	20 × 20	**0.000**	3.231	0.271	0.280	1.657	1.773	**0.000**	0.168	0.142	0.101	4.067	0.435	0.593	2.126	1.866	**0.091**	0.356	0.430
	50 × 5	0.220	15.466	1.437	1.947	7.646	8.588	**0.198**	1.168	0.521	0.282	17.676	1.399	1.714	8.227	8.468	**0.253**	1.028	0.590
	50 × 10	0.111	10.192	1.263	1.772	5.846	4.944	**0.100**	1.063	0.599	0.277	11.261	1.136	1.808	7.046	5.250	**0.249**	1.085	0.520
	50 × 20	0.111	6.580	0.873	1.383	4.283	4.695	**0.100**	0.830	0.260	0.145	7.411	0.947	1.204	5.346	6.434	**0.130**	0.723	0.292
4	100 × 5	0.816	19.986	1.333	1.212	6.361	6.226	0.734	0.727	**0.199**	1.411	22.837	1.468	1.334	6.675	6.701	1.269	0.801	**0.054**
	100 × 10	0.535	12.541	1.298	1.285	6.472	6.203	0.481	0.771	**0.201**	0.900	14.459	1.371	1.320	6.633	8.712	0.810	0.792	**0.112**
	100 × 20	0.254	7.961	0.975	1.061	5.722	5.101	0.228	0.637	**0.173**	0.693	9.165	0.897	1.148	6.078	5.075	0.623	0.689	**0.078**
	200 × 10	1.434	15.143	1.498	1.256	5.723	5.753	1.291	0.754	**0.000**	1.731	18.085	1.701	1.505	5.425	5.447	1.558	0.903	**0.020**
	200 × 20	0.929	9.601	0.928	0.889	5.175	4.864	0.836	0.534	**0.008**	1.354	11.414	1.148	1.061	5.052	5.319	1.219	0.637	**0.000**
	500 × 20	2.332	13.208	2.252	1.772	4.819	3.967	2.099	1.063	**0.000**	2.885	15.604	2.630	2.187	5.047	4.904	2.597	1.312	**0.000**
	Average	0.564	10.599	1.110	1.198	4.927	3.247	0.507	0.719	**0.241**	0.820	12.447	1.218	1.327	5.440	4.441	0.738	0.796	**0.286**

6.5. *Effectiveness of the proposed MBIST-TSM*

The effectiveness of the MBIST-TSM is further verified by comparing eight algorithms for SDST/DBFSP. The comparison algorithms include modified iterated greedy algorithm (MIG) (Lin and Ying, 2013), constraint-based local search (CBLS) algorithm (Riahi et al., 2021), the revised artificial immune system (RAIS) (Lin and Ying, 2013), discrete differential evolution algorithm (DDE) (Zhang et al., 2018), iterated greedy algorithm (IGA) (Ribas et al., 2017), ensemble discrete differential evolution (EDE) algorithm (Zhao et al., 2020c), IG_NM (Miyata and Nagano, 2022), and IG_RCM (Ribas et al., 2021). The MIG, HIG, DDE, IGA, and EDE are effective methods to solve other flowshop scheduling problems. Other three methods consider SDSTs, so the methods are selected to compare with the proposed MBIST-TSM for SDST/DBFSP. Each algorithm is run 5 times independently. It is noticed that the initial solutions of the nine algorithms are constructed by employing the proposed MBIST in Section 5.1 to maintain the fairness of algorithms comparison.

The ARPDs of the algorithms are shown in Tables 9–12. The algorithm with the minimum ARPD outperforms other algorithms. As observed in Tables 9–12, the overall performance of the MBIST-TSM has been significantly improved. It is worth noting that IGA and IG_NM obtain smaller ARPD values than the MBIST-TSM on certain small-scale problems, whereas the overall performance is still worse than the MBIST-TSM. Additionally, according to the observations in the tables, it is found that the performance of the MBIST-TSM is related to the setting time levels and problem scale. Therefore, the box plots of different scenarios and problem scales are displayed in Fig. 14. The box plots directly show the discrete degree of the results, and thereby further reflect the stability of the algorithms. As shown in Fig. 14, the length of the box (upper quartile minuses lower quartile) for the MBIST-TSM is short than other algorithms. Therefore, the results of the MBIST-TSM are concentrated. In other words, the proposed MBIST-TSM has stable performance on instances with different scenarios and problem scales. HIG is quite sensitive to the change of setting time and the number of machines, according to Fig. 14(a) and (c). In Fig. 14(c) and (d), the performance of the MBIST-TSM is gradually improved with the increase in the number of machines and factories. Overall, the MBIST-TSM is superior to other comparison algorithms in different scenarios and problem scales.

The Friedman-test is employed to make a statistical comparison between the MBIST-TSM and other algorithms in Tables 13–16, where CN is the number of cases. Friedman-test is a non-parametric statistical test for determining significant differences in multiple(related) samples (Huang et al., 2020). The p-value computed through the statistic of the test is close to 0, which is obtained with a level of significance $\alpha = 0.05$. The results of the Friedman test are correspondingly shown in Fig. 15. As observed in Fig. 15, the mean rank of the MBIST-TSM is the minimum except for some small-scale problems. Hence, the performance of the MBIST-TSM is excellent compared with the others at 90% and 95% confidence intervals.

The Wilcoxon test is utilized to verify the performance difference between the MBIST-TSM and other algorithms. Expressly, five pairwise comparisons between the MBIST-TSM and others are designed to better

Table 11
ARPD Values for $f = 5, 6, 7$. (SSD10 and SSD50).

f	$n \times m$	SSD10									SSD50								
		IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM	IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM
	20 × 5	0.013	5.239	0.115	0.669	2.575	2.847	**0.011**	0.371	0.210	0.053	6.954	0.350	0.765	2.980	2.055	**0.048**	0.308	0.383
	20 × 10	0.018	3.238	0.084	0.343	2.164	3.004	**0.017**	0.206	0.183	0.039	5.103	0.248	0.533	2.467	2.341	**0.035**	0.320	0.305
	20 × 20	0.021	1.710	0.167	0.375	1.586	2.216	**0.019**	0.225	0.137	0.052	2.524	0.242	0.419	2.067	2.050	**0.046**	0.251	0.222
	50 × 5	0.489	8.387	0.449	0.991	5.564	6.298	0.440	0.594	**0.258**	0.432	11.188	0.660	1.493	6.642	7.575	0.389	0.896	**0.252**
	50 × 10	0.542	4.821	0.431	0.710	5.448	5.801	0.488	0.426	**0.129**	0.368	6.955	0.518	1.137	6.121	5.481	0.331	0.682	**0.299**
	50 × 20	0.424	3.573	0.468	0.779	4.985	3.847	0.382	0.467	**0.131**	0.347	4.928	0.397	0.782	5.266	5.905	0.312	0.469	**0.172**
5	100 × 5	0.590	8.689	0.456	1.428	4.477	4.299	0.531	0.857	**0.134**	0.909	13.491	0.983	1.163	6.218	6.785	0.818	0.698	**0.057**
	100 × 10	0.716	5.804	0.507	1.255	5.153	5.254	0.644	0.753	**0.108**	0.866	8.369	0.821	1.153	6.369	5.526	0.780	0.692	**0.050**
	100 × 20	0.786	4.064	0.292	0.713	5.124	5.129	0.707	0.428	**0.125**	0.612	5.589	0.332	0.862	5.688	5.890	0.551	0.517	**0.095**
	200 × 10	0.675	7.200	0.784	1.228	3.673	2.985	0.607	0.737	**0.041**	1.142	10.887	1.041	1.384	5.047	5.207	1.027	0.830	**0.007**
	200 × 20	0.630	4.651	0.381	0.844	3.629	2.732	0.567	0.506	**0.089**	0.843	6.826	0.571	1.018	4.999	4.221	0.759	0.611	**0.007**
	500 × 20	0.742	6.995	1.012	0.992	2.330	1.825	0.668	0.595	**0.042**	1.736	9.682	1.493	1.363	4.157	4.421	1.563	0.818	**0.000**
	Average	0.470	5.364	0.429	0.860	3.892	2.575	0.423	0.516	**0.132**	0.617	7.708	0.638	1.006	4.835	2.980	0.555	0.604	**0.154**
	20 × 5	**0.000**	4.483	0.050	0.528	1.521	1.609	**0.000**	**0.000**	0.080	0.031	5.811	0.308	0.537	2.029	2.678	**0.028**	0.824	0.176
	20 × 10	0.002	2.637	0.065	0.147	1.287	0.972	**0.002**	0.088	0.080	0.003	3.723	0.200	0.340	1.709	1.586	**0.003**	0.204	0.174
	20 × 20	0.005	1.499	0.096	0.145	0.956	1.193	**0.004**	0.087	0.050	0.008	2.101	0.153	0.211	1.303	1.128	**0.007**	0.127	0.110
	50 × 5	0.105	6.639	0.809	1.141	5.394	6.336	**0.094**	0.684	0.253	0.219	9.713	1.087	1.319	6.746	7.549	**0.197**	0.791	0.290
	50 × 10	0.120	4.098	0.657	1.041	5.144	5.394	**0.108**	0.625	0.295	0.146	6.035	0.879	1.104	5.540	4.814	**0.131**	0.663	0.227
	50 × 20	0.105	3.285	0.683	0.726	4.474	3.903	**0.094**	0.436	0.168	0.092	4.225	0.608	0.874	4.599	4.646	**0.083**	0.525	0.249
6	100 × 5	0.419	7.081	0.682	1.041	4.702	3.741	0.377	0.624	**0.058**	0.651	11.557	0.939	1.122	6.007	5.715	0.586	0.673	**0.038**
	100 × 10	0.450	4.664	0.837	1.176	5.163	4.734	0.405	0.706	**0.063**	0.613	7.130	0.999	1.282	6.064	6.579	0.551	0.769	**0.054**
	100 × 20	0.516	3.400	0.469	1.019	4.871	5.946	0.465	0.611	**0.062**	0.567	4.880	0.671	1.062	5.344	5.766	0.510	0.637	**0.059**
	200 × 10	0.578	5.720	0.669	0.974	3.914	3.300	0.520	0.585	**0.025**	1.211	8.790	1.071	1.276	5.044	5.794	1.090	0.766	**0.000**
	200 × 20	0.603	4.014	0.495	0.754	3.968	3.174	0.543	0.453	**0.003**	0.909	5.938	0.801	0.941	5.058	4.359	0.819	0.564	**0.012**
	500 × 20	0.569	5.756	0.757	0.750	2.607	2.422	0.512	0.450	**0.018**	1.436	8.254	1.228	1.235	4.108	3.745	1.292	0.741	**0.000**
	Average	0.289	4.440	0.522	0.787	3.667	1.521	0.260	0.472	**0.096**	0.491	6.513	0.745	0.942	4.463	2.029	0.442	0.565	**0.116**
	20 × 5	**0.000**	3.890	0.160	0.340	1.514	2.131	**0.000**	0.164	0.046	**0.000**	5.711	0.376	0.644	2.339	3.138	**0.000**	**0.000**	0.201
	20 × 10	**0.000**	3.610	0.215	0.341	1.621	1.092	**0.000**	0.205	0.086	**0.000**	3.918	0.226	0.403	1.770	**0.000**	**0.000**	0.242	0.137
	20 × 20	**0.000**	1.470	0.080	0.125	0.968	0.928	**0.000**	0.075	0.015	0.001	2.106	0.112	0.122	1.173	1.106	**0.001**	0.073	0.034
	50 × 5	0.072	7.299	1.040	1.383	4.936	6.026	**0.065**	0.830	0.309	0.051	10.434	1.231	1.470	6.334	7.649	**0.046**	0.882	0.501
	50 × 10	0.054	4.452	0.905	1.087	4.578	5.294	**0.049**	0.652	0.305	0.032	6.568	1.028	1.371	5.570	6.001	**0.029**	0.823	0.540
	50 × 20	0.067	3.346	0.795	0.815	4.021	3.560	**0.060**	0.489	0.269	0.039	4.384	0.896	0.978	4.446	4.435	**0.036**	0.587	0.361
7	100 × 5	0.340	7.304	0.879	0.855	4.846	4.398	0.306	0.513	**0.071**	0.248	11.673	1.267	1.229	6.274	5.025	0.223	0.737	**0.118**
	100 × 10	0.253	4.934	0.738	0.925	4.820	4.981	0.228	0.555	**0.120**	0.254	7.117	1.161	1.162	5.746	6.117	0.228	0.697	**0.108**
	100 × 20	0.202	3.551	0.618	0.854	4.570	4.725	0.182	0.512	**0.098**	0.283	4.845	0.900	1.074	5.095	5.738	0.255	0.645	**0.105**
	200 × 10	0.621	5.778	0.705	0.711	3.651	2.924	0.558	0.427	**0.014**	1.078	8.596	1.102	1.068	5.257	5.506	0.971	0.641	**0.000**
	200 × 20	0.545	3.971	0.481	0.675	4.106	3.115	0.490	0.405	**0.030**	0.750	5.590	0.792	0.842	4.847	5.458	0.675	0.505	**0.008**
	500 × 20	0.520	5.316	0.905	0.312	2.458	1.899	0.468	0.187	**0.032**	1.213	7.431	1.208	0.859	4.121	4.889	1.092	0.515	**0.003**
	Average	0.223	4.577	0.627	0.702	3.508	1.514	0.201	0.421	**0.116**	0.329	6.531	0.858	0.935	4.414	2.339	0.296	0.561	**0.176**

visualize the significance of the proposed heuristics and search operators. In principle, the Wilcoxon test utilizes the sign test of paired observation data to deduce the probability when the difference appears (Huang et al., 2020). Furthermore, whether the difference between the two means is significant is also shown by the method. The results of the Wilcoxon sign test are shown in Table 17. The $R+$ represents that the results are outstanding compared with the other algorithms. $R-$ is the opposite. For each pair, the sum of $R+$, $R-$ and bonding values is equal to the total case numbers. In the Wilcoxon test, if the p-value is less than the α, which means there are significant differences in the pairwise algorithms. From Table 17, it is found that all the p-values are less than 0.05. Hence, the MBIST-TSM is significantly better than other algorithms.

7. Conclusion and future work

In this article, a type of realistic scheduling problem with the sequence-dependent setup times in the DBFSP environment is considered and solved via a discrete heuristic and three-stage meta-heuristic. The objective of the problem is to minimize the makespan of the system. The problem-specific knowledge of the SDST/DBFSP is combined with the search process of the MBIST-TSM. The main advantages of the proposed algorithm are as follows: (1) a problem-specific knowledge of the SDST/DBFSP is explored; (2) a heuristic that considers the problem-specific knowledge is proposed to minimize the blocking times and idle times created by SDSTs (MBIST); (3) a three-stage meta-heuristic is investigated to enhance the searching ability; (4) a speedup method for evaluating the insert neighborhood is proposed to reduce the complexity of evaluating the entire insertion neighborhood from $O(fmn^3)$ to $O(fmn^2)$, where f, n, and m represent the number of factories, machines, and jobs, respectively.

The effectiveness of the MBIST is verified by comparing it with four advanced heuristics. The critical parameters of the MBIST-TSM are calibrated to achieve the best performance by the DOE technique. Additionally, the effectiveness of each improvement strategy in the MBIST-TSM is verified through certain experiments. Ultimately, the MBIST-TSM is evaluated based on 2880 benchmark instances and 10 sets of actual production instances in the aluminum industry. From the results for the instances, the MBIST-TSM indeed obtains improvements over other algorithms and CPLEX.

Despite the excellent performance, the following issues need to be solved in future work: (1) the proposed search operator based on SDST/DBFSP characteristics can flexibly switch to make the operator adapt to different search stages; (2) the scheduling problems should be extended to adapt to certain widespread industrial applications; (3) the MBIST-TSM should be applied to solve certain complex scheduling problems (e.g., the job shop scheduling problems, the lot-streaming

Table 12
ARPD Values for $f = 5, 6, 7$. (SSD100 and SSD125).

f	$n \times m$	SSD100									SSD125								
		IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM	IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM	MBIST-TSM
5	20 × 5	0.137	9.450	0.526	0.736	4.062	5.286	**0.124**	2.302	0.621	0.049	9.880	0.616	1.095	4.411	4.304	**0.044**	0.235	0.520
	20 × 10	0.060	6.549	0.390	0.772	3.240	3.318	**0.054**	0.463	0.528	0.074	7.694	0.595	0.993	3.527	3.924	**0.066**	0.596	0.561
	20 × 20	0.050	4.189	0.356	0.587	2.309	1.907	**0.045**	0.352	0.450	0.105	4.295	0.433	0.654	2.479	2.645	**0.095**	0.393	0.446
	50 × 5	0.618	16.274	0.884	1.484	8.128	5.564	0.556	0.891	**0.343**	0.705	17.876	1.118	1.673	7.964	7.065	0.635	1.004	**0.336**
	50 × 10	0.514	10.731	0.741	1.141	6.913	6.533	0.463	0.685	**0.199**	0.579	11.700	1.024	1.492	6.956	6.898	0.521	0.895	**0.334**
	50 × 20	0.430	7.132	0.744	1.012	5.921	4.492	0.387	0.607	**0.175**	0.421	7.985	0.815	1.219	6.027	5.423	0.379	0.731	**0.160**
	100 × 5	1.155	19.729	0.986	1.287	6.173	6.529	1.039	0.772	**0.146**	1.508	22.784	1.302	1.860	6.617	6.470	1.357	1.116	**0.022**
	100 × 10	0.940	12.741	0.878	1.450	6.285	6.130	0.846	0.870	**0.041**	0.991	14.624	0.920	1.475	6.566	8.730	0.892	0.885	**0.069**
	100 × 20	0.957	8.436	0.723	1.464	6.295	5.004	0.862	0.878	**0.043**	0.908	9.416	0.895	1.190	5.906	4.653	0.817	0.714	**0.035**
	200 × 10	1.642	16.049	1.503	1.597	5.660	6.237	1.477	0.958	**0.000**	1.574	18.245	1.501	1.558	5.518	5.558	1.416	0.935	**0.016**
	200 × 20	1.232	10.236	1.116	1.429	5.149	5.016	1.108	0.858	**0.000**	1.383	11.655	1.277	1.170	5.014	5.352	1.245	0.702	**0.014**
	500 × 20	2.640	13.934	2.023	1.940	4.837	3.703	2.376	1.164	**0.000**	3.202	16.026	2.577	2.456	5.350	4.991	2.882	1.474	**0.000**
	Average	0.864	11.287	0.906	1.242	5.414	4.062	0.778	0.745	**0.212**	0.958	12.682	1.090	1.403	5.528	4.411	0.862	0.842	**0.209**
6	20 × 5	0.024	7.743	0.309	0.650	3.247	1.627	**0.022**	0.942	0.293	0.038	8.609	0.628	0.765	3.344	3.286	**0.034**	0.469	0.504
	20 × 10	0.005	5.607	0.282	0.523	2.359	2.437	**0.005**	0.314	0.327	0.043	6.205	0.398	0.431	2.622	0.595	**0.038**	0.258	0.448
	20 × 20	0.046	3.311	0.378	0.407	1.622	3.043	**0.042**	0.244	0.261	0.028	3.765	0.370	0.500	1.734	2.261	**0.026**	0.300	0.316
	50 × 5	0.238	14.043	1.430	1.479	8.323	6.742	**0.214**	0.888	0.432	0.366	16.219	1.427	1.702	8.412	8.874	**0.330**	1.021	0.597
	50 × 10	0.144	9.219	1.151	1.270	6.595	7.281	**0.129**	0.762	0.405	0.267	10.229	1.135	1.485	6.950	7.231	**0.240**	0.891	0.475
	50 × 20	0.160	6.172	0.940	1.206	5.328	5.617	**0.144**	0.724	0.275	0.185	6.980	0.815	1.071	5.277	5.429	**0.166**	0.643	0.284
	100 × 5	0.750	17.606	1.192	1.267	6.252	8.078	0.675	0.760	**0.113**	1.163	20.064	1.367	1.455	7.116	7.361	1.047	0.873	**0.086**
	100 × 10	0.688	11.036	1.148	1.453	6.307	6.214	0.619	0.872	**0.090**	0.744	12.744	1.214	1.373	6.284	8.490	0.670	0.824	**0.037**
	100 × 20	0.666	7.243	1.035	1.269	5.875	5.248	0.599	0.761	**0.038**	0.664	8.275	0.996	1.311	5.663	5.687	0.598	0.787	**0.020**
	200 × 10	1.658	14.200	1.470	1.674	5.788	5.911	1.493	1.004	**0.000**	1.501	16.189	1.439	1.379	5.459	5.291	1.351	0.827	**0.000**
	200 × 20	1.158	8.844	0.978	1.115	5.038	4.018	1.042	0.669	**0.000**	1.321	10.153	1.141	1.289	4.860	5.308	1.189	0.773	**0.010**
	500 × 20	2.081	12.077	1.712	1.650	4.693	3.692	1.873	0.990	**0.000**	2.524	13.964	2.208	2.039	5.060	4.937	2.271	1.223	**0.000**
	Average	0.635	9.758	1.002	1.164	5.119	3.247	0.571	0.698	**0.186**	0.737	11.116	1.095	1.233	5.232	3.344	0.663	0.740	**0.231**
7	20 × 5	0.007	7.377	0.659	1.121	3.261	2.888	**0.007**	3.008	0.526	**0.000**	8.267	0.748	0.776	3.146	3.587	**0.000**	0.336	0.539
	20 × 10	**0.000**	4.964	0.363	0.404	2.320	3.364	**0.000**	0.243	0.354	**0.000**	5.790	0.461	0.463	2.149	2.057	**0.000**	0.278	0.362
	20 × 20	0.011	3.294	0.385	0.422	1.481	2.020	**0.010**	0.253	0.207	**0.000**	3.421	0.471	0.471	1.622	2.035	**0.000**	0.283	0.397
	50 × 5	0.073	14.332	1.792	1.744	8.185	7.697	**0.065**	1.046	0.922	0.115	15.725	1.897	1.824	8.569	9.067	**0.104**	1.094	0.587
	50 × 10	0.021	9.197	1.575	1.624	6.562	6.790	**0.019**	0.975	0.585	0.061	10.410	1.610	1.970	6.895	7.220	**0.055**	1.182	0.633
	50 × 20	0.039	5.907	1.057	1.123	4.909	5.045	**0.035**	0.674	0.528	0.051	6.671	1.114	1.233	5.093	5.500	**0.046**	0.740	0.481
	100 × 5	0.326	17.259	1.357	1.322	6.544	4.830	0.294	0.793	**0.257**	0.689	19.805	1.751	1.791	7.446	7.037	0.620	1.075	**0.139**
	100 × 10	0.357	10.905	1.347	1.324	6.241	5.987	0.321	0.794	**0.100**	0.357	12.146	1.442	1.413	6.615	7.720	0.321	0.848	**0.139**
	100 × 20	0.410	7.040	1.100	1.177	5.519	4.408	0.369	0.706	**0.054**	0.218	7.659	1.058	1.083	5.546	6.147	0.196	0.650	**0.182**
	200 × 10	1.247	13.020	1.376	1.195	5.467	5.012	1.123	0.717	**0.002**	1.471	15.164	1.451	1.301	5.551	6.535	1.324	0.781	**0.007**
	200 × 20	0.877	8.046	0.972	0.941	4.993	4.863	0.789	0.565	**0.003**	1.150	9.295	1.131	1.052	4.785	5.339	1.035	0.631	**0.017**
	500 × 20	1.967	11.235	1.834	1.479	4.730	4.370	1.771	0.888	**0.000**	2.210	12.935	1.953	1.681	4.810	4.511	1.989	1.009	**0.000**
	Average	0.445	9.381	1.152	1.156	5.018	3.261	0.400	0.694	**0.295**	0.527	10.607	1.257	1.255	5.186	3.146	0.474	0.753	**0.290**

Table 13
Results achieved by Friedman test at different scenarios.

Algorithms	Mean rank			
	SSD10	SSD50	SSD100	SSD125
IGA	3.06	3.14	3.30	3.45
HIG	8.53	8.78	9.00	9.00
MIG	4.45	4.55	4.74	4.84
CBLS	5.77	5.64	5.42	5.29
DDE	7.84	7.53	7.64	7.52
EDE	7.62	7.62	7.35	7.48
IG_NM	2.07	2.12	2.16	2.24
IG_RCM	3.71	3.65	3.43	3.08
MBIST_TSM	**1.94**	**1.97**	**1.97**	**2.10**
CN	720	720	720	720
p-value	9.860E−110	4.749E−108	9.798E−109	1.684E−107
Crit. Diff $\alpha = 0.05$	0.208	0.208	0.208	0.208
Crit. Diff $\alpha = 0.1$	0.187	0.187	0.187	0.187

Table 14
Results achieved by Friedman test at different jobs.

Algorithms	Mean rank				
	$n = 20$	$n = 50$	$n = 100$	$n = 200$	$n = 500$
IGA	1.86	2.28	3.58	5.06	5.54
HIG	8.97	8.63	8.68	8.98	9.00
MIG	4.42	4.82	4.76	4.31	4.85
CBLS	5.88	5.97	5.75	4.96	3.29
DDE	7.42	7.68	7.68	7.60	7.75
EDE	7.53	7.69	7.64	7.42	7.25
IG_NM	1.25	1.23	2.36	3.48	4.31
IG_RCM	4.00	4.14	3.53	2.19	2.00
MBIST-TSM	3.67	2.56	1.01	1.00	1.00
CN	720	720	720	480	240
p-value	4.22E−111	9.04E−113	1.36E−109	2.07E−71	1.87E−35
Crit. Diff $\alpha = 0.05$	0.208	0.208	0.208	0.255	0.361
Crit. Diff $\alpha = 0.1$	0.187	0.187	0.187	0.229	0.324

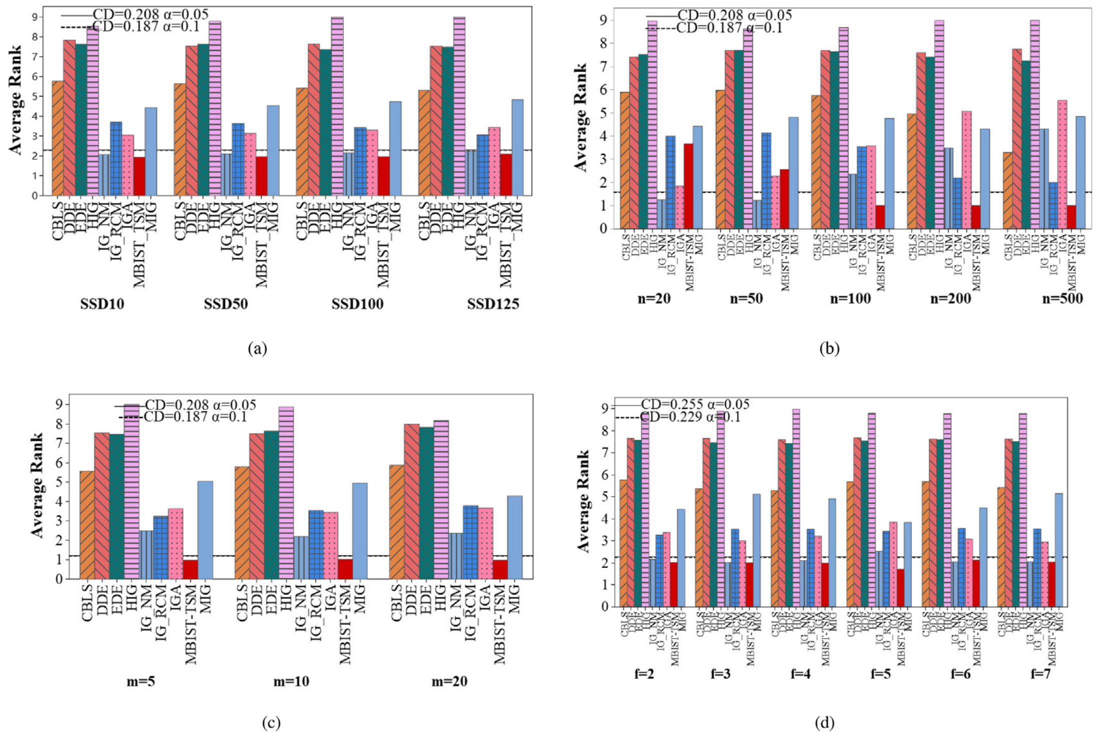


Fig. 15. The results of the Friedman-test. (a) at different scenarios. (b) at different jobs. (c) at different machines. (d) at different factories.

Table 15
Results achieved by Friedman test at different machines.

Algorithms	Mean rank		
	$m = 5$	$m = 10$	$m = 20$
IGA	3.63	3.46	3.67
HIG	9.00	8.88	8.17
MIG	5.04	4.96	4.29
CBLS	5.58	5.79	5.88
DDE	7.54	7.50	8.00
EDE	7.46	7.63	7.83
IG_NM	2.50	2.21	2.38
IG_RCM	3.25	3.54	3.79
MBIST-TSM	1.00	1.04	1.00
CN	720	960	1200
p-value	1.41E−34	4.75E−35	1.66E−33
Crit. Diff $\alpha = 0.05$	0.208	0.180	0.161
Crit. Diff $\alpha = 0.1$	0.187	0.162	0.145

flow shop scheduling problems and the multi-objective scheduling problems).

CRediT authorship contribution statement

Fuqing Zhao: Funding acquisition, Investigation, Supervision, Resources, Formal analysis. **Haizhu Bao:** Software, Writing – original draft, Experiments of the algorithms, Methodology. **Ling Wang:** Project administration, Review. **Tianpeng Xu:** Conceptualization. **Ningning Zhu:** Visualization. **Jonrinaldi:** Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Authors shared the code in the following website: https://github.com/banian2314/MBIST-TSM. In addition, supplemented the corresponding content in Section 6.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China under grant 62063021. It was also supported by the Key talent project of Gansu Province, China (ZZ2021G50700016), the Key Research Programs of Science and Technology Commission Foundation of Gansu Province, China (21YF5WA086), Lanzhou Science Bureau project (2018-rc-98), and Project of Gansu Natural Science Foundation, China (21JR7RA204), respectively.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix

A.1. The explicit calculation of makespan

A.1.1. The first calculation method

For the first calculation method, according to Eqs. (14)–(17), the departure time of each job in each factory on each machine is calculated

Table 16
Results achieved by Friedman test at different factories.

Algorithms	Mean rank					
	$f = 2$	$f = 3$	$f = 4$	$f = 5$	$f = 6$	$f = 7$
IGA	3.38	2.99	3.21	3.85	3.07	2.95
HIG	8.79	8.88	8.98	8.79	8.77	8.77
MIG	4.43	5.11	4.90	3.83	4.50	5.14
CBLS	5.77	5.37	5.29	5.67	5.69	5.41
DDE	7.65	7.65	7.60	7.67	7.63	7.62
EDE	7.56	7.46	7.42	7.54	7.60	7.50
IG_NM	2.15	2.02	2.10	2.52	2.05	2.04
IG_RCM	3.25	3.52	3.52	3.42	3.56	3.54
MBIST-TSM	2.02	2.00	1.98	1.71	2.13	2.03
CN	480	480	480	480	480	480
p-value	2.26E−72	8.17E−74	2.46E−72	6.98E−71	3.75E−72	2.23E−72
Crit. Diff $\alpha = 0.05$	0.255	0.255	0.255	0.255	0.255	0.255
Crit. Diff $\alpha = 0.1$	0.229	0.229	0.229	0.229	0.229	0.229

Table 17
Results achieved by Wilcoxon paired signed test.

Scenarios		MBIST-TSM *vs.*							
		IGA	HIG	MIG	CBLS	DDE	EDE	IG_NM	IG_RCM
SSD10	$R+$	2349.00	3003.00	2967.50	3003.00	3003.00	3003.00	2297.00	2980.00
	$R-$	654.00	0.00	35.50	0.00	0.00	0.00	706.00	23.00
	Bonding value	0	0	0	0	0	0	0	0
	p-value	1.68E−05	2.46E−14	9.80E−14	2.46E−14	2.46E−14	2.46E−14	5.36E−05	6.05E−14
SSD50	$R+$	2234.00	3003.00	2914.00	3003.00	3003.00	3002.00	2171.50	2968.50
	$R-$	769.00	0.00	12.00	0.00	0.00	1.00	831.50	34.50
	Bonding value	0	0	1	0	0	0	0	0
	p-value	2.00E−04	2.46E−14	5.81E−14	2.46E−14	2.46E−14	2.56E−14	6.69E−04	9.43E−14
SSD100	$R+$	2157.00	3003.00	2960.00	3003.00	3003.00	3003.00	2098.50	2932.00
	$R-$	846.00	0.00	43.00	0.00	0.00	0.00	904.50	71.00
	Bonding value	0	0	0	0	0	0	0	0
	p-value	8.74E−04	2.46E−14	1.31E−13	2.46E−14	2.46E−14	2.46E−14	2.44E−03	3.78E−13
SSD125	$R+$	2211.50	3003.00	2991.00	3002.00	3003.00	3003.00	2107.50	2851.50
	$R-$	791.50	0.00	12.00	1.00	0.00	0.00	818.50	151.50
	Bonding value	0	0	0	0	0	0	1	0
	p-value	3.12E−04	2.46E−14	3.94E−14	2.56E−14	2.46E−14	2.46E−14	8.47E−04	7.15E−12

as follows:

$$D_{1,\pi_1(1),0} = D_{1,\pi_1(0),1} + s_{\pi_1(0),\pi_1(1),1} = 97$$
$$D_{1,\pi_1(1),1} = \max\left\{D_{1,\pi_1(0),2} + s_{\pi_1(0),\pi_1(1),2}, D_{1,\pi_1(1),0} + p_{\pi_1(1),1}\right\} = 191$$
$$D_{1,\pi_1(1),2} = D_{1,\pi_1(1),1} + p_{\pi_1(1),2} = 246$$
$$D_{1,\pi_1(2),0} = D_{1,\pi_1(1),1} + s_{\pi_1(1),\pi_1(2),1} = 198$$
$$D_{1,\pi_1(2),1} = \max\left\{D_{1,\pi_1(1),2} + s_{\pi_1(1),\pi_1(2),2}, D_{1,\pi_1(2),0} + p_{\pi_1(2),1}\right\} = 252$$
$$D_{1,\pi_1(2),2} = D_{1,\pi_1(2),1} + p_{\pi_1(2),2} = 306$$
$$D_{1,\pi_1(3),0} = D_{1,\pi_1(2),1} + s_{\pi_1(2),\pi_1(3),1} = 343$$
$$D_{1,\pi_1(3),1} = \max\left\{D_{1,\pi_1(2),2} + s_{\pi_1(2),\pi_1(3),2}, D_{1,\pi_1(3),0} + p_{\pi_1(3),1}\right\} = 378$$
$$D_{1,\pi_1(3),2} = D_{1,\pi_1(3),1} + p_{\pi_1(3),2} = 390$$
$$D_{2,\pi_2(1),0} = D_{2,\pi_2(0),1} + s_{\pi_2(0),\pi_2(1),1} = 30$$
$$D_{2,\pi_2(1),1} = \max\left\{D_{2,\pi_2(0),2} + s_{\pi_2(0),\pi_2(1),2}, D_{2,\pi_2(1),0} + p_{\pi_2(1),1}\right\} = 128$$
$$D_{2,\pi_2(1),2} = D_{2,\pi_2(1),1} + p_{\pi_2(1),2} = 196$$
$$D_{2,\pi_2(2),0} = D_{2,\pi_2(1),1} + s_{\pi_2(1),\pi_2(2),1} = 170$$
$$D_{2,\pi_2(2),1} = \max\left\{D_{2,\pi_2(1),2} + s_{\pi_2(1),\pi_2(2),2}, D_{2,\pi_2(2),0} + p_{\pi_2(2),1}\right\} = 285$$
$$D_{2,\pi_2(2),2} = D_{2,\pi_2(2),1} + p_{\pi_2(2),2} = 368$$

The corresponding makespan is also $C_{max}\left(\prod\right) = \max\left\{D_{1,\pi_1(3),2}, D_{2,\pi_2(2),2}\right\} = 390$.

A.1.2. The second calculation method

For the second calculation method, according to Eqs. (19)–(23), the reverse departure time of each job in each factory on each machine is calculated as follows.

$$Q_{1,\pi_1(3),2} = Q_{1,\pi_1(3),3} + p_{\pi_1(3),2} = 12$$
$$Q_{1,\pi_1(3),1} = Q_{1,\pi_1(3),2} + p_{\pi_1(3),1} = 47$$
$$Q_{1,\pi_1(2),3} = Q_{1,\pi_1(3),2} + s_{\pi_1(2),\pi_1(3),2} = 64$$
$$Q_{1,\pi_1(2),2} = \max\left\{Q_{1,\pi_1(3),1} + s_{\pi_1(2),\pi_1(3),1}, Q_{1,\pi_1(2),3} + p_{\pi_1(2),2}\right\} = 138$$
$$Q_{1,\pi_1(2),1} = Q_{1,\pi_1(2),2} + p_{\pi_1(2),1} = 180$$
$$Q_{1,\pi_1(1),3} = Q_{1,\pi_1(2),2} + s_{\pi_1(1),\pi_1(2),2} = 144$$
$$Q_{1,\pi_1(1),2} = \max\left\{Q_{1,\pi_1(2),1} + s_{\pi_1(1),\pi_1(2),1}, Q_{1,\pi_1(1),3} + p_{\pi_1(1),2}\right\} = 199$$
$$Q_{1,\pi_1(1),1} = Q_{1,\pi_1(1),2} + p_{\pi_1(1),1} = 293$$
$$Q_{1,\pi_1(0),3} = Q_{1,\pi_1(1),2} + s_{\pi_1(0),\pi_1(1),2} = 207$$
$$Q_{1,\pi_1(0),2} = \max\left\{Q_{1,\pi_1(1),1} + s_{\pi_1(0),\pi_1(1),1}, Q_{1,\pi_1(0),3} + p_{\pi_1(0),2}\right\} = 390$$
$$Q_{1,\pi_1(0),1} = Q_{1,\pi_1(0),2} + p_{\pi_1(0),1} = 390$$
$$Q_{2,\pi_2(2),2} = Q_{2,\pi_2(2),3} + p_{\pi_2(2),2} = 83$$
$$Q_{2,\pi_2(2),1} = Q_{2,\pi_2(2),2} + p_{\pi_2(2),1} = 146$$
$$Q_{2,\pi_2(1),3} = Q_{2,\pi_2(2),2} + s_{\pi_2(1),\pi_2(2),2} = 172$$
$$Q_{2,\pi_2(1),2} = \max\left\{Q_{2,\pi_2(2),1} + s_{\pi_2(1),\pi_2(2),1}, Q_{2,\pi_2(1),3} + p_{\pi_2(1),2}\right\} = 240$$
$$Q_{2,\pi_2(1),1} = Q_{2,\pi_2(1),2} + p_{\pi_2(1),1} = 338$$
$$Q_{2,\pi_2(0),3} = Q_{2,\pi_2(1),2} + s_{\pi_2(0),\pi_2(1),2} = 215$$
$$Q_{2,\pi_2(0),2} = \max\left\{Q_{2,\pi_2(1),1} + s_{\pi_2(0),\pi_2(1),1}, Q_{2,\pi_2(0),3} + p_{\pi_2(0),2}\right\} = 368$$
$$Q_{2,\pi_2(0),1} = Q_{2,\pi_2(0),2} + p_{\pi_2(0),1} = 368$$

The corresponding makespan is also $C_{max}\left(\prod\right) = \max\left\{Q_{1,\pi_1(0),1}, Q_{2,\pi_2(0),1}\right\} = 390$.

A.2. The explicit process of the measurement d_ρ

$$d_\rho = \sum_{j=2}^{m} \max 0, (p_{\pi(\mu-1),j} - p_{\rho,j-1}) \xrightarrow{\rho=1,\mathrm{m}=2,\mu=2} d_1 = \sum_{j=2}^{2} \times \max\left\{0, \left(p_{\pi(1),j} - p_{1,j-1}\right)\right\} = max\left\{0, \left(p_{\pi(1),2} - p_{1,1}\right)\right\} = max\left\{0, \left(p_{\pi(1),2} - p_{\pi(1),1}\right)\right\} = max\left\{0, (54-42)\right\} = 12$$

Note: In the second step, U is $\{\pi(1)\}$ after the **Step 2**. Therefore, in the **Step 3**, ρ can only be 1. As a result, the following formula holds: $p_{\rho,j} \xrightarrow{\rho=1,U=\{\pi(1)\}} p_{\rho,j} = p_{1,j} = p_{\pi(1),j}$.

A.3. Details of the compared heuristics

Distributed SPT (DSPT) method: It contains three steps. First, arrange n jobs in the ascending order of the time $T_i = \sum_{j=1}^{m} p_{\pi_{i,j}}$, $i = 1, 2, \ldots, n$, to produce a job sequence $J = \{J_1, J_2, \ldots, J_n\}$. Then, construct the partial job schedules at each factory. To be specific, from J_1, J_2 and so on until J_n, orderly assign the job to the factory that has the earliest completion time after including this job. The processing of jobs at the same factory is according to their assigning order. Finally, combine the partial schedule of all factories to obtain a complete scheduling solution.

Distributed LPT (DLPT) method: The procedure of DLPT is the same as those in DSPT, except that the job sequence $J = \{J_1, J_2, \ldots, J_n\}$ is obtained in the descending order of T_i, $i = 1, 2, \ldots, n$.

Distributed large–small (DLS) method: Using the job sequence $J = \{J_1, J_2, \ldots, J_n\}$ from the DLPT, construct a new sequence $\xi = \left\{J_1, J_n, J_2, J_{\mathrm{n}-1}, \ldots\right\}$ by removing the first and the last jobs of J to ξ, and then the first and the last ones from the remaining sequence and so on. Then, apply the steps two and three in the DSPT for ξ to obtain a complete scheduling solution.

Distributed NEH (DNEH) method: It also consists of three steps. First, obtain the job sequence $J = \{J_1, J_2, \ldots, J_n\}$ as done in the DLPT. Then, starting from J_1, orderly assign each job to the position leading to the least makespan by inserting the job into all possible slots of all factories, which yields the partial job schedule of all factories. Finally, combine them to obtain a complete scheduling solution.

References

Bargaoui, H., Belkahla Driss, O., Ghédira, K., 2017. A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion. Comput. Ind. Eng. 111, 239–250. http://dx.doi.org/10.1016/J.CIE.2017.07.020.

Bastos, L.S.L., Marchesi, J.F., Hamacher, S., Fleck, J.L., 2019. A mixed integer programming approach to the patient admission scheduling problem. European J. Oper. Res. 273, 831–840. http://dx.doi.org/10.1016/J.EJOR.2018.09.003.

Chen, Q., Pan, Q., Zhang, B., Ding, J., Li, J., 2019. Effective hot rolling batch scheduling algorithms in compact strip production. IEEE Trans. Autom. Sci. Eng. 16, 1933–1951.

Cheng, C.Y., Pourhejazy, P., Ying, K.C., Liao, Y.H., 2021. New benchmark algorithms for no-wait flowshop group scheduling problem with sequence-dependent setup times. Appl. Soft Comput. 111, 107705. http://dx.doi.org/10.1016/j.asoc.2021.107705.

Deng, G., Su, Q., Zhang, Z., Liu, H., Zhang, S., Jiang, T., 2020. A population-based iterated greedy algorithm for no-wait job shop scheduling with total flow time criterion. Eng. Appl. Artif. Intell. 88, 103369. http://dx.doi.org/10.1016/J.ENGAPPAI.2019.103369.

Deng, J., Wang, L., 2017. A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. Swarm Evol. Comput. 32, 121–131. http://dx.doi.org/10.1016/j.swevo.2016.06.002.

Fernandez-Viagas, V., Framinan, J.M., 2015. A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. Int. J. Prod. Res. 53, 1111–1123. http://dx.doi.org/10.1080/00207543.2014.948578.

Framinan, J.M., Perez-Gonzalez, P., Fernandez-Viagas, V., 2019. Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. European J. Oper. Res. 273, 401–417. http://dx.doi.org/10.1016/j.ejor.2018.04.033.

Fu, Y., Wang, H., Wang, J., Pu, X., 2020. Multiobjective modeling and optimization for scheduling a stochastic hybrid flow shop with maximizing processing quality and minimizing total tardiness. IEEE Syst. J. 15, 4696–4707. http://dx.doi.org/10.1109/jsyst.2020.3014093.

Gao, J., Chen, R., Deng, W., 2013. An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. Int. J. Prod. Res. 51, 641–651. http://dx.doi.org/10.1080/00207543.2011.644819.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Ann. Discrete Math. 5, 287–326. http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

Han, Y., Gong, D., Jin, Y., Pan, Q., 2019. Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. IEEE Trans. Cybern. 49, 184–197. http://dx.doi.org/10.1109/TCYB.2017.2771213.

Han, Y., Li, J., Sang, H., Liu, Y., Gao, K., Pan, Q., 2020. Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time. Appl. Soft Comput. J. 93, http://dx.doi.org/10.1016/j.asoc.2020.106343.

Hatami, S., Ruiz, R., Andrés-Romano, C., 2013. The distributed assembly permutation flowshop scheduling problem. Int. J. Prod. Res. 51, 5292–5308. http://dx.doi.org/10.1080/00207543.2013.807955.

He, X., Pan, Q., Gao, L., Wang, L., Suganthan, P.N., 2021. A greedy cooperative co-evolution ary algorithm with problem-specific knowledge for multi-objective flowshop group scheduling problems. IEEE Trans. Evol. Comput. 639798 (1), http://dx.doi.org/10.1109/tevc.2021.3115795.

Huang, J.P., Pan, Q.K., Gao, L., 2020. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm Evol. Comput. 59, http://dx.doi.org/10.1016/j.swevo.2020.100742.

Huang, J.P., Pan, Q.K., Miao, Z.H., Gao, L., 2021. Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times. Eng. Appl. Artif. Intell. 97, 104016. http://dx.doi.org/10.1016/j.engappai.2020.104016.

Jing, X.L., Pan, Q.K., Gao, L., 2021. Local search-based metaheuristics for the robust distributed permutation flowshop problem. Appl. Soft Comput. 105, 107247. http://dx.doi.org/10.1016/j.asoc.2021.107247.

Li, J.Q., Bai, S.C., Duan, P.Y., Sang, H.Y., Han, Y.Y., Zheng, Z.X., 2019. An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. Int. J. Prod. Res. 57, 6922–6942. http://dx.doi.org/10.1080/00207543.2019.1571687.

Li, J.Q., Song, M.X., Wang, L., Duan, P.Y., Han, Y.Y., Sang, H.Y., Pan, Q.K., 2020. Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. IEEE Trans. Cybern. 50, 2425–2439.

Lin, S.W., Ying, K.C., 2013. Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega (U. K.) 41, 383–389. http://dx.doi.org/10.1016/J.OMEGA.2012.03.006.

Lin, S.W., Ying, K.C., 2016. Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. Comput. Ind. Eng. 99, 202–209. http://dx.doi.org/10.1016/j.cie.2016.07.027.

Lin, S.W., Ying, K.C., Huang, C.Y., 2013. Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. Int. J. Prod. Res. 51, 5029–5038. http://dx.doi.org/10.1080/00207543.2013.790571.

Liu, J., Bo, R., Wang, S., Chen, H., 2021. Optimal scheduling for profit maximization of energy storage merchants considering market impact based on dynamic programming. Comput. Ind. Eng. 155, http://dx.doi.org/10.1016/J.CIE.2021.107212.

Miyata, H.H., Nagano, M.S., 2022. An iterated greedy algorithm for distributed blocking flow shop with setup times and maintenance operations to minimize makespan. Comput. Ind. Eng. 171, 108366. http://dx.doi.org/10.1016/j.cie.2022.108366.

Montgomery, D.C., 2009. Design and Analysis of Experiments, seventh ed. John Wiley & Sons.

Naderi, B., Ruiz, R., 2010. The distributed permutation flowshop scheduling problem. Comput. Oper. Res. 37, 754–768. http://dx.doi.org/10.1016/j.cor.2009.06.019.

Naderi, B., Ruiz, R., 2014. A scatter search algorithm for the distributed permutation flowshop scheduling problem. European J. Oper. Res. 239, 323–334. http://dx.doi.org/10.1016/j.ejor.2014.05.024.

Naik, A., Satapathy, S.C., Abraham, A., 2020. Modified Social Group Optimization—a meta-heuristic algorithm to solve short-term hydrothermal scheduling. Appl. Soft Comput. J. 95, 106524. http://dx.doi.org/10.1016/j.asoc.2020.106524.

Newton, M.A.H., Riahi, V., Su, K., Sattar, A., 2019. Scheduling blocking flowshops with setup times via constraint guided and accelerated local search. Comput. Oper. Res. 109, 64–76. http://dx.doi.org/10.1016/j.cor.2019.04.024.

Osman, I., Potts, C., 1989. Simulated annealing for permutation flow-shop scheduling. Omega 17, 551–557. http://dx.doi.org/10.1016/0305-0483(89)90059-5.

Pan, Q.K., Gao, L., Wang, L., 2020. An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. IEEE Trans. Cybern. http://dx.doi.org/10.1109/TCYB.2020.3041494.

Prata, B. de A., Rodrigues, C.D., Framinan, J.M., 2021. Customer order scheduling problem to minimize makespan with sequence-dependent setup times. Comput. Ind. Eng. 151, 106962. http://dx.doi.org/10.1016/j.cie.2020.106962.

Prata, B. de A., Rodrigues, C.D., Framinan, J.M., 2022. A differential evolution algorithm for the customer order scheduling problem with sequence-dependent setup times. Expert Syst. Appl. 189, 116097. http://dx.doi.org/10.1016/j.eswa.2021.116097.

Riahi, V., Newton, M.A.H., Sattar, A., 2021. Constraint based local search for flowshops with sequence-dependent setup times. Eng. Appl. Artif. Intell. 102, 104264. http://dx.doi.org/10.1016/J.ENGAPPAI.2021.104264.

Ribas, I., Companys, R., 2021. A computational evaluation of constructive heuristics for the parallel blocking flow shop problem with sequence-dependent setup times. Int. J. Ind. Eng. Comput. 12, 321–328. http://dx.doi.org/10.5267/j.ijiec.2021.1.004.

Ribas, I., Companys, R., Tort-Martorell, X., 2017. Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst. Appl. 74, 41–54. http://dx.doi.org/10.1016/j.eswa.2017.01.006.

Ribas, I., Companys, R., Tort-Martorell, X., 2019. An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem. Expert Syst. Appl. 121, 347–361. http://dx.doi.org/10.1016/j.eswa.2018.12.039.

Ribas, I., Companys, R., Tort-Martorell, X., 2021. An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. Expert Syst. Appl. 184, 115535. http://dx.doi.org/10.1016/j.eswa.2021.115535.

Rifai, A.P., Nguyen, H.T., Dawal, S.Z.M., 2016. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl. Soft Comput. J. 40, 42–57. http://dx.doi.org/10.1016/j.asoc.2015.11.034.

Rossi, F.L., Nagano, M.S., 2021. Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. Comput. Ind. Eng. 157, 107337. http://dx.doi.org/10.1016/j.cie.2021.107337.

Ruiz, R., Maroto, C., Alcaraz, J., 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. European J. Oper. Res. 165, 34–54.

Ruiz, R., Pan, Q.K., Naderi, B., 2019. Iterated Greedy methods for the distributed permutation flowshop scheduling problem. Omega (U. K.) 83, 213–222. http://dx.doi.org/10.1016/j.omega.2018.03.004.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European J. Oper. Res. 177, 2033–2049. http://dx.doi.org/10.1016/J.EJOR.2005.12.009.

Shao, Z.S., Pi, D.C., Shao, W.S., 2018. A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. Swarm Evol. Comput. 40, 53–75. http://dx.doi.org/10.1016/j.swevo.2017.12.005.

Shao, Z.S., Pi, D.C., Shao, W.S., 2020a. Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. Expert Syst. Appl. 145, 113147. http://dx.doi.org/10.1016/j.eswa.2019.113147.

Shao, Z.S., Pi, D.C., Shao, W.S., Yuan, P.S., 2019. An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. Eng. Appl. Artif. Intell. 78, 124–141. http://dx.doi.org/10.1016/j.engappai.2018.11.005.

Shao, Z.S., Shao, W.S., Pi, D.C., 2020b. Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. Swarm Evol. Comput. 59, http://dx.doi.org/10.1016/j.swevo.2020.100747.

Taillard, E., 1990. Some efficient heuristic methods for the flow shop sequencing problem. European J. Oper. Res. 47, 65–74.

Takano, M.I., Nagano, M.S., 2019. Evaluating the performance of constructive heuristics for the blocking flow shop scheduling problem with setup times. Int. J. Ind. Eng. Comput. 10, 37–50. http://dx.doi.org/10.5267/j.ijiec.2018.5.002.

Tasgetiren, M.F., Kizilay, D., Pan, Q.K., Suganthan, P.N., 2017. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput. Oper. Res. 77, 111–126. http://dx.doi.org/10.1016/j.cor.2016.07.002.

Tomazella, C.P., Nagano, M.S., 2020. A comprehensive review of Branch-and-Bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem. Expert Syst. Appl. 158, http://dx.doi.org/10.1016/J.ESWA.2020.113556.

Wang, K., Huang, Y., Qin, H., 2016. A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown. J. Oper. Res. Soc. 67, 68–82. http://dx.doi.org/10.1057/jors.2015.50.

Wang, Y., Li, X., Ruiz, R., Sui, S., 2017. An iterated greedy heuristic for mixed no-wait flowshop problems. IEEE Trans. Cybern. 1–14. http://dx.doi.org/10.1109/TCYB.2017.2707067.

Wang, J.J., Wang, L., 2020. A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. IEEE Trans. Syst. Man Cybern. Syst. 50, 1805–1819. http://dx.doi.org/10.1109/TSMC.2017.2788879.

Wang, S.Y., Wang, L., Liu, M., Xu, Y., 2013. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. Int. J. Prod. Econ. 145, 387–396. http://dx.doi.org/10.1016/J.IJPE.2013.05.004.

Wang, Z.K., Zhen, H.L., Deng, J.D., Zhang, Q.F., Li, X.J., Yuan, M.X., Zeng, J., 2022. Multiobjective optimization-aided decision-making system for large-scale manufacturing planning. IEEE Trans. Cybern. 52, 8326–8339. http://dx.doi.org/10.1109/TCYB.2021.3049712.

Xin, X., Jiang, Q.Q., Li, S.H., Gong, S.Y., Chen, K., 2021. Energy-efficient scheduling for a permutation flow shop with variable transportation time using an improved discrete whale swarm optimization. J. Clean. Prod. 293, http://dx.doi.org/10.1016/j.jclepro.2021.126121.

Ying, K.C., Lin, S.W., 2017. Minimizing makespan in distributed blocking flowshops using hybrid iterated greedy algorithms. IEEE Access 5, 15694–15705. http://dx.doi.org/10.1109/ACCESS.2017.2732738.

Ying, K.C., Lin, S.W., Cheng, C.Y., He, C.D., 2017. Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. Comput. Ind. Eng. 110, 413–423. http://dx.doi.org/10.1016/j.cie.2017.06.025.

Zhang, S.X., Chan, W.S., Tang, K.S., Zheng, S.Y., 2021. Adaptive strategy in differential evolution via explicit exploitation and exploration controls. Appl. Soft Comput. 107, 107494. http://dx.doi.org/10.1016/j.asoc.2021.107494.

Zhang, G.H., Xing, K.Y., Cao, F., 2018. Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Eng. Appl. Artif. Intell. 76, 96–107. http://dx.doi.org/10.1016/J.ENGAPPAI.2018.09.005.

Zhao, F.Q., He, X., Wang, L., 2020a. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. IEEE Trans. Cybern. 1–13. http://dx.doi.org/10.1109/tcyb.2020.3025662.

Zhao, F.Q., Shao, D.Q., Wang, L., Xu, T.P., Zhu, N.N., Jonrinaldi, 2022a. An effective water wave optimization algorithm with problem-specific knowledge for the distributed assembly blocking flow-shop scheduling problem. Knowl.-Based Syst. 243, 108471. http://dx.doi.org/10.1016/j.knosys.2022.108471.

Zhao, F.Q., Xu, Z.S., Wang, L., Zhu, N.N., Xu, T.P., Jonrinaldi, 2022b. A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem. IEEE Trans. Ind. Inf. http://dx.doi.org/10.1109/TII.2022.3192881.

Zhao, F.Q., Zhang, L.X., Cao, J., Tang, J.X., 2021. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. Comput. Ind. Eng. 153, 107082. http://dx.doi.org/10.1016/J.CIE.2020.107082.

Zhao, F.Q., Zhang, L.X., Zhang, Y., Ma, W.M., Zhang, C., Song, H.B., 2020b. A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. Expert Syst. Appl. 146, http://dx.doi.org/10.1016/j.eswa.2019.113166.

Zhao, F.Q., Zhao, L.X., Wang, L., Song, H.B., 2020c. An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. Expert Syst. Appl. 160, http://dx.doi.org/10.1016/j.eswa.2020.113678.

Zhen, H.L., Wang, Z.K., Li, X.J., Zhang, Q.F., Yuan, M.X., Zeng, J., 2021. Accelerate the optimization of large-scale manufacturing planning using game theory. Complex Intell. Syst. 8, 2719–2730. http://dx.doi.org/10.1007/s40747-021-00352-7.