Journal of Manufacturing Systems 62 (2022) 650–667

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

An improved genetic algorithm for flexible job shop scheduling problem considering reconfigurable machine tools with limited auxiliary modules

Jiaxin Fan, Chunjiang Zhang, Qihao Liu, Weiming Shen *, Liang Gao

State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

Keywords:
Reconfigurable manufacturing systems
Flexible job shop scheduling
Reconfigurable machine tools
Auxiliary modules
Mixed integer linear programming
Improved genetic algorithm

ABSTRACT

Reconfigurable manufacturing system is widely regarded as a major drive towards the next-generation manufacturing, where one of the most common scenarios is that advanced reconfigurable machine tools (RMT) are equipped with auxiliary modules (AM) to improve the production flexibility, thus rapidly responding to market demands. Production scheduling is facing great challenges in this situation, because the limited number of AMs becomes the primary difficulty for the resource allocation. This paper investigates a flexible job shop scheduling problem with machine reconfigurations (FJSP-MR) to minimize the total weighted tardiness (TWT). Firstly, a sequence-based mixed integer linear programming (MILP) model is established for the FJSP-MR. Afterwards, considering the characteristics of the AM selection sub-problem, an improved genetic algorithm (IGA) with problem-specific encoding and decoding strategies is developed, in which an iterated local search with a modified *k*-insertion neighborhood structure is introduced for further optimization. Critical paths for the less common TWT objective are fully considered to ensure the local search is performed on bottlenecks of incumbent solutions. The proposed MILP model and IGA are tested on three groups of instances extended from public benchmark sets. Numerical experimental results indicate that the proposed IGA is highly efficient for the FJSP-MR in a variety of scenarios, where the specially designed local search is an effective complementary component for the basic genetic algorithm. Furthermore, a real-world FJSP-MR case is studied to show the proposed IGA is applicable to large-scale engineering problems.

1. Introduction

To accommodate diversified needs of consumers, the manufacturing mode is gradually shifting to the multi-variety and small batch production. Reconfigurable manufacturing system (RMS) becomes a hot research topic for its ability to provide various machining processes, where machine configurations can be converted by assemblying different auxiliary modules (AMs) [22,33]. Machine reconfigurations significantly improve the flexibility and productivity of manufacturing systems, meanwhile, making the production scheduling complicated due to an extra decision-making process for organizing modules. Since this scenario generally exists in flexible job shops [11], which are already with strong flexibility in the machine assignment, obtaining satisfactory scheduling plans becomes even more challenging. Module selection plays a critical role under this circumstance. If a set of consecutive operations on a machine share the same AM, the reconfiguration processes can be left out, thus bringing a significant reduction in the waiting time and contributing to on-time delivery. Therefore, considering the module allocation in the production scheduling is of great importance to modern shop floors.

Flexible job shop scheduling problem (FJSP) deals with the machine assignment and operation sequencing simultaneously [16]. If reconfigurable machine tools (RMT) are introduced [23], the complexity of the scheduling problem will exponentially grow for assigning available AMs to RMTs. The AM selection obviously interacts with the two aforementioned sub-problems, which makes them unlikely to be optimized individually. Most of the previous literature considers a machine reconfiguration as a production preparation, and can be included in the machining process or carried out anytime before the operation as a non-anticipatory setup. The assumptions apply to plug-and-play RMTs or machines with sufficient tools, and simplify the AM availability contraint for computational efficiency. When AMs are in short supply, it is important to allocate the AMs reasonably to avoid unnecessary reconfiguration processes. As far as we know, the existing research work

* Corresponding author.
E-mail addresses: fanjx@hust.edu.cn (J. Fan), zhangcj@hust.edu.cn (C. Zhang), lllqh@hust.edu.cn (Q. Liu), shenwm@hust.edu.cn (W. Shen), gaoliang@mail.hust.edu.cn (L. Gao).

https://doi.org/10.1016/j.jmsy.2022.01.014
Received 10 November 2021; Received in revised form 2 January 2022; Accepted 27 January 2022
Available online 9 February 2022

either simplifies the influence caused by machine reconfigurations to some extent, or takes the AM allocation as an independent optimization problem out of the production scheduling. However, integrating the AM selection into FJSP and optimizing them as a whole are more feasible and practically significant.

The motivations of this paper are to fill the gap for scheduling a limited number of modules in a flexible job shop and to develop an effective scheduling method for real-world engineering problems in this production scenario. Hence, an FJSP with machine reconfigurations (FJSP-MR) for minimizing the total weighted tardiness (TWT) is formulated and thoroughly investigated. The main contributions of this paper are as follows:

(1) A sequence-based mixed integer linear programming (MILP) model is established for the FJSP-MR, and has been validated on small-scale instances.
(2) For large-scale problems, an improved genetic algorithm (IGA) is proposed with problem-focused encoding and decoding strategies, and genetic operators. To overcome the many-to-one situations resulting from the encoding form and to improve the local search ability, a disjunctive graph model for the FJSP-MR is formulated. According to newly defined arcs and calculations of path lengths, an effective iterated local search (ILS) is developed with a modified *k*-insertion neighborhood structure which performs on bottlenecks for the TWT objective.
(3) A real-world FJSP-MR case is tested to further investigate the performance of IGA. By identifying numbers of available AMs for each RMT, the ILS is simplified to only re-assign the operations with scarce AMs. The proposed IGA has been proved to be capable of addressing practical FJSP-MRs after some problem-specific adjustments.

The remainder of this paper is organized as follows: Section 2 provides a related literature review. Section 3 presents the problem description and MILP model for the FJSP-MR. Section 4 introduces the proposed IGA and ILS component in detail. Section 5 shows the numerical experimental results and analyses. Section 6 gives a real-world case study for validation. Section 7 summarizes this work and discusses future research directions.

2. Related literature review

2.1. Production scheduling in RMS

Reconfigurable machine is one of the basic components in an RMS, and can be classified into four types [47], RMT, reconfigurable assembly machine, reconfigurable inspection machine, and reconfigurable fixture, where RMT is the main focus of this paper. An RMT is usually designed with a group of AMs which can be assembled to process different operations or increase the processing speed [15]. Assembly and disassembly processes for RMTs are complicated and time-consuming, which usually delay the production progress. Therefore, the machine reconfiguration is an important issue for the production scheduling in RMSs [6]. Bensmaine et al. [4] proposed a novel heuristic to solve an integrated process planning and scheduling problem in RMSs. A mathematical model for the production scheduling in an RMS was established and optimally solved by a commercial solver in [3]. Dou et al. [12] formulated a bi-objective problem for minimizing the total cost and total tardiness in a reconfigurable flow line, and then developed an effective non-dominated sorting genetic algorithm II (NSGA-II) to solve the real-world problem. More existing research work about scheduling problems in RMSs were summarized by [47] and [6], where the machine selection flexibility, as another primary characteristic of RMSs, is generally taken into consideration. Hence, the majority of scheduling problems concerning RMTs are extended from the classical FJSP.

The most related research work about FJSP with RMTs was presented in [31], which addressed a reconfiguration process as a non-anticipatory setup, and formulated the problem as an FJSP with configuration dependent setup time (FJSP-CDST). Two MILP models, based on the operation-position and operation-sequence respectively, were proposed to find optimal solutions. Besides, a self adaptive differential evolution algorithm (SADE) with Nelder-Mead mutation strategy was developed for large-scale FJSP-CDST instances. The sequence-based MILP model demonstrated its superiority over the position-based model, while the proposed mutation strategy had been proved to be effective by extensive experimental results. This work considers a plug-and-play scenario for machine reconfigurations, which means the necessary AMs for each RMT are sufficient and reconfiguration processes can be performed anytime, thus the CDSTs are introduced and attached to operations as non-anticipatory setups. However, in most of real-world shop floors, AMs are usually scarce manufacturing resources and cannot be regarded as unlimited. Allocating AMs to the proper operations to omit reconfiguration processes as many as possible has practical values and is worth research.

Recently, more research efforts had been paid to the AM allocation in RMSs. Bortolini et al. [5] established an optimization model to dynamically manage a limited number of AMs in cellular RMSs. The reported method tried to minimize the reconfiguration time, part flow time, and AM flow time for manufacturing cells, while balancing the number of available AMs for a given series of time periods. In terms of assigning modules to machines, the research work presents a very similar scenario to that in this paper. Nonetheless, the role of AMs in production scheduling had not been discussed. This work extends the existing FJSP-CDST reported in [31] to a more general and realistic scheduling problem by integrating the AM selection with FJSP and subdividing a CDST into assembly and disassembly processes.

2.2. Resource constrained FJSP

Since AMs are scarce manufacturing resources for a shop floor, the FJSP-MR studied in this paper can be viewed as a special resource constrained scheduling problem. Referring to the existing literature, resource constrained FJSPs are divided into two categories: multiple resource constrained FJSP (MRC-FJSP) and dual resource constrained FJSP (DRC-FJSP) [19]. Most research efforts on the MRC-FJSP defined the auxiliary resources as tester, handler, and accessory, and were validated on a standard MRC-FJSP benchmark set given by the Decision Analysis Lab website. Some effective evolutionary algorithms were developed to solve the benchmark instances, such as a novel fruit fly optimization algorithm (nFOA) [49], a hybrid estimation of distribution algorithm (HEDA) [43], and a shuffled multi-swarm micro-migrating birds optimizer (SM^2-MBO) [18], where the SM^2-MBO showed better overall performance over other methods. The MRC-FJSP, considering three types of selected resources, usually exists in the semiconductor final testing scheduling, which is quite different from the FJSP-MR investigated in this work.

As discussed in [19], a dual resource-constrained scheduling problem considers a second resource (labour or auxiliary resource) besides the machine availability contraint, which seems similar to the FJSP-MR. However, the recent literature about DRC-FJSP specifies worker assignment as the additional sub-problem, where the processing time of an operation varies from workers with different proficiency [45]. Dhiflaoui et al. [10] reviewed recently published papers about the DRC-FJSP and classified the existing work by solution methods. Lei and Guo [24] employed an integrated encoding representation to describe all the sub-problems of the DRC-FJSP within a single vector, and proposed a variable neighborhood search (VNS) to address the problem. Andrade-Pineda et al. [2] established a bi-objective DRC-FJSP with makespan and due date-oriented criteria according to an automobile collision repair shop, where the expertise of workers was the main concern. Li and Liu [26] proposed a three-vector encoding strategy to represent sub-problems individually and demonstrated the feasibility of

operators was easy to guarantee under the encoding scheme. This paper developed an encoding representation for the FJSP-MR referring to some of the aforementioned literature.

Although both DRC-FJSP and FJSP-MR add a resource contraint to the classical FJSP, optimization ideas for the two problems are completely different. DRC-FJSP tries to assign a worker with higher proficiency to the corresponding operation to shorten the processing time, and transitions for a worker to manipulate different machines are neglected. On the contrary, transferring the auxiliary resource is the primary focus in the FJSP-MR and requires more targeted strategies.

2.3. Solution methods

The FJSP was initially discussed as a job shop scheduling problem (JSP) with machine flexibility [8], which addresses the machine assignment and operation sequencing simultaneously and has been proved to be NP-hard [7]. If an extra module selection sub-problem is considered, the production scheduling will be more complicated. Frequently-used approaches for combinatorial optimization problems with multiple decision-making processes can be divided into exact methods and approximate methods.

MILP is the most popular approach among mathematical modeling based exact methods [29]. There are usually four modeling ideas for establishing MILP models of scheduling problems [34]: position-based modeling, sequence-based modeling, time-indexed modeling, and adjacent sequence-based modeling. Position-based modeling defines a binary decision variable to identify whether a processing position on a machine is assigned to a certain operation to represent the operation sequencing of FJSPs. Fattahi et al. [14] first employed this idea to establish an MILP model for the FJSP with makespan criterion. Sequence-based modeling determines the operation sequence on a machine by their precedence relationships [37], thus it is suitable for the FJSP with sequence dependent setup time (SDST) [40]. Time-indexed modeling considers an available machine as a series of time intervals, and introduces a decision variable to identify whether an interval is applied to process a certain operation [44]]. Adjacent relation-based modeling, unlike the sequence-based modeling, only considers the relations of consecutive operations on a machine. Mousakhani [35] proposed an MILP model based on this idea to minimize the total tardiness for a flexible job shop with SDST. Obviously, the four modeling ideas have different applications [28]. For a scheduling problem with AMs, which can be viewed as a special kind of manufacturing resources, the sequence-based and adjacent relation-based modeling ideas are more suitable because the sequence or neighboring relation of two operations on a machine determines whether a reconfiguraion process is needed. As discussed in [27], the sequence-based modeling shows higher efficiency over the adjacent relation-based modeling in most cases, because it describes adjacent relations more concisely without restricting the unique predecessor or successor for an operation. Therefore, the MILP model in this paper is established based on operation sequences.

Intelligent optimization algorithms can efficiently find some sub-optimal solutions for large-scale combinatorial optimization problems, thus they are extensively applied to address the classic FJSP and some extended scheduling problems [17]. Lu et al. [30] proposed a knowledge-based multi-objective memetic algorithm for a job shop scheduling problem with energy-aware objectives, where the processing speed for a machine can be adjusted to reduce the energy consumption. Ghaleb et al. [20] addressed the real-time production scheduling and maintenance planning in a flexible job shop simultaneously, and developed a modified hybrid genetic algorithm (GA) with rescheduling rules to solve the proposed optimization model. As presented in [1], the main framework of GA and genetic operators are easy to implement, therefore GA becomes one of the most popular optimization algorithms for FJSPs. However, since evolution-based algorithms contain stochastic optimization mechanisms for simulating natural phenomena, the stability of these methods is questionable, which means the obtained solutions are not always satisfactory. To overcome the weakness, more and more research efforts are devoted to local search strategies for further exploring the solution space.

Table 1
$3 \times 3 \times 2$ FJSP-MR example.

Job	Operation	Processing Time		
		M_1	M_2	M_3
J_1	$O_{1,1}$	–	4 / 3(A_1)	–
	$O_{1,2}$	–	5	3(A_1)
	$O_{1,3}$	3(A_2)	–	4(A_1) / 2(A_2)
J_2	$O_{2,1}$	2(A_2)	–	5
	$O_{2,2}$	–	3	5 / 4(A_2)
J_3	$O_{3,1}$	5(A_1) / 3(A_2)	2(A_2)	4(A_2)
	$O_{3,2}$	–	3 / 2(A_1)	3
	$O_{3,3}$	4(A_2)	–	3(A_1)
	$O_{3,4}$	–	2(A_1) / 2(A_2)	3

The effectiveness of local search approaches strongly depends on the optimization objectives of problems. For example, since the makespan of a schedule cannot be optimized unless the critical path is destructed [38], almost all the local search methods for scheduling problems with makespan criterion are designed to re-assign operations on critical paths [13]. Compared to the makespan minimization, the total weighted tardiness (TWT) objective is less common in the existing literature about JSPs and FJSPs [46]. Singer and Pinedo [41] defined a disjunctive graph model for the JSP and employed a branch and bound algorithm to solve it. Pinedo and Singer [39] developed a shifting bottleneck (SB) method based on disjunctive graphs for the JSP with TWT criterion. Zhang and Wu [48] proposed a simulated annealing algorithm for the JSP, where a neighborhood structure focusing on the critical blocks of tardy jobs was designed. Although the these research articles deal with the JSP, definitions of disjunctive graph representation and critical paths for TWT objective are significant to FJSPs. Sobeyko and MÃűnch [42] hybridized a SB heuristic, a local search approach, and a variable neighborhood search for the FJSP with TWT objective, where a novel neighborhood structure was developed to change machine assignments for critical operations. In general, the disjunctive graph representation is necessary for local search methods, because it is able to avoid many-to-one situations caused by frequently-used encoding strategies. Meanwhile, the bottleneck for improving a given solution, i.e. critical path, is another important issue. Hence, this work establishes a disjunctive graph model for the FJSP-MR and proposes an iterated local search based on critical paths, where a searching process is guided by the weights of tardy jobs.

3. Problem formulation

The FJSP-MR addressed in this paper can be described as follows.

There are a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$ and a set of l AMs $A = \{A_1, A_2, \ldots, A_l\}$ in the shop floor. Some machines can be equipped with selected AMs to process different operations or to shorten the processing time. Given a set of n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ with corresponding release dates $R = \{r_1, r_2, \ldots, r_n\}$, due dates $D = \{d_1, d_2, \ldots, d_n\}$, and tardy weights $W = \{w_1, w_2, \ldots, w_n\}$, where a job J_i contains N_i operations $O_i = \{O_{i,1}, O_{i,2}, \ldots, O_{i,N_i}\}$ to be processed in accordance with the given sequence. An operation $O_{i,j}$ can be processed by a set of machines $M_{i,j} \in M$ with alternative AMs, and the processing time varies from different machine configurations. When an operation is finished, the AM used for processing this operation can be detached from the machine, and becomes available for other pending operations. The assembly time and disassembly time of an AM A_q cannot be neglected, and are denoted as α_q and β_q respectively. Additionally, the following requirements are also met in the FJSP-MR.

- All the machines and AMs are available from time 0.
- There is no preemption for all the jobs.
- A job can only have one operation being processed at a time.

- A machine can only process one operation at a time.
- An AM can only be attached to one machine at a time.
- A process cannot be interrupted once it starts.

A $3 \times 3 \times 2$ FJSP-MR instance is presented in Table 1, in which a symbol "–" suggests that the operation cannot be processed by any configuration of the machine. Notations in the brackets indicate the required AMs for processing the operations on corresponding machines, while the brackets are omitted if the operations can be directly processed by the machines. Alternative configurations are seperated by slashes. The assembly time and disassembly time for AMs are $\alpha = \{2, 1\}$ and $\beta = \{1, 2\}$ respectively. The delivery information and tardy weights for the jobs are: $R = \{3, 2, 4\}$, $D = \{8, 12, 10\}$, and $W = \{4, 1, 2\}$. This example will be used for explaining the encoding and decoding strategies in later sections.

Production scheduling in such a shop floor includes three decision-making processes, the machine assignment, AM selection, and operation sequencing, with the objective to minimize the TWT. After addressing the three sub-problems, the starting time and completion time of $O_{i,j}$, denoted by $s_{i,j}$ and $c_{i,j}$ respectively, can be easily identified, thereby a feasible schedule for the FJSP-MR is obtained. The TWT of a schedule can be calculated as $TWT = \sum_{i=1}^{n} w_i \cdot t_i$, where $t_i = max\{c_{i,n_i} - d_i, 0\}$.

In this section, a sequence-based MILP model is established, which helps to identify whether a reconfiguration process is requisite between two consecutive operations. Notations for the proposed model are predefined as follows.

Indice	
$i, i^{'}$	Indice of jobs, $i, i^{'} = 1, 2, \ldots, n$.
$j, j^{'}$	Indice of operations, $j, j^{'} = 1, 2, \ldots, N_i$.
$k, k^{'}$	Indice of machines, $k, k^{'} = 1, 2, \ldots, m$.
$q, q^{'}$	Indice of AMs, $q, q^{'} = 0, 1, 2, \ldots, l$.

Parameters	
N_i	Number of operations in J_i.
r_i	Release date of J_i.
d_i	Due date of J_i.
w_i	Tardiness weight of J_i.
$p_{i,j,k,q}$	Processing time of $O_{i,j}$ on M_k with A_q.
α_q	Assembly time of A_q.
β_q	Disassembly time of A_q.
$\gamma_{q,q^{'}}$	Reconfiguration time for a machine to disassembly$A_{q^{'}}$ and assembly A_q.
$\tau_{k,k^{'},q}$	Setup time for A_q to be detached from $M_{k^{'}}$ andattached to M_k.
$u_{i,j,k}$	$u_{i,j,k} = 1$, if $O_{i,j}$ can be processed by M_k;$u_{i,j,k} = 0$, otherwise.
$v_{i,j,k,q}$	$v_{i,j,k,q} = 1$, if A_q can be used to process $O_{i,j}$ on M_k;$v_{i,j,k,q} = 0$, otherwise.
Ω	A very large positive number.

Calculations of $\gamma_{q,q^{'}}$ and $\tau_{k,k^{'},q}$ are given as follows. It can be inferred that the reconfiguration time equals zero when a machine selects the same AM for consecutive operations, and the setup time is neglected if an AM is attached to process operations on the same machine. The parameters are introduced to describe the necessity for reconfiguration processes in the proposed MILP model.

$$\gamma_{q,q^{'}} = \begin{cases} 0 & , \quad q = q^{'}; \\ \beta_{q^{'}} + \alpha_q & , \quad otherwise. \end{cases}$$

$$\tau_{k,k^{'}q} = \begin{cases} 0 & , \quad k = k^{'}; \\ \beta_q + \alpha_q & , \quad otherwise. \end{cases}$$

Decision variables	
t_i	Tardiness of J_i.
$c_{i,j}$	Completion time of $O_{i,j}$.
$x_{i,j,k}$	$x_{i,j,k} = 1$, if $O_{i,j}$ is processed by M_k;$x_{i,j,k} = 0$, otherwise.
$y_{i,j,k,q}$	$y_{i,j,k,q} = 1$, if $O_{i,j}$ is processed on M_k with A_q;$y_{i,j,k,q} = 0$, otherwise.
$\eta_{i,j,i^{'}j^{'}}$	$\eta_{i,j,i^{'}j^{'}} = 1$, if $O_{i,j}$ is processed after $O_{i^{'}j^{'}}$ on a machine; $\eta_{i,j,i^{'}j^{'}} = 0$, otherwise.
$\sigma_{i,j,i^{'}j^{'}}$	$\sigma_{i,j,i^{'}j^{'}} = 1$, if an AM is attached to $O_{i,j}$ afterbeing attached to $O_{i^{'}j^{'}}$; $\sigma_{i,j,i^{'}j^{'}} = 0$, otherwise.

It is important to note that a dummy AM A_0 is introduced in the proposed MILP model for consistency. If an operation can be directly processed by a machine, the required AM is assumed to be A_0. Obviously, both α_0 and β_0 are set to zero. Constraints of the MILP model are listed below.

$$Min\ TWT = \sum_{i=1}^{n} w_i \cdot t_i$$

s.t.

$$t_i \geq c_{i,N_i} - d_i \qquad \forall i \tag{1}$$

$$t_i \geq 0 \qquad \forall i \tag{2}$$

Constraints (1) and (2) determine the tardiness for each job.

$$\sum_{k=1}^{m} x_{i,j,k} = 1 \qquad \forall i,j \tag{3}$$

$$x_{i,j,k} \leq u_{i,j,k} \qquad \forall i,j,k \tag{4}$$

Constraint (3) ensures an operation to be scheduled exactly once. Constraint (4) guarantees that an operation is assigned to an applicable machine.

$$\sum_{q=0}^{l} y_{i,j,k,q} = x_{i,j,k} \qquad \forall i,j,k \tag{5}$$

$$y_{i,j,k,q} \leq v_{i,j,k,q} \qquad \forall i,j,k,q \tag{6}$$

Constraint (5) and (6) force an operation to select an eligible AM on the corresponding machine.

$$c_{i,j} \geq r_i + \sum_{k=1}^{m} \sum_{q=0}^{l} (p_{i,j,k,q} \cdot y_{i,j,k,q}) \qquad \forall i, j = 1 \tag{7}$$

Constraint (7) avoids the first operation of a job from being processed before the release date.

$$c_{i,j} \geq c_{i,j-1} + \sum_{k=1}^{m} \sum_{q=0}^{l} (p_{i,j,k,q} \cdot y_{i,j,k,q}) \qquad \forall i, j > 1 \tag{8}$$

Constraint (8) restricts that an operation cannot be processed until its job predecessor operation has been finished.

$$c_{i,j} \geq \sum_{k=1}^{m} \sum_{q=0}^{l} (p_{i,j,k,q} + \alpha_q) \cdot y_{i,j,k,q} \qquad \forall i,j \tag{9}$$

Constraint (9) prevents an operation from being processed before the selected AM is attached, mainly focusing on the operations without machine predecessors.

$$c_{i,j} \geq c_{i^{'},j^{'}} + p_{i,j,k,q} + \gamma_{q,q^{'}} - \left(3 - y_{i,j,k,q} - y_{i^{'},j^{'},k,q^{'}} - \eta_{i,j,i^{'}j^{'}}\right) \cdot \Omega \quad \forall i, i^{'}, j, j^{'}, k, q, q^{'}, O_{i,j} \neq O_{i^{'}j^{'}} \tag{10}$$

$$c_{i^{'}j^{'}} \geq c_{i,j} + p_{i^{'}j^{'},k,q^{'}} + \gamma_{q^{'},q} - \left(2 - y_{i,j,k,q} - y_{i^{'},j^{'},k,q^{'}} + \eta_{i,j,i^{'}j^{'}}\right) \cdot \Omega \quad \forall i, i^{'}, j, j^{'}, k, q, q^{'}, O_{i,j} \neq O_{i^{'}j^{'}} \tag{11}$$

Constraints (10) and (11) avoid the overlap of operations processed by the same machine. When an operation is finished, the next operation

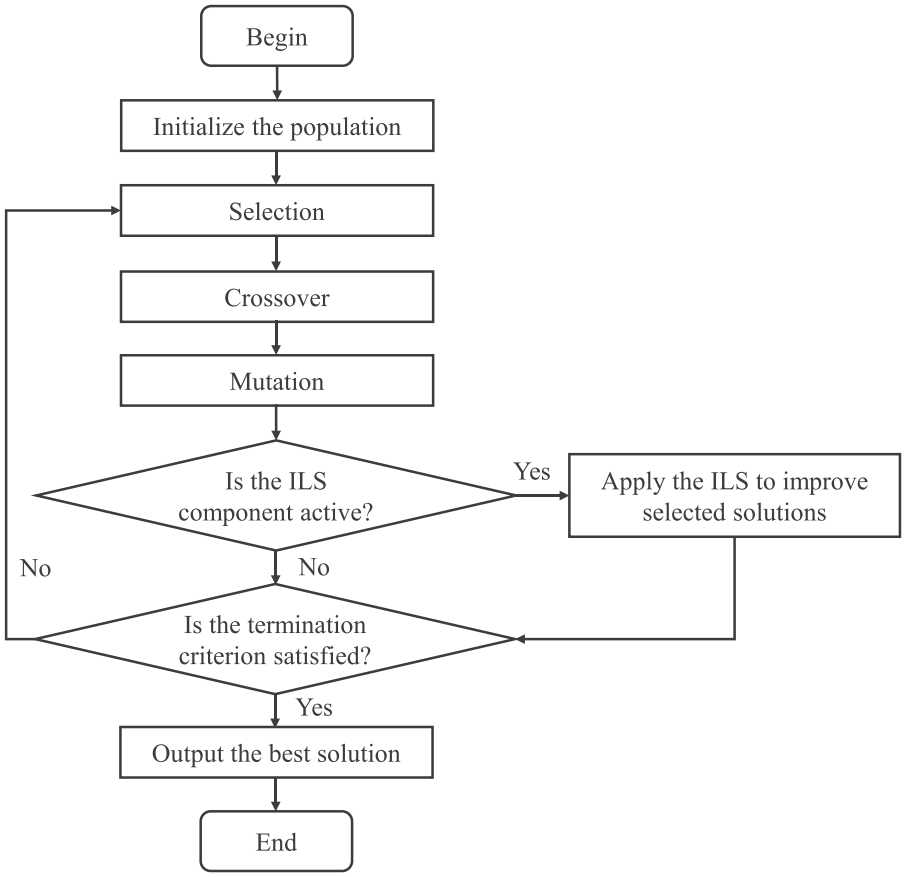

Fig. 1. Workflow of the proposed IGA.

on the machine cannot be processed before the machine configuration has been transformed. If the two consecutive operations select the same AM, the reconfiguration can be left out and the next operation is allowed to be started immediately.

$$c_{i,j} \geq c_{i',j'} + p_{i,j,k,q} + \tau_{k,k'q} - \left(3 - y_{i,j,k,q} - y_{i',j',k',q} - \sigma_{i,j,i',j'}\right)\cdot\Omega \quad \forall i, i', j, j', k, k', q \neq 0, O_{i,j} \neq O_{i',j'} \tag{12}$$

$$c_{i',j'} \geq c_{i,j} + p_{i',j',k',q} + \tau_{k',k,q} - \left(2 - y_{i,j,k,q} - y_{i',j',k',q} + \sigma_{i,j,i',j'}\right)\cdot\Omega \quad \forall i, i', j, j', k, k', q \neq 0, O_{i,j} \neq O_{i',j'} \tag{13}$$

Constraints (12) and (13) ensure the non-overlapping operations selecting the same AM (except A_0). An operation is allowed to use an AM when it has been detached from the previous machine and attached to the current machine. If the AM is just on the current machine, the setup time equals zero. It is noteworthy that only when two operations are adjacent on the machine and choose the same AM successively, the reconfiguration process can be omitted.

4. Improved genetic algorithm

4.1. Framework of the proposed improved genetic algorithm

Genetic algorithm (GA) was proposed by John Hollad [21]], and has become one of the most commonly-used meta-heuristics. GA improves solutions in a population through three phases, selection, crossover, and mutation, simulating the process of biological evolution. This paper adopts the classical GA as the main optimization framework, and introduces an iterated local search (ILS) in the later period for further improvements. Fig. 1 shows the workflow of the proposed improved genetic algorithm (IGA).

Primary concerns for applying GA to combinatorial optimization problems are encoding and decoding strategies, GA operators, and problem-focused local search methods. To overcome these difficulties, this work designs a two-vector encoding strategy for three decision-making processes of the FJSP-MR. Afterwards, a semi-active decoding method is employed to generate feasible schedules represented by the encoding scheme. Effective GA operators in the existing literature are adopted after some adjustments to machine configuration vectors. Moreover, an ILS is developed based on characteristics of the FJSP-MR, and is activated in the second half of IGA iterations.

4.2. Encoding and decoding

4.2.1. Two-vector encoding

Generally, an individual in a GA population is represented by several chromosomes (usually vectors or matrices for scheduling problems), which can be decoded to a feasible scheduling plan and modified by GA operators. The FJSP-MR contains three decision-making processes, thus a solution formally needs three encoding vectors to cover all the sub-problems. Since the AM selection depends on the machine assignment, this paper proposed a two-vector encoding scheme to represent the configuration selection (CS) and operation sequencing (OS), where a CS vector includes the information for both machine assignment and AM selection.

A CS vector indicates the configurations for all the operations sequentially through an array of tuples, each of which contains two elements. The first element and second element suggest the assigned machine and selected AM respectively. {⟨2, 1⟩, ⟨3, 1⟩, ⟨1, 2⟩, ⟨3, 0⟩, ⟨2, 0⟩, ⟨1, 2⟩, ⟨2, 0⟩, ⟨3, 1⟩, ⟨2, 2⟩} is a legal CS vector for the instance given

in Table 1, and represents the configurations for $\{O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{2,2}, O_{3,1}, O_{3,2}, O_{3,3}, O_{3,4}\}$ correspondingly. For example, the third element of the CS vector implies that operation $O_{1,3}$ is processed by machine M_1 with AM A_2. As mentioned in the MILP model, selecting A_0 means an operation can be processed by the machine without any AM.

The representation of an OS vector in this paper can be referred to [25]]. Simply, an OS vector consists of serial numbers of jobs, which indicate the operations to be scheduled by the numbers of appearances. For example, {2, 1, 3, 2, 1, 3, 1, 3, 3} is a legal OS vector for the FJSP-MR instance in Table 1, and suggests the scheduling sequence of operations $\{O_{2,1}, O_{1,1}, O_{3,1}, O_{2,2}, O_{1,2}, O_{3,2}, O_{1,3}, O_{3,3}, O_{3,4}\}$.

4.2.2. Semi-active decoding

According to the discussions in [38]], this paper applies the classical semi-active decoding strategy to the two-vector encoding scheme. Semi-active decoding sequentially visits an OS vector to find the operation to be assigned, and then identifies the processing configuration by the CS vector. After that, it is necessary to calculate the starting time and completion time for the operation to obtain a partial scheduling plan. In the FJSP-MR, the starting time for an operation $O_{i,j}$ depends on its job predecessor operation $JP_{O_{i,j}}$, machine predecessor operation $MP_{O_{i,j}}$, and AM predecessor operation $AP_{O_{i,j}}$, completion times of which are denoted as $c_{JP_{O_{i,j}}}$, $c_{MP_{O_{i,j}}}$, and $c_{AP_{O_{i,j}}}$ respectively. Pseudo codes of the semi-active decoding strategy are given in Algorithm 1, where $CS_p(X)$ and $OS_p(X)$ represent the p^{th} elements in the CS vector and OS vector of solution X respectively.

Algorithm 1. Semi-active decoding for the two-vector encoding scheme.

Input: CS and OS vectors of solution X
Output: A feasible schedule for solution X represented by the CS and OS vectors

for $p = 1$ *to* $\sum_i^n N_i$ **do**
  $i = OS_p(X)$
  This is j^{th} time that i appears in the OS vector
  $\Rightarrow O_{i,j}$ is the operation to be scheduled
  if $j = 1$ **then**
    $c_{JP_{O_{i,j}}} = r_i$
  else
    $c_{JP_{O_{i,j}}} = c_{i,j-1}$
  end
  Check the position of $O_{i,j}$ in the CS vector
  $\Rightarrow O_{i,j}$ is assigned to be processed by M_k with A_q
  $\Rightarrow$ Identify the last operation on M_k as $MP_{O_{i,j}}$
  $\Rightarrow$ Identify the last operation using A_q as $AP_{O_{i,j}}$
  if $MP_{O_{i,j}} == AP_{O_{i,j}}$ **then**
    $s_{i,j} = max\{c_{O_{JP_{i,j}}}, c_{O_{MP_{i,j}}}\}$
  else
    Identify $MP_{O_{i,j}}$ is processed by M_k with $A_{q'}$
    $s_{i,j} = max\{c_{JP_{O_{i,j}}}, c_{O_{MP_{i,j}}} + \beta_{q'} + \alpha_q, c_{AP_{O_{i,j}}} + \beta_q + \alpha_q\}$
  end
  $c_{i,j} = s_{i,j} + p_{i,j,k,q}$
end

Following the semi-active decoding strategy, encoding vectors in the previous section, i.e., CS vector = {⟨2, 1⟩, ⟨3, 1⟩, ⟨1, 2⟩, ⟨3, 0⟩, ⟨2, 0⟩, ⟨1, 2⟩, ⟨2, 0⟩, ⟨3, 1⟩, ⟨2, 2⟩}, OS vector = {2, 1, 3, 2, 1, 3, 1, 3, 3}, can be decoded to the schedule shown in Fig. 2, in which a white block represents a reconfiguration process for replacing AMs, and an arrow indicates an AM transfer between machines.

4.3. Selection

Selection phase tries to keep the solutions with good objective values to the next iteration, thereby applying GA operators and the local search to these solutions is more likely to obtain high-quality results. In this paper, elitist selection and tournament selection strategies are performed successively. The elitist selection chooses *eSize* solutions with minimal TWT values and directly adds them to the new population, where *eSize* denotes the elitist size. The tournament selection randomly picks out *tSize* solutions from the current population at a time, and add the best solution among them to the next population.

During the selection stage, the proposed IGA first obtains *eSize* solutions by the elitist strategy. Afterwards the tournament selection is performed repeatedly until the size of the next population reaches *popSize*. The selection pressure can be adjusted by tuning the parameters *eSize* and *tSize*.

4.4. Crossover

Crossover phase shares the information of high-quality solutions by manipulating their encoding vectors to generate new solutions. It randomly chooses two solutions from the population obtained by the selection phase at a time, and performs GA operators to construct new solutions with a crossover probability C_r. To ensure the feasibility of generated solutions, GA operators for CS vectors and OS vectors are different.

A two-point crossover strategy, illustrated in Fig. 3, is taken to obtain new CS vectors. If the configurations represented by parents X_{p1} and X_{p2} are legal, the feasibility of generated CS vectors is easy to guarantee. GA operators for OS vectors are precedence operation crossover (POX) and job-based crossover (JBX) referring to [25]].

4.5. Mutation

Mutation phase introduces random disturbances to avoid the premature convergence of current population. It randomly modifies encoding vectors to generate new solutions with a mutation probability M_r. When solutions are trapped to local minima, mutation strategies will possibly lead the solutions to jump out and to improve the exploration. A random mutation operator is performed on CS vectors as shown in Fig. 4, where an element shifts to another available configuration for the corresponding operation with 50% probability. A grey block indicates the element will change to another processing configuration, while white blocks stay the same in the CS vector of child X_c. For OS vectors, the frequently-used swap operator is adopted as reported in [25]].

4.6. Iterated local search for the FJSP-MR

4.6.1. Framework of the ILS

The proposed IGA introduces an ILS based on a disjunctive graph representation for the FJSP-MR to address many-to-one situations caused by the encoding scheme. Firstly, the disjunctive graph model is established and calculations of path lengths are defined according to characteristics of the FJSP-MR. Secondly, critical paths for the TWT objective are analyzed for improving the searching efficiency. Finally, a modified neighborhood structures is developed and performed on critical operations. Pseudo codes for the proposed ILS are given in Algorithm 2, where *iterMax* represents the maximal number of ILS iterations.

The proposed ILS is performed on the *popSize*/2 solutions with good TWT values. It applies a neighborhood structure to re-assign selected operations on a critical path to generate neighboring solutions, and selects the best candidate as the initial solution of the next ILS iteration. A searching procedure will terminate when the iteration reaches *iterMax* or the current solution cannot be improved by re-constructing any critical path. Important components of the proposed ILS, the disjunctive graph model, critical paths, and modified neighborhood structure, are explained in following subsections.

Algorithm 2. Procedure for the ILS.

Input: An initial solution X
Output: A solution X' obtained by applying the ILS to solution X

```
X' = X
while iter ≤ iterMax do
    Find critical paths for X', and sort the critical paths by a metric (see section 4.6.3)
    flag = false        \\ identify whether solution X' has been improved
    foreach critical path in X' do
        Apply the neighborhood structure to selected operations on the critical path (see section 4.6.4)
        ⇒ Obtain a set of neighboring solutions
        ⇒ Choose the best solution X_local from candidate solutions
        iter++
        if X_local is better than X' then
            replace X' by X_local
            flag = true
            break
        end
    end
    if flag==false then
        break
    end
end
return solution X'
```

4.6.2. Disjunctive graph model for the FJSP-MR

A feasible scheduling plan of an FJSP can be represented by an acyclic disjunctive graph with nodes and directed arcs [38]], where a node refers to an operation and an arc connects two adjacent operations from the same job or processed by the same machine. For the FJSP-MR, the third kind of arcs are required to present relations for the operations

selecting the same AM. This paper proposed a disjunctive graph model with three types of arcs extended from [36]], and calculations of path lengths are re-defined for the FJSP-MR. The disjunctive graph for the solution in Fig. 2 is depicted as Fig. 5.

As shown in Fig. 5, the disjunctive graph contains two dummy nodes, O_{begin} and O_{end}, to present the beginning and end of the schedule. For a real operation node $O_{i,j}$, there are at least four adjacent nodes connected by arcs: job predecessor operation $JP_{O_{i,j}}$, job successor operation $JS_{O_{i,j}}$, machine predecessor operation $MP_{O_{i,j}}$, and machine successor operation $MS_{O_{i,j}}$. When an AM is required for processing $O_{i,j}$, an AM predecessor operation $AP_{O_{i,j}}$ and AM successor operation $AS_{O_{i,j}}$ are considered. The dummy node O_{begin} leads the operations which have no job/ machine/ AM predecessor with corresponding arcs. And conversely, the nodes without certain successor operations point the arcs to O_{end}. The path length between two neighboring nodes equals the processing time of the previous operation plus the required setup time. Head and tail lengths for $O_{i,j}$ are defined as $R_{O_{i,j}}$ and $Q_{O_{i,j}}$ respectively for explaining the modified neighborhood structure, where $R_{O_{i,j}}$ indicates the longest path from O_{begin} to $O_{i,j}$, and $Q_{O_{i,j}}$ refers to the longest path from $O_{i,j}$ to O_{end}. Calculations of $R_{O_{i,j}}$ and $Q_{O_{i,j}}$ for the proposed disjunctive graph model are presented as follows.

$$R_{O_{i,j}} = \begin{cases} max\{R_{JP_{O_{i,j}}} + p_{JP_{O_{i,j}}}, R_{MP_{O_{i,j}}} + p_{MP_{O_{i,j}}}, R_{AP_{O_{i,j}}} + p_{AP_{O_{i,j}}}\}, & MP_{O_{i,j}} = AP_{O_{i,j}}; \\ max\{R_{JP_{O_{i,j}}} + p_{JP_{O_{i,j}}}, R_{MP_{O_{i,j}}} + p_{MP_{O_{i,j}}} + \beta_{q_{MP_{O_{i,j}}}} + \alpha_{q_{O_{i,j}}}, R_{AP_{O_{i,j}}} + p_{AP_{O_{i,j}}} + \beta_{q_{O_{i,j}}} + \alpha_{q_{O_{i,j}}}\}, & otherwise. \end{cases}$$

$$Q_{O_{i,j}} = \begin{cases} max\{Q_{JS_{O_{i,j}}} + p_{JS_{O_{i,j}}}, Q_{MS_{O_{i,j}}} + p_{MS_{O_{i,j}}}, Q_{AS_{O_{i,j}}} + p_{AS_{O_{i,j}}}\}, & MS_{O_{i,j}} = AS_{O_{i,j}}; \\ max\{Q_{JS_{O_{i,j}}} + p_{JS_{O_{i,j}}}, Q_{MS_{O_{i,j}}} + p_{MS_{O_{i,j}}} + \beta_{q_{O_{i,j}}} + \alpha_{q_{MS_{O_{i,j}}}}, Q_{AS_{O_{i,j}}} + p_{AS_{O_{i,j}}} + \beta_{q_{O_{i,j}}} + \alpha_{q_{O_{i,j}}}\}, & otherwise. \end{cases}$$

Since all the solutions are active schedules for the FJSP-MR, the head length of an operation can be simply identified as the starting time. Additionally, to make notations more concise, the processing time of operation $O_{i,j}$ in the given solution is denoted by $p_{O_{i,j}}$, and $q_{O_{i,j}}$ indicates the AM attached to $O_{i,j}$. If the reconfiguration between two operations can be left, calculations of $R_{O_{i,j}}$ and $Q_{O_{i,j}}$ are the same as those in the classical FJSP, otherwise the assembly time and disassembly time of corresponding AMs should be added. All the path lengths are given in Fig. 5 as an example.

4.6.3. Critical paths for the TWT objective

The concept of critical path originates from the classical JSP with makespan criterion and has been thoroughly applied to local search methods for FJSPs [13]. Critical path is the longest path from O_{begin} to O_{end} in a disjunctive graph. If a move cannot destruct the critical path in a given schedule, the makespan of the solution will not be optimized. However, for a scheduling problem with due date oriented objective, each of tardy jobs leads to a critical path [38], thus a local search strategy should not only focus on the last operation of the whole schedule. As discussed in the Literature Review section, there are already some research articles about critical paths for JSPs or FJSPs with the TWT objective. However, the existing literature treats all the critical paths as the same without considering tardy weights. Besides, since disjunctive graphs for the FJSP-MR contain AM arcs, the way to identify critical paths is also different.

This work employs a backstepping method to find critical paths, which means it starts from the last operation of a tardy job and recursively looks for the predecessor operation restricting the starting time of current operation. Taking the solution in Fig. 2 as an example, $O_{1,3}$ is the last operation of J_1 and has two predecessors $O_{1,2}$ and $O_{3,1}$. It is clear that $O_{1,3}$ cannot be processed earlier due to $O_{1,2}$ ($R_{O_{1,2}} + p_{O_{1,2}} > R_{O_{3,1}} + p_{O_{3,1}}$), thus $O_{1,2}$ belongs to the critical path led by the tardy job J_1. According to this procedure, a critical path can be identified as {$O_{2,1}$, $O_{1,2}$, $O_{1,3}$}. The most frequently-used method, which checks whether $R_{O_{i,j}} + p_{O_{i,j}} + Q_{O_{i,j}}$ equals the makespan or c_{i,N_i}, is not applicable to the FJSP-MR. $R_{O_{i,j}}$ and $Q_{O_{i,j}}$ are obtained from a global perspective, which makes them probably greater than the longest path (completion time) for a tardy job. The backstepping calculation is a more straightforward method compared to the common one.

If there is more than one critical path in a schedule, searching all the critical paths helps to the stability of algorithms [13]. Nevertheless, concerning the number of tardy jobs, a large portion of operations is involved, which will decline the efficiency of ILS. Thus it is necessary to decide which critical path to be visited first. For the TWT objective, the importance of a critical path depends on two factors, the tardy weight w_i and tardiness t_i. The former reveals the urgency of a tardy job, while the latter demonstrates the room for improvements. The proposed ILS sorts all the critical paths in a descending sequence by $w_i \cdot (c_{i,N_i} - d_i)$, so that the jobs with higher priorities can be preferentially considered. Once a better solution is found, the ILS will skip the remaining critical paths with less importance, and accept the new solution as the initial point for the next iteration.

4.6.4. Modified neighborhood structure

The neighborhood structure for the proposed ILS is modified from the *k*-insertion approach [32], which was originally developed to address the FJSP for makespan minimization. A *k*-insertion procedure first deletes the machine arcs of a critical operation, and then checks another available machine for a set of feasible positions to insert the operation. After evaluating all the objective values of possible solutions, the operation will be assign to the best position. A *k*-insertion move can easily generate a group of neighboring solutions by altering the machine assignment of a critical operation. However, for the FJSP-MR, only re-linking the machine arcs for an operation is insufficient to ensure the feasibility. Since the head length, tail length, and attached AM for the operation are likely to be changed, the AM arcs needs to be re-constructed as well.

An alternative way to identify the position on an AM is to remove all the AM arcs in the graph temporarily. When the machine assignment is determined, operation sequences on AMs can be obtained according to updated head lengths. It is obvious that this method only generates one neighboring solution, and is too time-consuming for re-calculating all the AM arcs. Therefore, feasible insertions in terms of AMs require to be discussed. The procedure for the modified neighborhood structure is presented in Algorithm 3.

Algorithm 3. Procedure for the modified neighborhood structure.

Input: A critical path for the initial solution X
Output: A solution X' obtained by the modified neighborhood structure
Define a set Φ for neighboring solutions
foreach *critical operation* $O_{i,j}$ *on the critical path of solution* X **do**
  if $MP_{O_{i,j}} == AP_{O_{i,j}}$ *or* $MS_{O_{i,j}} == AS_{O_{i,j}}$ **then**
    continue
  end
  Choose an alternative configuration $<M_k, A_q>$ for $O_{i,j}$ randomly
  $\Rightarrow$ Obtain a set of feasible insertions Γ_k by applying the k-insertion to $O_{i,j}$ and M_k
  $\Rightarrow$ Obtain a set of feasible insertions Γ_q by applying the k-insertion to $O_{i,j}$ and A_q
  foreach *insertion* ϕ_k *in* Γ_k **do**
    foreach *insertion* ϕ_q *in* Γ_q **do**
      if ϕ_k *and* ϕ_q *are conflicting* **then**
        continue (see Figure 6)
      else
        Generate a neighboring solution $\bar{X}$ by performing ϕ_k and ϕ_q
        Add solution $\bar{X}$ to set Φ
      end
    end
  end
end
return the best solution X' in Φ

In correspondence with the original k-insertion procedure, feasible positions for assigning an operation to another AM can be identified by classifying operations on the AM, where head and tail lengths for AMs are calculated according to the proposed disjunctive graph model for the FJSP-MR. After applying the k-insertion to a machine and an AM, two sets of feasible insertions, Γ_k and Γ_q, on the machine and the AM are identified respectively, so that a combination of insertions ϕ_k and ϕ_q leads to a candidate solution $\bar{X}$. However, according to the proofs in [32], ϕ_k only guarantees that there is no cycle between job arcs and machine arcs. Similarly, ϕ_q can ensure the paths between job arcs and AM arcs are acyclic. Therefore, it is necessary to check whether the updated machine arcs and AM arcs for ϕ_k and ϕ_q are conflicting, thus ensuring the feasibility of $\bar{X}$. An example for inserting operation O_ω is illustrated in Fig. 6.

As shown in Fig. 6, a pair of insertions, ϕ_k and ϕ_q, link MP_{O_ω}, MS_{O_ω}, AP_{O_ω}, and AS_{O_ω}, to O_ω to generate a neighboring solution. For a predecessor operation (MP_{O_ω} or AP_{O_ω}), there must be a path leading to O_ω. Hence, if there is no inflow arc from O_ω to the predecessor operation, the feasibility can be guaranteed. Take MP_{O_ω} as an example, by the proofs in [32], if Condition 1 is satisfied, there is no path from O_ω to MP_{O_ω}. Since the insertion ϕ_k has already been proved to be feasible, $Q_{MP_{O_\omega}} + p_{MP_{O_\omega}} > Q_{JS_{O_\omega}} + p_{JS_{O_\omega}}$ and $Q_{MP_{O_\omega}} + p_{MP_{O_\omega}} > Q_{MS_{O_\omega}} + p_{MS_{O_\omega}}$ are unquestionably valid, because the tail lengths for any predecessor operation should be greater than those of successor operations. Therefore, Condition 1 can be simplified to Condition 2. Similarly, for the AM predecessor AP_{O_ω}, Condition 3 ensures that there is no inflow arc from O_ω. Constraints for successor operations (MS_{O_ω} and AS_{O_ω}) can be referred to the definition of operation sets raised by the original k-insertion, then Conditions 4 and 5 are obtained to guarantee the feasibility of outflow arcs for O_ω. Once the four conditions are satisfied, the combination of ϕ_k and ϕ_q will be applied to generate a candidate solution $\bar{X}$.

Condition 1. $Q_{MP_{O_\omega}} + p_{MP_{O_\omega}} > max\{Q_{JS_{O_\omega}} + p_{JS_{O_\omega}}, Q_{MS_{O_\omega}} + p_{MS_{O_\omega}}, Q_{AS_{O_\omega}} + p_{AS_{O_\omega}}\}$.

Condition 2. $Q_{MP_{O_\omega}} + p_{MP_{O_\omega}} > Q_{AS_{O_\omega}} + p_{AS_{O_\omega}}$.

Condition 3. $Q_{AP_{O_\omega}} + p_{AP_{O_\omega}} > Q_{MS_{O_\omega}} + p_{MS_{O_\omega}}$.

Condition 4. $R_{MS_{O_\omega}} + p_{MS_{O_\omega}} > R_{AP_{O_\omega}} + p_{AP_{O_\omega}}$.

Condition 5. $R_{AS_{O_\omega}} + p_{AS_{O_\omega}} > R_{MP_{O_\omega}} + p_{MP_{O_\omega}}$.

For example, as shown in Fig. 2, $O_{2,1}$ is a critical operation and can be alternatively processed by M_1 with A_2. Therefore, the modified neighborhood structure visits two sets of operations, the operations on M_1 and the operations using A_2, and applies the procedure proposed by [32] to find two sets of feasible insertions. After evaluating the conflicts for all the insertion combinations, the modified neighborhood structure will allocate $O_{2,1}$ to the best position by connecting machine arcs and AM arcs to the corresponding nodes. It is noteworthy that the local search will not be performed on the critical operations without reconfiguration processes (Row 3, Algorithm 3). The strategy leaves out a lot of candidate solutions for efficiency, and is likely to keep some good assignments of the initial solutions.

5. Numerical experiment

5.1. Experimental setup

In this subsection, three groups of test instances, with different

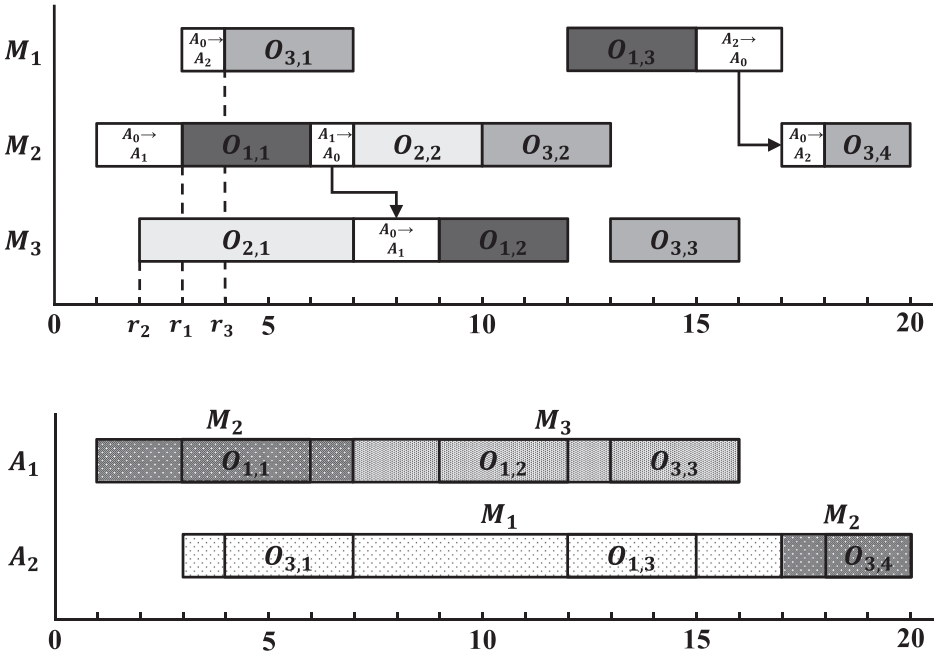

Fig. 2. Gantt charts of machines and AMs for the 3 × 3 × 2 FJSP-MR instance.

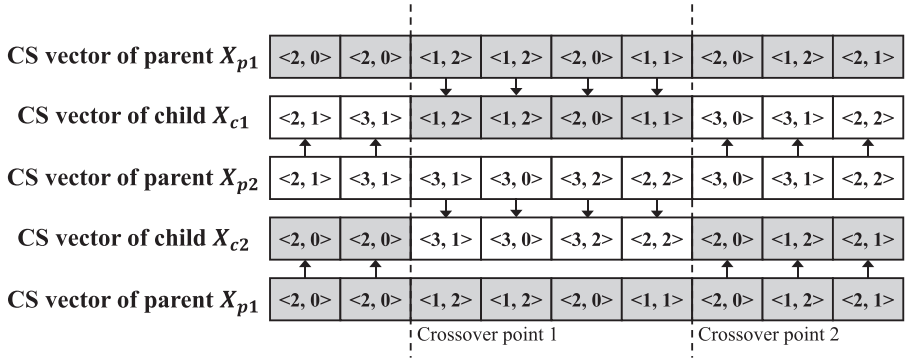

Fig. 3. Illustration of the two-point crossover operator for CS vectors.

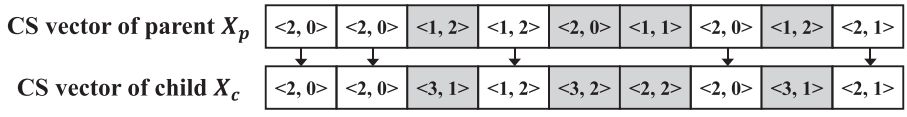

Fig. 4. Illustration of the random mutation operator for CS vectors.

parameter value ranges, are generated according to the existing literature. The FJSP-MR instances will be employed to investigate the efficiency of comparative algorithms in different scenarios. After that, significant parameters of the proposed IGA are calibrated by a full factorial test, and are adopted for numerical experiments.

5.1.1. Test instances

To evaluate the performance of the proposed MILP model and IGA, twenty instances from the well-known FJSP benchmark sets, Fdata (MFJS01-10) [14] and BRdata (Mk01–10) [7], are extended to the form of FJSP-MR, and then taken for numerical experiments. Since Fdata and BRdata were originally generated for the FJSP with makespan criterion, the supplementary data for FJSP-MR instances includes release dates, due dates, tardy weights, and the information of AMs. A parameter *flex* is also introduced for each instance to indicate the AM flexibility. For example, *flex* = 2 means an operation can be processed by a machine with two different configurations. The test data generation in this paper refers to [9]. Two auxiliary parameter $p_{i,j}^{min}$ and p^{max} are pre-defined, where $p_{i,j}^{min}$ denotes the shortest processing time of operation $O_{i,j}$ among all the possible configurations, and p^{max} represents the longest processing time for all the operations. Value ranges for the parameters are shown below, where all the numbers are rounded to integers.

- Number of AMs $l \in [m/4, m/2]$ for all the instances.
- Release date $r_i \in [0, p^{max}]$ for all the instances.
- Due date d_i varies from controlled groups: (1) $d_i = 1.2 \cdot \sum_{j=1}^{N_i} p_{i,j}^{min}$ for group A and C; (2) $d_i = 1.0 \cdot \sum_{j=1}^{N_i} p_{i,j}^{min}$ for group B.

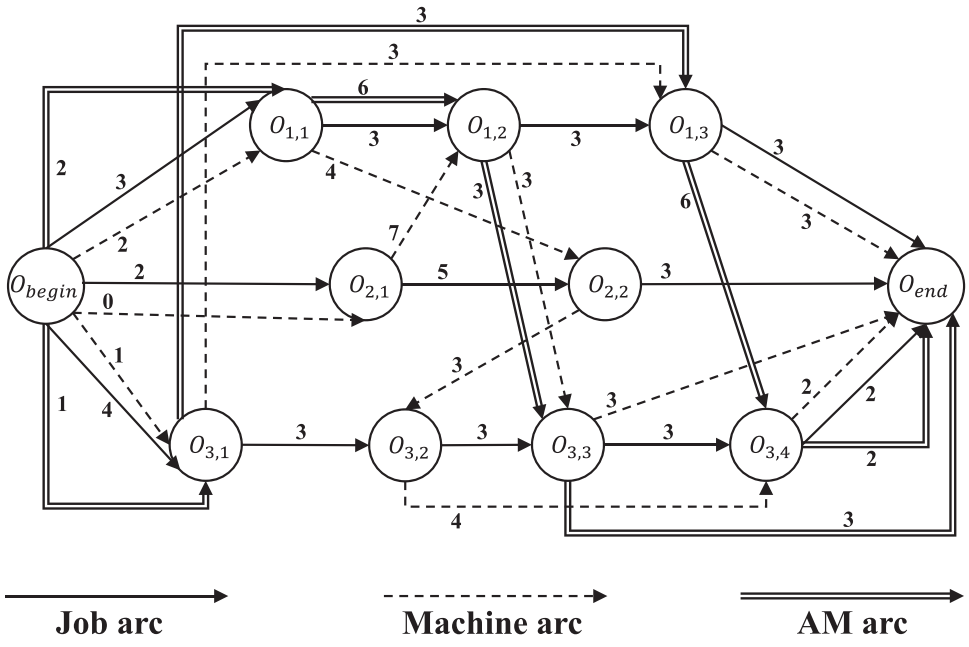

Fig. 5. Disjunctive graph for the feasible FJSP-MR schedule in Fig. 2.

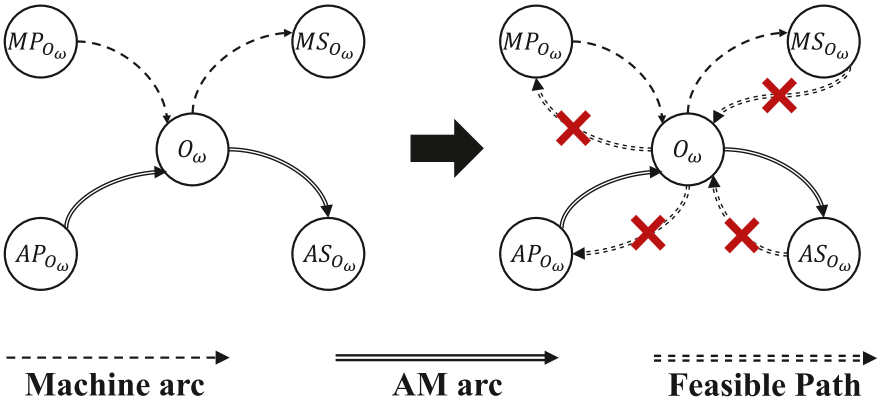

Fig. 6. Illustration for ensuring the feasibility for re-linking machine arcs and AM arcs of O_ω.

- Assembly time α_q varies from original benchmark sets: (1) $\alpha_q \in [p^{max}, 2 \cdot p^{max}]$ for Fdata instances; (2) $\alpha_q \in [2 \cdot p^{max}, 4 \cdot p^{max}]$ for BRdata instances.
- Disassembly time $\beta_q \in [0.8 \cdot \alpha_q, 1.2 \cdot \alpha_q]$ for all the instances.
- AM flexibility *flex* varies from controlled groups: (1) $flex = 2$ for group A and B; (2) $flex = 3$ for group C.

For all the FJSP-MR instances, numbers of AMs and release dates are generated in the same way, where l is set to be significantly less than m to ensure that AMs are inadequate for RMTs. Assembly time α_q for an AM varies from source benchmark instances, and the disassembly time β_q is predictably close to α_q. The only difference between Group A and Group B is the value range for due dates, which is set to investigate the performance of comparative algorithms under urgent delivery requirements. While the AM selection of Group C is clearly more flexible than that of Group A. All the generated instances are with a suffix "-MR" and corresponding group identifiers.

5.1.2. Parameter setting

According to preliminary experiments, the population size *popSize*, tournament size *tSize*, crossover probability C_r, and mutation probability M_r, are chosen for the parameter tuning. Since the ILS can be viewed as a special variable neighborhood search (VNS) approach and the proposed IGA will be compared with a GA with VNS component (GA-VNS) [45], the parameter *iterMax* for the ILS is set to 60 as the GA-VNS, so that differences on the algorithm performance primarily depend on neighborhood structures. The number of elite solutions *eSize* will be identified after *popSize* is calibrated.

A full factorial test is adopted to tune significant parameters of the IGA. Parameter levels are set as: *popSize* ∈ {100, 200, 300, 400, 500, 600}, *tSize* ∈ {2, 3, 4, 5, 6, 7}, $C_r \in \{0.75, 0.8, 0.85, 0.90, 0.95\}$, and $M_r \in \{0.05, 0.10, 0.15, 0.20, 0.25\}$. Hence, there are $6 \times 6 \times 5 \times 5 = 900$ different parameter combinations, each of which is tested on all the FJSP-MR instances generated in Section 5.1.1 with a termination criterion $10 \cdot n \cdot m \cdot l$ milliseconds. The relative percent deviation (RPD) is employed to evaluate results provided by different parameter levels. As presented in Eq. (14), F indicates the objective value obtained by a certain parameter configuration and F_{best} represents the best result for an instance among all the parameter combinations. The average RPD (ARPD) values are depicted in Fig. 7.

$$RPD = \frac{F - F_{best}}{F_{best}} \cdot 100 \tag{14}$$

It can be seen that $popSize = 500$ and $tSize = 4$ are significantly better

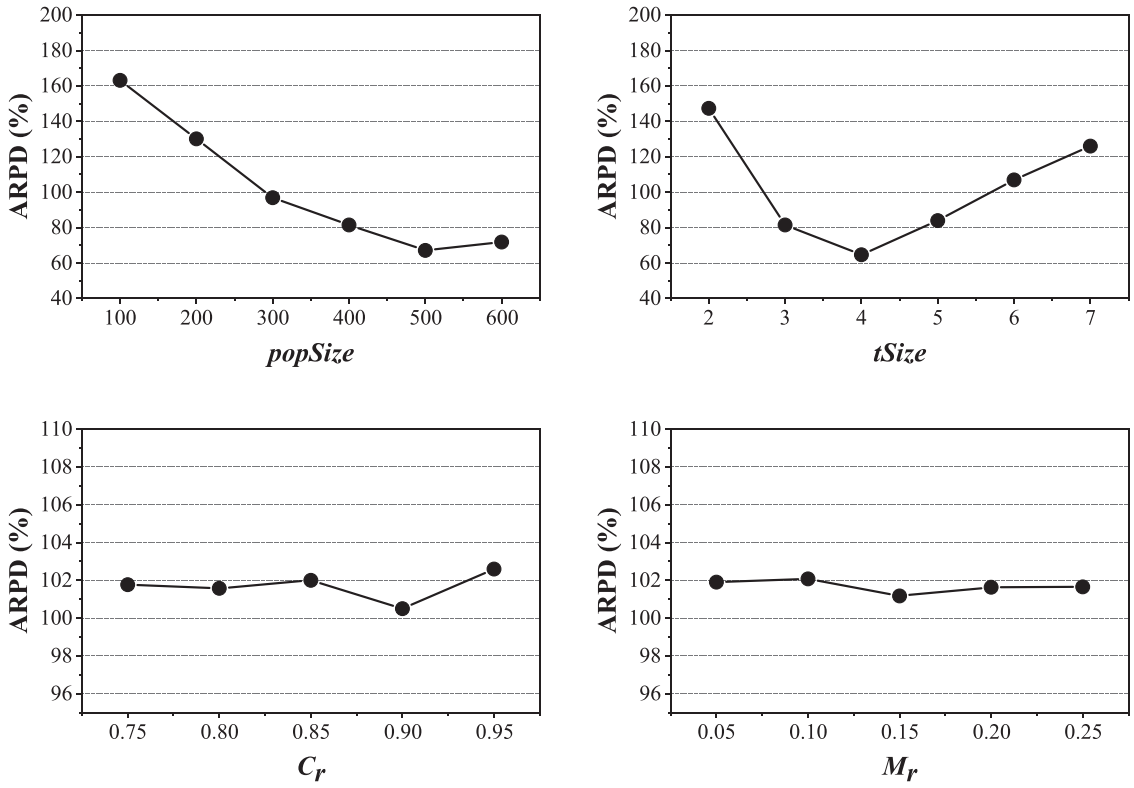

Fig. 7. Main effects plot of significant parameters for the proposed IGA.

Table 2
Experimental results and comparisons for Group A.

Instance	n×m×l	MILP			GA		GA-VNS		IGA		Significant? ($p \leq 0.05$)	
		best	Gap (%)	CPU time	best	avg	best	avg	best	avg	IGA vs GA	IGA vs GA-VNS
MFJS01-MR-A	5×6×2	**453**	0.00	7.33	**453**	533.60	**453**	456.30	**453**	453.00	Yes	No
MFJS02-MR-A	5×7×2	**495**	0.00	121.70	**495**	610.05	**495**	545.45	**495**	519.10	Yes	No
MFJS03-MR-A	6×7×2	**839**	0.00	93.59	**839**	1237.20	**839**	1016.05	**839**	987.65	Yes	No
MFJS04-MR-A	7×7×2	**1355**	68.85	3601.84	1591	1971.10	1387	1545.10	1366	1586.85	Yes	No
MFJS05-MR-A	7×7×3	**2343**	45.23	3600.73	2351	2731.30	2351	2568.50	**2343**	2513.90	Yes	No
MFJS06-MR-A	8×7×2	**1392**	68.38	3600.80	1438	2022.30	**1392**	1645.60	**1392**	1715.60	Yes	No
MFJS07-MR-A	8×7×3	**2643**	100.00	3601.39	3102	3964.75	2900	3212.60	2872	3210.15	Yes	No
MFJS08-MR-A	9×8×3	4628	100.00	3600.76	**4456**	5821.80	4644	5625.15	4636	5186.90	Yes	Yes
MFJS09-MR-A	11×8×4	8468	100.00	3603.03	6343	8013.90	5458	6760.00	**4732**	6889.80	Yes	No
MFJS10-MR-A	12×8×4	11,459	100.00	3601.03	8936	11,029.00	7834	9565.65	**7693**	9244.85	Yes	No
Mk01-MR-A	10×6×2	310	100.00	3601.78	316	383.15	**278**	307.75	**278**	307.85	Yes	No
Mk02-MR-A	10×6×3	613	100.00	3601.89	352	396.85	**245**	302.15	250	299.70	Yes	No
Mk03-MR-A	15×8×2	–	–	–	3819	4070.70	4201	4633.90	**3222**	3532.80	Yes	Yes
Mk04-MR-A	15×8×3	1461	100.00	3601.77	1069	1246.60	957	1159.00	**934**	1046.35	Yes	Yes
Mk05-MR-A	15×4×2	–	–	–	3646	4501.15	2894	3447.55	**2739**	3055.55	Yes	Yes
Mk06-MR-A	10×15×4	–	–	–	2010	2222.80	1372	1809.75	**1312**	1483.00	Yes	Yes
Mk07-MR-A	20×5×2	–	–	–	3156	3563.60	3001	3572.35	**2164**	2523.20	Yes	Yes
Mk08-MR-A	20×10×3	–	–	–	19,287	21,351.70	20,297	22,678.65	**19,102**	21,087.35	No	Yes
Mk09-MR-A	20×10×4	–	–	–	23,790	29,655.55	24,772	27,835.65	**17,412**	22,753.00	Yes	Yes
Mk10-MR-A	20×15×5	–	–	–	19,592	21,351.50	14,177	15,970.65	**9829**	12,007.75	Yes	Yes
#better		12			16		14		–		19	9
#even		5			3		5		–		1	11
#worse		3			1		1		–		0	0

Table 3
Experimental results and comparisons for Group B.

Instance	n × m × l	MILP			GA		GA-VNS		IGA		Significant? ($p \leq 0.05$)	
		best	Gap (%)	CPU time	best	avg	best	avg	best	avg	IGA vs GA	IGA vs GA-VNS
MFJS01-MR-B	5×6×2	**1013**	0.00	31.27	**1013**	1035.65	**1013**	1013.00	**1013**	1013.00	No	No
MFJS02-MR-B	5×7×2	**893**	0.00	219.31	912	988.90	901	960.55	**893**	943.20	Yes	No
MFJS03-MR-B	6×7×2	**1552**	0.00	655.09	1730	1924.15	1730	1823.45	**1552**	1722.45	Yes	Yes
MFJS04-MR-B	7×7×2	**2431**	65.50	3601.03	2687	3217.90	2449	2780.80	2471	2592.05	Yes	Yes
MFJS05-MR-B	7×7×3	**3333**	47.60	3600.91	**3333**	3949.00	**3333**	3405.20	**3333**	3555.85	Yes	No
MFJS06-MR-B	8×7×2	**2207**	59.72	3601.16	2397	2883.25	**2207**	2552.55	**2207**	2544.80	Yes	No
MFJS07-MR-B	8×7×3	4668	92.05	3601.06	4358	5302.05	4101	4403.20	**3994**	4312.50	Yes	No
MFJS08-MR-B	9×8×3	7234	93.65	3600.70	6790	8425.85	6657	7467.95	**6591**	6902.70	Yes	Yes
MFJS09-MR-B	11×8×4	10,752	99.24	3604.27	8575	10,550.45	8253	8976.05	**7277**	8781.40	Yes	No
MFJS10-MR-B	12×8×4	16,467	97.42	3600.91	11,975	13,574.30	11,105	12,114.80	**10,311**	11,872.10	Yes	No
Mk01-MR-B	10×6×2	462	99.52	3601.78	406	473.20	365	398.70	**357**	391.70	Yes	No
Mk02-MR-B	10×6×3	686	98.54	3602.13	375	462.60	337	381.75	**295**	346.90	Yes	Yes
Mk03-MR-B	15×8×2	–	–	–	3966	4429.35	4764	4918.70	**3725**	3886.65	Yes	Yes
Mk04-MR-B	15×8×3	1442	99.58	3601.59	1284	1416.90	1325	1445.30	**1190**	1288.90	Yes	Yes
Mk05-MR-B	15×4×2	–	–	–	3987	4733.10	3304	3811.65	**3159**	3555.55	Yes	Yes
Mk06-MR-B	10×15×4	–	–	–	2119	2401.25	1724	2111.90	**1466**	1682.35	Yes	Yes
Mk07-MR-B	20×5×2	–	–	–	3386	3988.00	3479	3841.80	**2664**	2942.25	Yes	Yes
Mk08-MR-B	20×10×3	–	–	–	20,017	22,295.55	20,637	23,405.50	**19,958**	21,876.10	No	Yes
Mk09-MR-B	20×10×4	–	–	–	28,714	32,094.30	27,440	28,620.00	**21,783**	24,626.65	Yes	Yes
Mk10-MR-B	20×15×5	–	–	–	19,687	22,000.15	15,403	17,016.25	**11,409**	13,107.00	Yes	Yes
#better		14			18		16		–		18	12
#even		5			2		3		–		2	8
#worse		1			0		1		–		0	0

Table 4
Experimental results and comparisons for Group C.

Instance	n×m×l	MILP			GA		GA-VNS		IGA		Significant? ($p \leq 0.05$)	
		best	Gap (%)	CPU time	best	avg	best	avg	best	avg	IGA vs GA	IGA vs GA-VNS
MFJS01-MR-C	5×6×2	**440**	0.00	620.38	583	765.15	**440**	630.30	**440**	570.05	Yes	No
MFJS02-MR-C	5×7×2	**90**	0.00	17.64	132	278.65	115	238.40	**90**	187.70	Yes	Yes
MFJS03-MR-C	6×7×2	**632**	100.00	3601.09	1028	1318.05	**632**	1006.00	**632**	1007.50	Yes	No
MFJS04-MR-C	7×7×2	**899**	77.75	3601.13	1181	2223.85	**899**	1458.05	**899**	1324.55	Yes	No
MFJS05-MR-C	7×7×3	1039	100.00	3601.14	1118	1527.85	1226	1468.65	**956**	1205.95	Yes	Yes
MFJS06-MR-C	8×7×2	1358	100.00	3601.34	1557	2149.10	1169	1719.80	**1085**	1532.60	Yes	No
MFJS07-MR-C	8×7×3	2077	100.00	3602.53	3249	4406.00	2311	2919.95	**2075**	2913.15	Yes	No
MFJS08-MR-C	9×8×3	3321	100.00	3602.33	3791	4581.35	3117	3708.20	**2980**	3671.70	Yes	No
MFJS09-MR-C	11×8×4	5195	100.00	3602.22	4666	5444.35	3277	4067.00	**3139**	4007.00	Yes	No
MFJS10-MR-C	12×8×4	10,190	100.00	3602.56	8185	9711.00	5892	7664.90	**5150**	6919.25	Yes	Yes
Mk01-MR-C	10×6×2	220	100.00	3601.70	236	284.55	128	178.45	**104**	172.25	Yes	No
Mk02-MR-C	10×6×3	524	100.00	3604.06	354	414.25	**192**	267.25	**192**	229.10	Yes	Yes
Mk03-MR-C	15×8×2	–	–	–	4340	4839.75	5166	5464.10	**3496**	4006.20	Yes	Yes
Mk04-MR-C	15×8×3	1209	100.00	3604.23	901	1059.35	633	739.70	**590**	697.20	Yes	No
Mk05-MR-C	15×4×2	–	–	–	2510	2825.15	1911	2071.40	**1802**	1980.65	Yes	Yes
Mk06-MR-C	10×15×4	–	–	–	1749	1974.00	1082	1295.65	**1055**	1154.30	Yes	Yes
Mk07-MR-C	20×5×2	–	–	–	4640	5093.50	2239	3242.60	**2145**	2419.00	Yes	Yes
Mk08-MR-C	20×10×3	–	–	–	18,479	21,716.85	13,133	14,863.55	**10,556**	13,046.30	Yes	Yes
Mk09-MR-C	20×10×4	–	–	–	18,994	21,541.75	12,012	15,204.45	**9447**	10,778.75	Yes	Yes
Mk10-MR-C	20×15×5	–	–	–	21855	23317.10	12563	15030.70	**9308**	11282.40	Yes	Yes
#better		16			20		16		–		20	11
#even		4			0		4		–		0	9
#worse		0			0		0		–		0	0

than other parameter levels, thus they are adopted for the IGA. Since there are not discernible differences on APRD values for C_r and M_r according to Fig. 7, C_r and M_r are set to 0.90 and 0.15 respectively because they provide the minimum ARPDs. In addition, the size of elite individuals is set to 5. The proposed IGA adopts this parameter configuration for all the numberical experiments.

5.2. Experimental results and analyses

The proposed IGA was tested on the FJSP-MR instances generated in Section 5.1.1, and was compared with the proposed MILP model, a basic GA, and a GA with VNS (GA-VNS). The basic GA, denoted by GA in statistical tables, is a simplified version of the IGA, where the ILS component stays inactive. This control group is set to reveal the improvements brought by the proposed ILS. In addition, a GA-VNS was modified from [45], which was designed for a DRC-FJSP with learning effects, and was taken for the comparison. The encoding scheme of the original GA-VNS is very similar to that of this paper, and three operators are successively employed to manipulate chromosomes at local search stages. The GA-VNS is a typical algorithm for scheduling problems, which tries to improve solutions by changing encoding vectors and is usually hindered by repetitive individuals and invalid operations. While

the disjunctive graph representation based ILS is just developed to overcome many-to-one situations and to advance the targeted search. Hence, the GA-VNS was duplicated with some adjustments to solve the FJSP-MR, where the main workflow and parameter setting are the same as those of the proposed algorithm. In other words, the only difference between the IGA and GA-VNS is the neighborhood structure, which helps to investigate the performance of the modified *k*-insertion local search.

All the algorithms were coded in C+ + on the development environment Visual Studio 2019, and were run on a computer with Intel(R) Core i9–10885 H CPU 2.40 GHz and 16.0 GB RAM. While the MILP model was implemented by calling the commercial solver ILOG CPLEX 12.8 on the same platform. Twenty independent runs were performed on all the instances with a termination criterion $n \cdot m \cdot l$ seconds, where the best values were recorded. Table 2Table 34 present experimental results and comparisons, where the best TWT for each instance is highlighted in bold and a "-" indicates data unavailable. Statistics on best TWT values obtained by the proposed IGA compared to other approaches are given at bottoms of tables.

As shown in Tables 2–4, the proposed MILP model can optimally solve the first three instances for both group A and group B, and finds optimal solutions for the first two instances of group C. For all the FJSP-MR instances extended from Mk03 and Mk05-Mk10, the MILP cannot find feasible solutions within 3600 s due to the scale issue. Sometimes, numbers of binary variables and constraints even exceed the capacity of CPLEX solver, which implies the complexity of problems is beyond the boundary of the proposed MILP model. Besides, the MILP provides high-quality results on most of the remaining problems, but the computational cost is considerable. Generally, The proposed MILP model is highly efficient for small-scale FJSP-MR instances. As a mathematical programming-based exact method, the performance of MILP evidently declines when the problem scale increases, thus the MILP model cannot be directly applied to solve real-world FJSP-MRs in most cases.

The basic GA can obtain feasible solutions for all the instances because of the problem-specific semi-active decoding. For small-scale problems, the GA can find optimum within a short time, and even provides a better solution than any other methods on MFJS08-MR-A. However, for the majority of instances, the GA is greatly inferior to the GA-VNS and IGA due to the absence of local search strategies, which reveals that a local search is necessary for the basic GA.

The GA-VNS contains a local search component directly acting on encoding vectors, thus it shows great advantages over the basic GA. In comparison to the proposed IGA, the GA-VNS obtains better results on Mk02-MR-A and MFJS04-MR-B, and provides the same best TWT values on 12 instances. For the rest of problems, the proposed IGA is obviously better than the GA-VNS in respect of best values and averages. It can be inferred that introducing the encoding vector based local search brings great improvements to the GA, so the GA-VNS is very effective on simple instances. However, when the scale and AM flexibility of a problem grow, the performance of GA-VNS degrades due to limitations of the encoding scheme. Local search operators for the GA-VNS are randomly performed on encoding vectors, which cannot ensure that bottlenecks of a schedule are destroyed. In addition, simply re-assigning an element on an OS vector will not necessarily yields a different solution especially for large-scale problems. The two factors make the GA-VNS less efficient on difficult instances compared with the proposed IGA.

To investigate whether there are statistical differences between the solution methods, Wilcoxon signed-rank tests were performed on paired RPD observations of comparative algorithms at the significance level 0.05. The null hypothesis for each significance test is defined as: H_0:

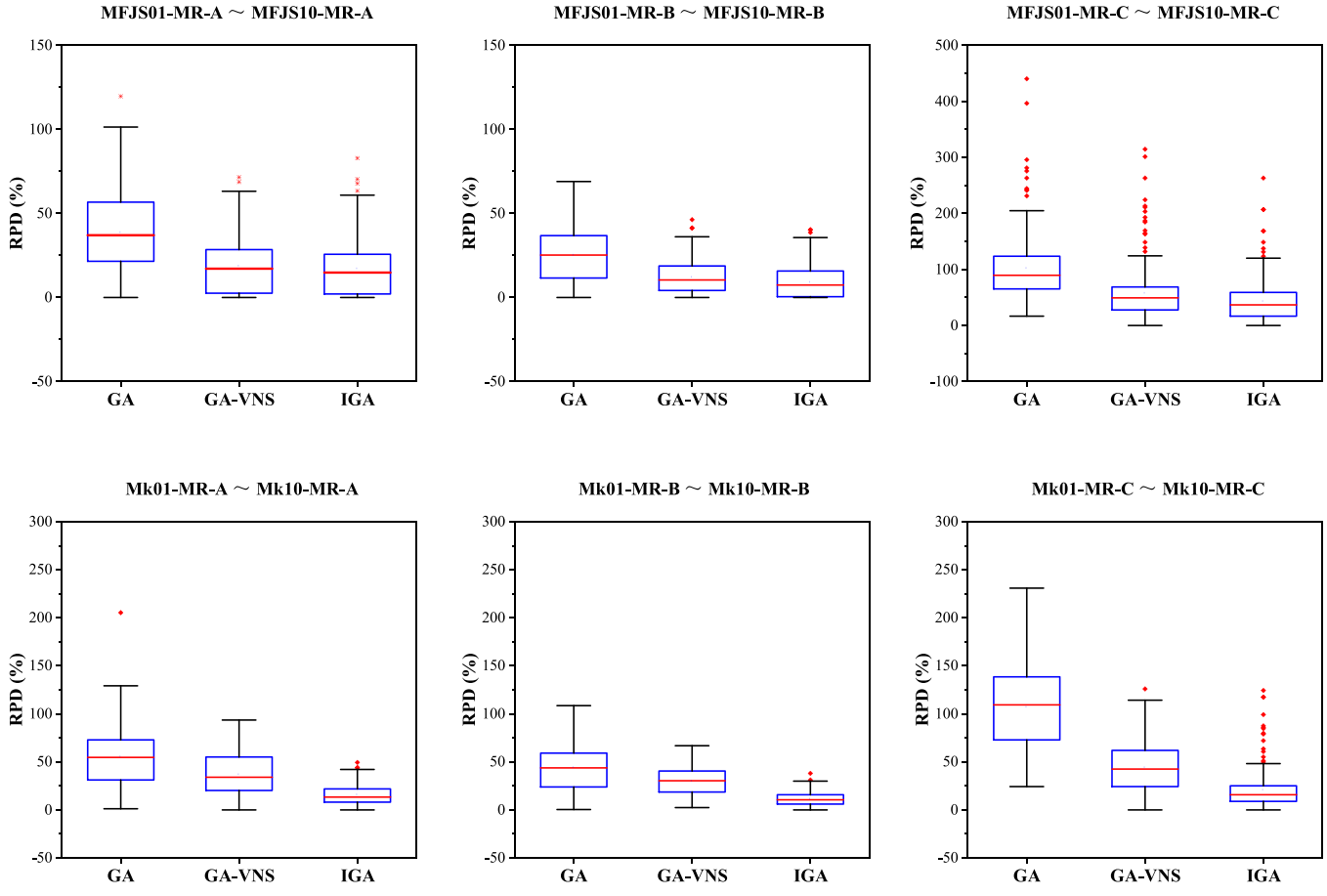

Fig. 8. Box plots for RPD values of comparative algortihms on the FJSP-MR instances.

$D_{RPD} = 0$, which means there is no difference between RPD values. Correspondingly, the alternative hypothesis is: H_1: $D_{RPD} \ddagger 0$, indicating there is a significant difference between RPD values obtained by the compared algortihms. Results of Wilcoxon signed-rank tests are attached to the ends of Tables 2–4.

Results in Tables 2–4 suggest that the proposed IGA is either significantly better, or shown no statistical differences compared to the GA and GA-VNS. It is predictable that the basic GA without any local search is evidently worse than the others. The only difference of the GA-VNS and proposed IGA is the local search strategy. Since the IGA is significantly better than the GA-VNS on 32 instances out of 60 instances, it can be inferred that the disjunctive graph based ILS component is more effective than common operators performing directly on encoding vectors, and leads to better overall performance.

Fig. 8 presents box plots for the basic GA, GA-VNS, and IGA, where the FJSP-MR instances are further divided into six groups because the difficulty for source benchmark sets is different. The first row of Fig. 8 shows RPD values for comparative algortihms on the instances extended from Fdata [14]], problem scales of which are smaller than those of BRdata instances [7]]. The proposed IGA achieves almost the same performance as the GA-VNS on simple problems, while averages and medians provided by the IGA are slightly better. For box plots in the second row, the IGA shows distinct advantages over the GA-VNS in both optimality and stability. Instances of group C are evidently harder than the others because all the algorithms tend to be instable due to the AM flexibility. In this case, the proposed IGA can still obtain high-quality results and shows comparatively little fluctuation, where the modified neighborhood structure is the most important component.

Based on obtained average RPD values, Friedman tests were carried out to investigate whether there are statistical differences for comparative algorithms on groups of test instances at the significance level 0.05. Results of Friedman tests are presented in Table5. It can be seen from the table that there are statistical differences on three comparative algorithms for all test groups. With the growth of computational complexity, average ranks and p-values of the proposed IGA are both getting smaller, which suggests advantages of the IGA is even more evident for difficult problems compared to other approaches.

In summary, the proposed MILP model can optimally solve small-scale FJSP-MR instances, and find good solutions for medium-sized problems with higher costs. If the complexity of an instance approaches the capacity boundary of the MILP model, the GAs with problem-focused encoding and decoding strategies and operators can be applied to efficiently solve the problem. Computational results and significant tests suggest that the proposed ILS based on the disjunctive graph representation is highly effective because it can well address many-to-one situations caused by the encoding scheme and directly act on the bottlenecks. Hence, the proposed IGA shows better performance over other methods, and can be considered as a reasonable solution method for the FJSP-MR.

Table 5
Friedman tests for the performance of comparative algorithms on three groups of FJSP-MR test instances.

Instance	Algorithm	Average Rank	p-value	Significant? ($p \leq 0.05$)
	GA	2.850		
Group A	GA-VNS	1.950	1.18E-06	Yes
	IGA	1.200		
	GA	2.850		
Group B	GA-VNS	2.075	1.08E-07	Yes
	IGA	1.075		
	GA	2.950		
Group C	GA-VNS	2.000	1.45E-08	Yes
	IGA	1.050		

6. Industrial case study

This section investigates a real-world scheduling problem with RMTs and limited AMs to validate the proposed methods. Firstly, the production scenario and needs for the enterprise are described in detail. Secondly, the scarcity of AMs is analyzed, and the IGA is modified with some targeted adjustments to the local search. Finally, the proposed IGA is tested on two instances abstracted from the shop floor and is compared with the basic GA and GA-VNS.

6.1. Background and requirements

This case comes from a machining workshop for processing large-sized structures, where RMTs can be dynamically reconfigured by installing a series of AMs to process different operations. Additionally, at some manual workstations of the shop floor, workers need selected tools to finish the jobs. This situation is also considered as reconfiguring RMTs with AMs, thus the scheduling problem can be formulated as an FJSP-MR.

An order to be delivered within five months is adopted to generate test instances, where jobs with four priority levels have different release dates and due dates. Totally, this instance contains 34 jobs, 47 machines, and 19 AMs, and the scale is far more than those of previous benchmark instances. The shift for a workday is flexible and can be adjusted according to the production progress, because only working with the normal shift is usually insufficient to finish orders on time. However, the shift management is not the main focus of this paper. To eliminate this decision-making process, a normal shift (average 8 working hours per day) and an alternative shift (average 10 working hours per day) are simply taken for the order. Hence, a $34 \times 47 \times 19$ FJSP-MR instance is generated and will be tested with two different production capacity levels.

Since the assembly and disassembly for AMs are time-consuming, the enterprise expects to minimize the TWT, reconfiguration time, and transfers for scarce AMs simultaneously. On the basis of preliminary experiments, there are not obvious conflicts between the objectives, which means TWT values can be reduced without compromising the requirements for reconfigurations. Therefore it is inappropriate to formulate the FJSP-MR as a multi-objective optimization problem. The real-world case is addressed with the only objective TWT as Section 3, where the reconfiguration time and transfers for scarce AMs (simply denoted by AM transfers) are given for reference.

6.2. Discussions for the AM availability

Matching relations for AMs and available RMTs are given in supplementary files. Actually, not all the AMs are in short supply for corresponding RMTs. For example, A_{19} can only be attached to M_{18}, which suggests that M_{18} can be reconfigured at any time. Besides, although $M_{37} \sim M_{44}$ share only 1 AM A_{12} for drilling operations, this kind of work is not common in the practical production of the shop floor, thereby A_{12} is not regarded as the primary obstacle for scheduling either. In other words, only a portion of AMs (i.e. A_1, A_2, A_{10}, A_{11}, and $A_{13} \sim A_{17}$) are the bottlenecks for the case study.

The proposed ILS spends much time optimizing the AM selection. However, excessive attempts for less busy AMs is not cost-efficient. To further improve the searching efficiency, the ILS only re-allocates the relatively scarce AMs in consideration of the situation discussed above. If two consecutive operations on an RMT can be processed with a group of non-bottleneck AMs (e.g. $A_5 \sim A_7$), the ILS will adjust the AM selection to ensure that only one available AM is attached in correspondence with enterprise requirements.

6.3. Experimental analyses

The basic GA, GA-VNS, and proposed IGA were tested twenty times

Table 6
Test results for comparative algorithms on the real-world case with different shifts.

Shift	Algorithm	TWT		Reconfiguration time		AM transfers	
		best	avg	best	avg	best	avg
Normal	GA	3256	4038.15	122	136.70	37	42.15
	GA-VNS	2750	3653.15	117	133.20	37	41.95
	IGA	**2023**	2297.80	124	135.20	36	40.70
Alternative	GA	397	643.15	122	145.70	39	45.10
	GA-VNS	313	494.25	135	145.35	42	45.50
	IGA	**110**	209.10	130	143.20	42	45.10

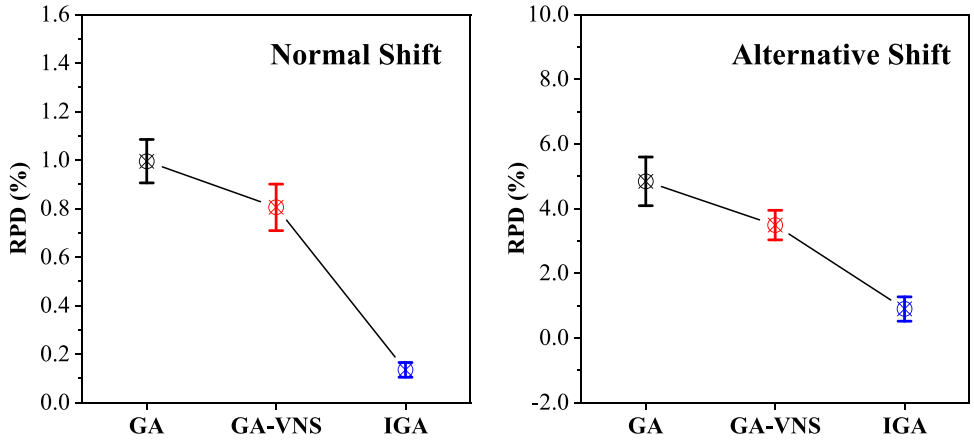

Fig. 9. Interval plots for comparative algorithms with 95% CI for the mean.

with a stopping criterion 10 min on the real-world case. Computational results are presented in Table 6, where the reconfiguration times and AM transfers are also given to meet the requirement. Interval plot for the GAs is shown in Fig. 9 to compare the overall performance.

It can be seen from Table 6 and Fig. 9 that the reconfiguration time and AM transfers are very close for all the algorithms, which suggests the TWT objective can be optimized without increasing other indicators. Experimental results for reconfiguration time and AM transfers are also consistent with the initial decision for not formulating the real-world case as a multi-objective problem. For the normal shift, tardy jobs are evidently more than those of alternative shift due to the shortage of working hours. In this difficult situation, the IGA significantly outperforms the other algorithms in both optimality and stability on TWT values. When the production capacity grows, the proposed IGA is able to find a solution with only a few tary jobs. For the case study, the performance of the basic GA and GA-VNS is comparatively close, which is far different from results of numerical experiments. It can be inferred that there are some jobs with considerable operations for the real-world problem, which makes the many-to-one situation even worse because re-assigning an element for such jobs is unlikely to generate a new solution. The proposed ILS based on disjunctive graphs can well address the issue and ensure the local search is performed on bottleneck AMs, thus the superiority is even more evident than that on benchmark instances.

To further investigate the real-time performance of IGA, convergence

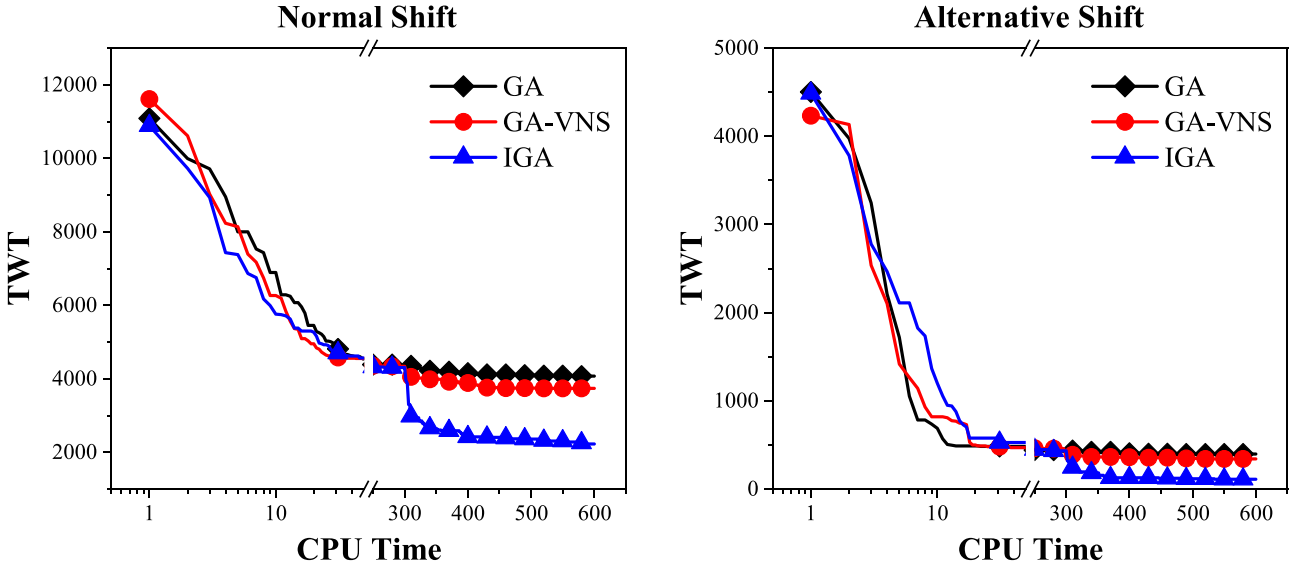

Fig. 10. Convergence curves of comparative algorithms for normal and alternative shifts of the real-world FJSP-MR case.

curves of comparative algorithms for normal and alternative shifts are depicted in Fig. 10. It can be seen that all the algorithm converge very fast at the beginning, and the curves show little fluctuation after approximately 100 s. Since local search strategies are activated, GA-VNS and IGA can immediately find some better solutions while the GA tends comparatively weak. Improvements brought by the proposed ILS is significantly greater compared with the VNS, which reveals the superiority of disjunctive graph model and modified *k*-insertion over common local search operators.

In general, based on test results and analyses, the IGA has been demonstrated to be capable of addressing FJSP-MRs with real-world scales. The ILS is very effective for the AM selection sub-problem and helps to improve the stability. It is also important to note that the IGA is potential to achieve higher efficiency after some reasonable adjustments by considering practical circumstances, which is of great significance to apply the proposed approach to solve complicated engineering problems.

7. Conclusions

This paper investigates an FJSP considering RMTs with limited AMs for minimizing the TWT. Firstly, an MILP model is established for the FJSP-MR to describe production constraints. Secondly, an IGA is developed with an ILS component based the disjunctive graph model, where bottlenecks for an FJSP-MR schedule are thoroughly analyzed and are applied to improve the neighborhood structure. Thirdly, the proposed MILP and IGA are tested on benchmark instances extended from existing test sets, and are compared with some typical algortithms reported in the recent literature. Finally, a real-world FJSP-MR case is studied to further prove the IGA is able to solve practical FJSP-MRs with some minor changes.

To sum up, the MILP only applies to small-scale instances, while the IGA is highly effective in more complicated FJSP-MRs. The disjunctive graph representation, which establishes one-to-one correspondence between graphs and schedules, is a reasonable complementary component to the encoding strategy. Besides, the neighboring solution generation based on critical paths for TWT and modified *k*-insertion also contributes to the performance of ILS. For real-world FJSP-MRs, this paper provides an idea to simplify the neighborhood structure in consideration of the availability of AMs, making the obtained schedules in line with enterprises' demands.

The proposed method adopts the most classical meta-heuristic GA as the basis, and devotes more efforts to the local search phase. Some advanced optimization algorithms and frameworks can be potentially combined with the ILS to solve the FJSP-MR more efficiently. Additionally, the lower bound approximate calculation, TWT objective estimation, and other TWT based neighborhood structures, which helps to screen out inferior solutions and improve the searching efficiency, are also worth research.

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.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grants no. 51905199 and no. U20A6004, and in part by the National Science Fund for Distinguished Young Scholars of China under Grant no. 51825502.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.jmsy.2022.01.014.

References

[1] Amjad MK, Butt SI, Kousar R, Ahmad R, Agha MH, Faping Z, et al. Recent research trends in genetic algorithm based flexible job shop scheduling problems. Math Probl Eng 2018;2018:1–32.
[2] Andrade-Pineda JL, Canca D, Gonzalez-R PL, Calle M. Scheduling a dual-resource flexible job shop with makespan and due date-related criteria. Ann Oper Res 2020;291(1):5–35.
[3] Azab A, Naderi B. Modelling the problem of production scheduling for reconfigurable manufacturing systems. Procedia CIRP 2015;33:76–80.
[4] Bensmaine A, Dahane M, Benyoucef L. A new heuristic for integrated process planning and scheduling in reconfigurable manufacturing systems. Int J Prod Res 2014;52(12):3583–94.
[5] Bortolini M, Ferrari E, Galizia FG, Regattieri A. An optimisation model for the dynamic management of cellular reconfigurable manufacturing systems under auxiliary module availability constraints. J Manuf Syst 2021;58:442–51.
[6] Bortolini M, Galizia FG, Mora C. Reconfigurable manufacturing systems: literature review and research trend. J Manuf Syst 2018;49:93–106.
[7] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 1993;41(3):157–83.
[8] Brucker P, Schlie R. Job-shop scheduling with multi-purpose machines. Computing 1990;45(4):369–75.
[9] Bülbül K. A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem. Comput Oper Res 2011;38(6):967–83.
[10] Dhiflaoui M, Nouri HE, Driss OB. Dual-resource constraints in classical and flexible job shop problems: a state-of-the-art review. Procedia Comput Sci 2018;126:1507–15.
[11] Doh H-H, Yu J-M, Kim J-S, Lee D-H, Nam S-H. A priority scheduling approach for flexible job shops with multiple process plans. Int J Prod Res 2013;51(12):3748–64.
[12] Dou J, Li J, Su C. Bi-objective optimization of integrating configuration generation and scheduling for reconfigurable flow lines using nsga-ii. Int J Adv Manuf Technol 2016;86(5):1945–62.
[13] Fan J, Shen W, Gao L, Zhang C, Zhang Z. A hybrid jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths. J Manuf Syst 2021;60:298–311.
[14] Fattahi P, Mehrabad MS, Jolai F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J Intell Manuf 2007;18(3):331–42.
[15] Gadalla M, Xue D. Recent advances in research on reconfigurable machine tools: a literature review. Int J Prod Res 2017;55(5):1440–54.
[16] Gao K, Cao Z, Zhang L, Chen Z, Han Y, Pan Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA J Autom Sin 2019;6(4):904–16.
[17] Gao K, Yang F, Zhou M, Pan Q, Suganthan PN. Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm. IEEE Trans Cybern 2019;49(5):1944–55.
[18] Gao L, Pan QK. A shuffled multi-swarm micro-migrating birds optimizer for a multi-resource-constrained flexible job shop scheduling problem. Inf Sci 2016;372:655–76.
[19] Gargeya VB, Deane RH. Scheduling research in multiple resource constrained job shops: a review and critique. Int J Prod Res 1996;34(8):2077–97.
[20] Ghaleb M, Taghipour S, Zolfagharinia H. Real-time integrated production-scheduling and maintenance-planning in a flexible job shop with machine deterioration and condition-based maintenance. J Manuf Syst 2021;61:423–49.
[21] Holland JH. Genetic algorithms. Sci Am 1992;267(1):66–73.
[22] Koren Y, Heisel U, Jovane F, Moriwaki T, Pritschow G, Ulsoy G, et al. Reconfigurable manufacturing systems. CIRP Ann 1999;48(2):527–40.
[23] Koren Y, Shpitalni M. Design of reconfigurable manufacturing systems. J Manuf Syst 2010;29(4):130–41.
[24] Lei D, Guo X. Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. Int J Prod Res 2014;52(9):2519–29.
[25] Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 2016;174:93–110.
[26] Li X, Liu Y. Approach of solving dual resource constrained multi-objective flexible job shop scheduling problem based on moea/d. Int J Online Eng 2018;14(07):75.
[27] Liu Q, Li X, Gao L. Mathematical modeling and a hybrid evolutionary algorithm for process planning. J Intell Manuf 2021;32(3):781–97.
[28] Liu Q, Li X, Gao L. A novel milp model based on the topology of a network graph for process planning in an intelligent manufacturing system. Engineering 2021;7(6):807–17.
[29] Liu Q, Li X, Gao L, Wang G. Mathematical model and discrete artificial bee colony algorithm for distributed integrated process planning and scheduling. J Manuf Syst 2021;61:300–10.
[30] Lu C, Zhang B, Gao L, Yi J, Mou J. A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds. IEEE Syst J 2021.
[31] Mahmoodjanloo M, Tavakkoli-Moghaddam R, Baboli A, Bozorgi-Amiri A. Flexible job shop scheduling problem with reconfigurable machine tools: an improved differential evolution algorithm. Appl Soft Comput 2020;94:106416.
[32] Mastrolilli M, Gambardella LM. Effective neighbourhood functions for the flexible job shop problem. J Sched 2000;3(1):3–20.
[33] Mehrabi MG, Ulsoy AG, Koren Y, Heytler P. Trends and perspectives in flexible and reconfigurable manufacturing systems. J Intell Manuf 2002;13(2):135–46.

[34] Meng L, Zhang C, Ren Y, Zhang B, Lv C. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Comput Ind Eng 2020;142:106347.
[35] Mousakhani M. Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. Int J Prod Res 2013;51(12):3476–87.
[36] Müller D, Kress D. Filter-and-fan approaches for scheduling flexible job shops under workforce constraints. Int J Prod Res 2021;10:1–23.
[37] Özgüven C, Özbakír L, Yavuz Y. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 2010;34(6):1539–48.
[38] Pinedo, M. (2012). Scheduling. Springer.
[39] Pinedo M, Singer M. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Nav Res Logist 1999;46(1):1–17.
[40] Shen L, Dauzère-Pérès S, Neufeld JS. Solving the flexible job shop scheduling problem with sequence-dependent setup times. Eur J Oper Res 2018;265(2):503–16.
[41] Singer M, Pinedo M. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. IIE Trans 1998;30(2):109–18.
[42] Sobeyko O, Mönch L. Heuristic approaches for scheduling jobs in large-scale flexible job shops. Comput Oper Res 2016;68:97–109.
[43] Wang S, Wang L, Liu M, Xu Y. A hybrid estimation of distribution algorithm for the semiconductor final testing scheduling problem. J Intell Manuf 2015;26(5):861–71.
[44] Wu J-Z, Chien C-F. Modeling semiconductor testing job scheduling and dynamic testing machine configuration. Expert Syst Appl 2008;35(1–2):485–96.
[45] Wu R, Li Y, Guo S, Xu W. Solving the dual-resource constrained flexible job shop scheduling problem with learning effect by a hybrid genetic algorithm. Adv Mech Eng 2018.
[46] Xie J, Gao L, Peng K, Li X, Li H. Review on flexible job shop scheduling. IET Collab Intell Manuf 2019;1(3):67–77.
[47] Yelles-Chaouche AR, Gurevsky E, Brahimi N, Dolgui A. Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. Int J Prod Res 2021;59(21):6400–18.
[48] Zhang R, Wu C. A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. Comput Oper Res 2017;38(5):854–67.
[49] Zheng XL, Wang L, Wang SY. A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem. Knowl-Based Syst 2014;57:95–103.