Journal of Manufacturing Systems 73 (2024) 143–158

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Technical paper

Machine-fixture-pallet resources constrained flexible job shop scheduling considering loading and unloading times under pallet automation system

Yulu Zhou, Shichang Du *, Molin Liu, Xiaoxiao Shen

Department of Industrial Engineering and Management, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China

ARTICLE INFO

Keywords:
Flexible job shop scheduling
Machine-fixture-pallet resources
Loading and unloading times
Pallet automation system
Adaptive large neighborhood search algorithm

ABSTRACT

Pallet automation system (PAS) has gained more and more attention from many manufacturing enterprises with the development of flexible automation, where machine, fixture, and pallet are three critical resources. Few scholars study flexible job shop scheduling considering fixture and pallet resources. Meanwhile, they ignore the loading and unloading times, potentially leading to an extended makespan. Therefore, this paper presents a novel machine-fixture-pallet resources constrained flexible job shop scheduling problem considering loading and unloading times under a pallet automation system (FJSP-PAS). A mixed-integer programming model is established to minimize makespan. Considering the mutual constraints among resources, a new decoding method to choose resources for minimizing delay time (MDT) is proposed. An improved adaptive large neighborhood search algorithm (IALNS) with the simulated annealing method for local search is applied. The case study illustrates that IALNS with MDT rule for decoding (IALNS-MDT) can effectively reduce makespan while MDT enhances the quality of initial solutions. Furthermore, the importance of fixture-pallet resource allocation is emphasized by a given example, which inspires enterprises to make better resource allocation decisions.

1. Introduction

In the fourth industrial revolution wave, flexible job shop gradually played an essential role in manufacturing enterprises. Due to the high flexibility of products and machines, flexible automation has been widely applied in upgrading enterprises [1], which can decrease non-processing time. According to the research report of Li and Jiang [2], the pallet automation system (PAS) has become one of the critical strategies in flexible automation due to its small footprint and fast transfer speed.

PAS is typically employed in multi-variety, small-batch workshops to provide customized solutions for enterprises. In a PAS, as illustrated in Fig. 1, the workpiece is loaded onto the fixture-pallet at the setup station, and subsequently, the stacker crane transfers it to the storage rack or directly to the machine. Machines with the capability to process different operations are integrated, enabling the system to manufacture a diverse range of products. The compact arrangement of pallets and machines minimizes the footprint, thereby saving space. The high-speed moving and lifting stacker crane eliminates waiting time for machines or workers. The setup station establishes a convenient and safe working environment for workers. The modular design facilitates the system to expand easily and adjust the layout flexibly to meet production needs. Consequently, PAS can enhance production flexibility, reduce workers' operation time, minimize resource investments, and boost production efficiency. Furthermore, if PAS can address the challenge of loading and unloading, it will further advance toward unmanned production.

In real-world production, a PAS usually comprises various types of machines, fixtures, pallets, a setup station for loading and unloading, a stacker crane, a track, storage racks, and a central computer control system (see Fig. 1), which collectively meets the requirements for whole processing. Machines, fixtures, and pallets are three vital resources. A fixture fixed on a pallet forms a fixture-pallet where a workpiece is placed and clamped, so no fixture is installed on the machine fixedly. Fixture-pallet-workpiece can be moved by stacker crane to any machine's workstation because pallets are the same size as all workstations. All machines in the PAS can use the same fixture-pallet at different time because of their mobility, ensuring that the combination of fixtures and pallets remains unchanged. Hence, fixtures have no need to be loaded and unloaded from pallets, while workpieces do from fixture-pallet in some cases. That's because each operation may require to be clamped by distinct fixtures during processing and transportation, owing to variations in the size, shape, and processing mode of workpieces. Therefore, if

* Corresponding author.
E-mail address: lovbin@sjtu.edu.cn (S. Du).

https://doi.org/10.1016/j.jmsy.2024.01.010
Received 21 November 2023; Received in revised form 8 January 2024; Accepted 25 January 2024
Available online 6 February 2024

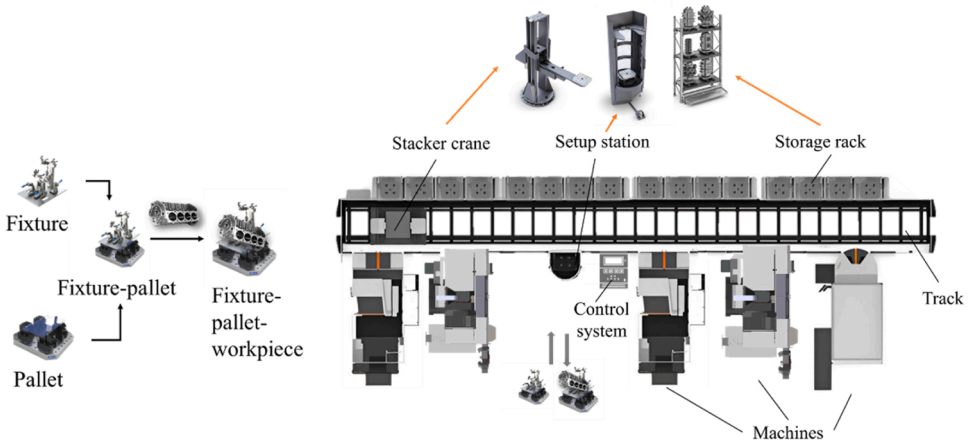


Fig. 1. The configuration of PAS.

Table 1
Summary of research work in FJSP with related resources and loading and unloading times.

Resource constrained FJSP	Non-Consider loading and unloading times	Consider loading and unloading times
Machine	Girish and Jawahar[3]; Wu and Wu[4]	Nouiri et al.[7]; Shen et al.[8]
Machine-fixture	Thörnblad et al.[14]	Chan et al.[11]; Wu et al.[12]; Xu et al.[13]
Machine-pallet	Mati et al.[15]	Liao et al.[16]
Machine-fixture-pallet	Yu et al.[9]; Lee et al.[10]	Will be studied in this paper

the next operation needs another fixture-pallet, or if the current fixture-pallet should be used for other workpieces, loading and unloading workpieces becomes necessary.

In a large manufacturing enterprise, loading or unloading a workpiece takes between 5 to 20 min depending on the size of the workpiece and the complexity of the fixture, significantly increasing non-processing time and potentially leading to an extended makespan. Obviously, the loading and unloading times heve an essential impact on scheduling solutions. Therefore, a reasonable scheduling strategy is needed to minimize makespan, considering the constraints of the machine, fixture, and pallet. However, there is limited research on this critical issue.

Most scholars studied flexible job shop scheduling problem (FJSP) primarily focused on machine resources [3–8], while few considered fixture and pallet resources. Meanwhile, they were not aware of the flexibility of fixtures and overlooked the loading and unloading times, treating them as constants included in the processing times[9,10]. To the authors' best knowledge, there is no research to model and solve machine-fixture-pallet resources constrained FJSP considering loading and unloading times.

Hence, this paper presents machine-fixture-pallet resources constrained flexible job shop scheduling problem considering loading and unloading times under pallet automation system (FJSP-PAS). The challenge in this problem lies in effectively coordinating machine and fixture-pallet resources, allocating the appropriate machine and fixture-pallet for each operation, and determining the operation sequencing of using machines and fixture-pallets. Furthermore, decision-makers often prioritize selecting machines that can complete processing at the earliest time when scheduling subsequent operations, with little consideration for the availability of fixtures. If the fixture-pallet required for the current workpiece is in use by another workpiece, loading and unloading are bound to happen. Alternatively, when consecutive operations of the same workpiece use the same fixture, there is no need for loading and unloading. However, this can create a bottleneck for the fixture, resulting in continuous waiting for other workpieces that need to be processed with this specific fixture. In both scenarios, no matter the loading and unloading times or waiting times could increase the makespan. To overcome these challenges, the proposed solution and main contributions of this paper are described as follows:

1. A mixed-integer programming model of FJSP-PAS is presented.
2. A new decoding method to choose resources for minimizing delay time (MDT) is proposed.
3. An improved adaptive large neighborhood search algorithm (IALNS) with MDT rule (IALNS-MDT) is developed.

The paper is organized as follows. In Section 2, the related literature in recent years is reviewed. In Section 3, the problem description and model's assumptions, constraints, and objectives are presented. In Section 4, an improved adaptive large neighborhood search algorithm is proposed and the details of each step are described. Section 5 reports the three different experiments. In Section 6, the summarization of this paper and future work will be introduced.

2. Literature review

Machine is one of the most critical resources in FJSP, but other resources like fixture and pallet still cannot be ignored. Some scholars' considerations of machine, fixture, pallet resources, and loading and unloading times are summarized in Table 1. While most scholars studied FJSP mainly focusing on machine resources [3–8], some did research on machine-fixture resources [11–14] and machine-pallet resources respectively [15,16]. Girish and Jawahar [3] and Wu and Wu [4] considered machine resources, but loading and unloading times is negligible. Nouiri et al. [7] and Shen et al.[8] considered both machine resources and sequence-dependent loading and unloading times. Thörnblad et al. [14] studied a problem in fixture availability. Chan et al. [11] considered an operation-dependent loading and unloading times for fixtures while solving FJSP. Wu et al. [12] evaluated the loading and unloading times of fixtures fixed on machines. Xu et al. [13] proposed a multi-resource FJSP with the time of replacing the fixture. Mati et al. [15] presented a practical FJSP with limited pallets, causing blocking constraints but did not take loading and unloading times into account. Liao et al. [16] studied an automatic pallet changer system where a machine has a certain number of pallets, and the workpieces can be set up while processing another.

Some other scholars considered the constraints of machine, fixture, and pallet resources simultaneously. Yu et al. [9] focused on the impact of fixture and pallet resources in a reconfigurable manufacturing system and proposed a priority rules method. A practical problem was proposed

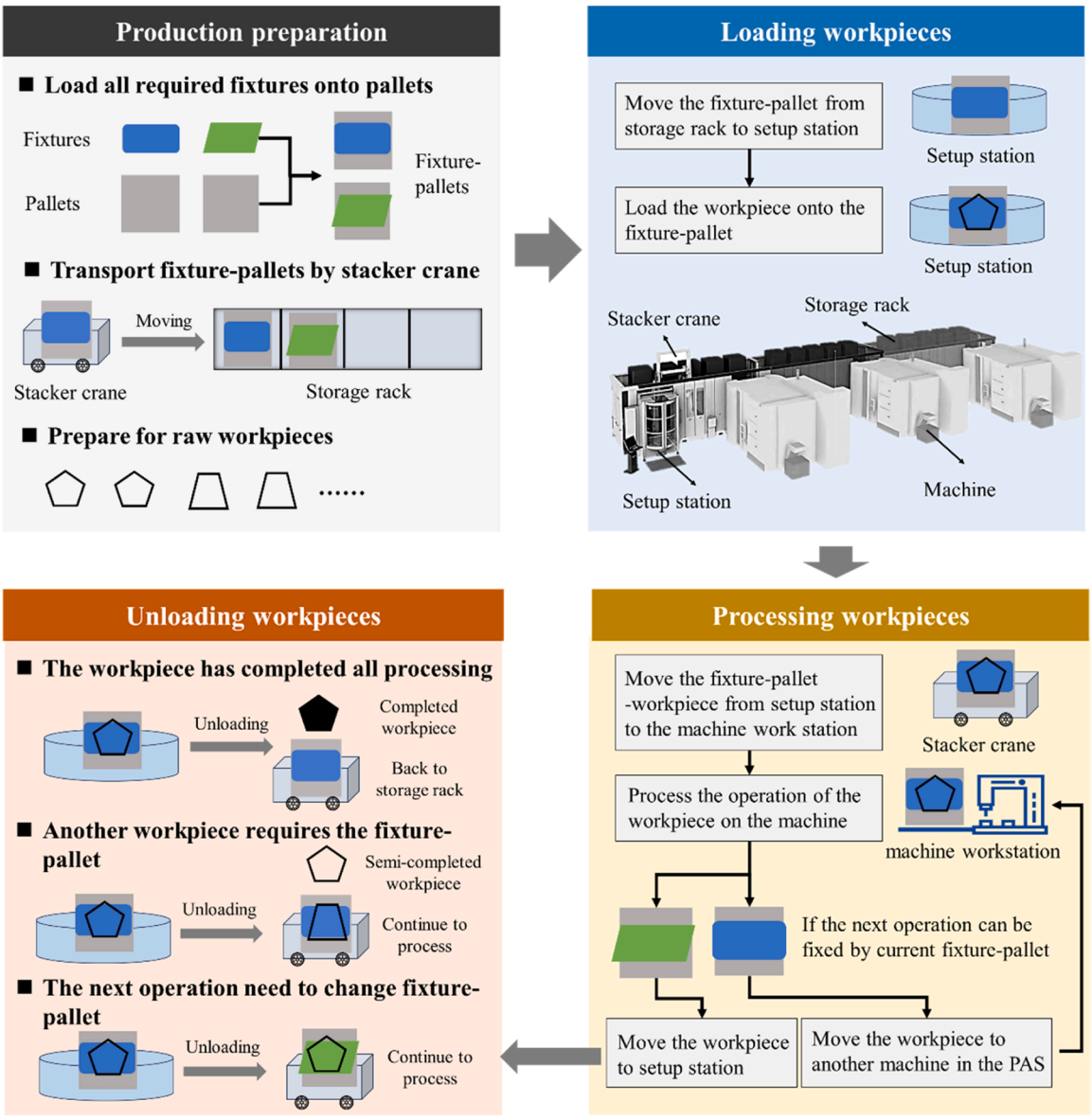


Fig. 2. Pictorial description of processing in the PAS.

by the author: the job could only be processed when the required fixture is available. Lee et al. [10] presented a problem of multi-fixturing pallets but were not aware of the flexibility of fixtures and did not propose a mathematical model, which is necessary as a prerequisite for heuristic methods. However, they treated the loading and unloading times as constants for all parts and decided in advance .

Some scholars have undertaken research on similar systems, such as flexible manufacturing systems (FMS). Maccarthy and Liu [17] categorized FMS into five main types: SFM (single flexible machine), FMC (flexible manufacturing cell), MMFMS (multi-machine flexible manufacturing system), and MCFMS (multi-cell flexible manufacturing system). The PAS studied in this paper represents an extended application of pallet automation within FMC, driven by the evolution of intelligent technologies. Godinho [18] provided a comprehensive overview of 40 articles about FMS, considering six criterias for evaluation: types of FMS, resource constraints, characteristics of workpieces, scheduling problems, and solving methods. Among the 40 articles, only a few considered multi-resource-constrained scheduling problem, and none took care of fixture or pallet resources. Some of the scholars investigated loading and unloading issue, like Keung [19] and Turkcan [20]. However, they focused on the loading and unloading of cutting tools, which differs significantly from the loading and unloading of flexible fixtures considered in this paper. Godinho [18] also pointed out that the majority of articles primarily use makespan as an evaluation metric, which aligns with the objective of this paper's research. In addition to time-based objectives (such as makespan, tardiness, earliness, etc.), there are various goals like cost, total production, machine utilization, energy consumption, forming multi-objective problems along with makespan that are also worthy of investigation in future studies [21].

After the in-depth research of resource-constrained FJSP, this NP-hard problem [22] needs to be solved. The current solution methods are classified into two main categories: exact and approximation methods [23,24]. The representative of the exact methods are the mathematical programming approach [25,26], Lagrange relaxation method [27], branch and bound method [28], and benders decomposition [29]. With the advancement of computer intelligence technology, an increasing number of scholars are employing approximation algorithms like priority dispatching rules [30,31], bottleneck-based heuristics [32], local search methods [33], machine learning tools [34–36], and meta-heuristic algorithms [37–40].

Table 2
2workpieces- 5machines- 3fixture-pallets (min).

Workpiece	Operation	Machine					Alternative fixture-pallet
		M_1	M_2	M_3	M_4	M_5	
1	O_{11}	2	6	5	3	4	F_1, F_2
	O_{12}	–	8	–	4	–	F_2, F_3
2	O_{21}	3	–	6	–	5	F_1, F_2, F_3
	O_{22}	4	6	5	–	–	F_2, F_3
	O_{23}	–	7	11	5	8	F_1, F_3

Table 3
The loading and unloading time of a workpiece on the fixture-pallet.

Fixture-pallet	Loading time or unloading Time
F_1	5
F_2	10
F_3	15

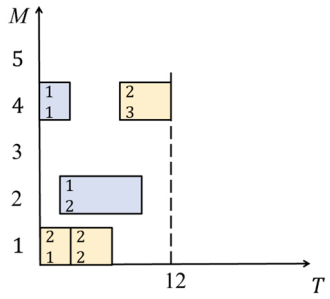


Fig. 3. Initial scheduling plan.

Adaptive large neighborhood search algorithm (ALNS), one of the meta-heuristic algorithms, has been widely applied due to its flexible framework. Ropke and Pisinger [41] first proposed ALNS to solve the pickup and delivery problem with time windows. Then, ALNS were brought to other fields' attention, such as manufacturing system [42] and healthcare [43]. ALNS uses multiple neighborhoods in the same searching process, adjusting the selection of operators dynamically based on performance. In order to strengthen the ability of local search and accept a worse solution within a certain range, ALNS was integrated with simulated annealing (SA) and proved to perform well [43]. However, the traditional way of perturbing the encoded string in ALNS is difficult to be applied in FJSP, and the multi-resource problem has no appropriate strategy. Therefore, this paper applied ALNS with SA and improved it with a new decoding method based on resources and new crossover and mutation operators.

To sum up, although the multi-resource constrained FJSP has been widely studied, few scholars considered the machine, fixture, and pallet resources simultaneously. Among the existing literature, there are insufficient studies considering the flexibility of fixtures, loading and unloading times, and the combination of fixtures and pallets.

3. The model of FJSP-PAS

3.1. System narrative

The depiction of processing within the PAS is illustrated in Fig. 2. The processing is divided into four primary phases: Production preparation, Loading workpieces, Processing workpieces, and Unloading workpieces. In the PAS, both machines and fixtures are flexible. Each operation has various available one or more fixture-pallets with different loading and unloading times and machines with different processing times.

1) **Production preparation**: All fixtures required for processing are loaded onto pallets and sequentially transported to storage racks. Simultaneously, raw workpieces have arrived in the PAS.
2) **Loading workpieces:** Move the fixture-pallet from the storage rack to the setup station, and then load the workpiece onto the fixture-pallet.
3) **Processing workpieces:** The fixture-pallet-workpiece is conveyed to the machine's workstation for processing by stacker crane. If the current fixture can be used in the next operation, it can be directly moved to the next machine. Otherwise, the workpiece needs to be unloaded and then loaded.
4) **Unloading workpieces:** There are three scenarios requiring unloading workpieces.

a) The workpiece has completed all processing. Unload the workpiece and move the fixture-pallet back to the storage rack.
b) Another workpiece requires the current fixture-pallet. Unload the current workpiece and load another workpiece onto the fixture-pallet.
c) The next operation of the current workpiece requires changing fixture-pallet. Unload the workpiece and load it onto another fixture-pallet.

3.2. Problem narrative

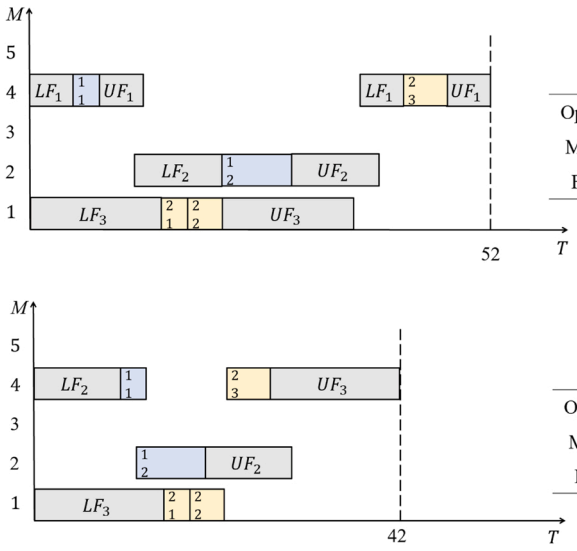
In order to specify the problem, it takes 2workpieces-5machines-3fixture-pallets as an example. As shown in Table 2, the available machines and fixture-pallets for each operation of each workpiece are listed. Table 3 shows the loading and unloading time of a workpiece on each fixture-pallet. For example, the first operation of Workpiece1(O_{11})can be processed on available machines $(M_1, M_2, M_3, M_4, M_5)$ and available fixture-pallets (F_1, F_2). When$F_2$is used for both the first and second operations of Workpiece2(O_{21}, O_{22}), and even if the two operations are not processed on the same machine, there is no need to load or unload. If fixture-pallet constraints are not considered, a feasible scheduling plan is shown in Fig. 3. Fig. 4 shows the result that one of the available fixture-pallets is selected for each operation without changing the processing machine. In the Gantt chart, the top number represents the workpiece number and the bottom number represents the operation number.LF_imeans loading the workpiece onF_i, andUF_imeans unloading the workpiece fromF_i. It can be observed that selecting different fixture-pallets may lead to varying makespan, making the problem significantly more complex.

3.3. Formulation of FJSP-PAS

The proposed FJSP-PAS is defined as follows. There are$|M|$machines and$|F|$ fixture-pallets in a PAS, and$|I|$workpieces waiting to be processed. Each workpiece i is composed of $J_i$operations with predetermined processing sequences. And the$j$*th*operation of the workpiece$i$ isO_{ij}.O_{ij}can be processed on a machine selected from the available machine set$M_{ij}$with different processing times T_{ijm} . Each operation of a workpiece also can be clamped and placed by one of the available fixture-palletsF_{ij} with the loading time or unloading timeE_f. The objective of scheduling is to minimize makespan$C_{\max}$by considering the loading and unloading times of workpieces from different fixture-pallets. However, if adjacent operations of the same workpiece choose the same fixture-pallet, there is no need for loading and unloading between the two operations.

Some assumptions are described as follows:

1. When time= 0, each machine and fixture-pallet can be used, and workpieces have arrived in PAS.
2. One workpiece can be processed by a machine and be fixed by a fixture-pallet.



Operation	O_{11}	O_{12}	O_{21}	O_{22}	O_{23}
Machine	M_4	M_2	M_1	M_1	M_4
Fixture	F_1	F_2	F_3	F_3	F_1

Operation	O_{11}	O_{12}	O_{21}	O_{22}	O_{23}
Machine	M_4	M_2	M_1	M_1	M_4
Fixture	F_2	F_2	F_3	F_3	F_3

Fig. 4. Different fixture-pallet selection plans based on Fig. 3.

Table 4
Notations used in the formulation.

Parameters	Definition
I	Workpiece set, $i = 1,2.....\|I\|$
M	Machine set, $m = 1,2.....\|M\|$
F	Fixture-pallet set, $f = 1,2.....\|F\|$
J_i	Operation set of workpiece i indexed by$j$, $j = 1,2.....\|J_i\|$
O_{ij}	jth operation of workpiece i, $i \in I, j \in J_i$
M_{ij}	Available machine set of operationO_{ij} , $i \in I, j \in J_i, M_{ij} \epsilon M$
F_{ij}	Available fixture-pallet set of operationO_{ij} , $i \in I, j \in J_i, F_{ij} \epsilon F$
T_{ijm}	Processing time of operationO_{ij} on machine m , $i \in I, j \in J_i, m \epsilon M_{ij}$
E_f	Loading time or unloading time of any workpiece on fixture-pallet $f, f \epsilon F$
L	A large number, L> 0
W_{ijm}	If the operationO_{ij}can be processed on machine m, $W_{ijm} = 1$, otherwise, $W_{ijm} = 0, i \in I, j \in J_i, m \epsilon M_{ij}$
B_{ijf}	If the operationO_{ij}can be fixed by fixture-pallet f, $B_{ijf} = 1$, otherwise, $B_{ijf} = 0, i \in I, j \in J_i, f \in F_{ij}$
Decision variables	Definition
FT_{ij}	Sum of loading and unloading time of operationO_{ij}before and after processing, $i \in I, j \in J_i$
S_{ij}	Starting time of operation$O_{ij}, i \in I, j \in J_i$
C_{ij}	Ending time of operation$O_{ij}, i \in I, j \in J_i$
$C_{\max}$	Total completion time (makespan), $C_{\max} > 0$
X_{ijm}	If the operation$O_{ij}$is processed on machine m,$X_{ijm} = 1$, otherwise,$X_{ijm} = 0$, $i \in I, j \in J_i, m \in M_{ij}$
$Y_{ijij'm}$	If operation$O_{ij}$is processed on machine$m$before operation$O_{ij'}$, $Y_{ijij'm} = 1$otherwise,$Y_{ijij'm} = 0$, $i \in I, j \in J_i, i' \in I, j' \in J_{i'}, m \in M_{ij} \cap M_{ij'}$
A_{ijf}	If the operation O_{ij} is fixed by fixture-pallet f, $A_{ijf} = 1$, otherwise, $A_{ijf} = 0, i \in I, j \in J_i, f \in F_{ij}$
$Z_{iji'j'f}$	If the fixture$f$ is used by operation$O_{ij}$and then operation$O_{ij'}$,$Z_{iji'j'f} = 1$, otherwise, $Z_{iji'j'f} = 0, i \in I, j \in J_i, i' \in I, j' \in J_{i'}, f \in F_{ij} \cap F_{ij'}$

3. The pallet is universal, and only one fixture-pallet and one workpiece can be placed on a pallet.
4. The fixture and pallet binding do not change during the processing, but the workpiece can be loaded and unloaded among different fixture-pallet。
5. The loading and unloading times of a workpiece cannot be ignored, and it varies on different fixture-pallets.
6. The processing of operation is supposed to be continuous, and breakdown is not allowed in principle once started.
7. The time of moving fixture-pallet or fixture-pallet-workpiece by stacker crane among setup station, machines, and storage racks is ignored.

Some notations are listed in Table 4 before formulating FJSP-PAS. The FJSP-PAS can be formulated as follows:

$$\min C_{\max} \tag{1}$$

$$\sum_{m=1}^{M} X_{ijm} = 1, \forall i \in I, \forall j \in J_i \tag{2}$$

$$\sum_{f=1}^{F} A_{ijf} = 1, \forall i \in I, \forall j \in J_i \tag{3}$$

$$FT_{ij} = \sum_{f=1}^{F} \left(E_f \times \left(1 - A_{i(j-1)f}\right) \times A_{ijf}\right) + \sum_{f=1}^{F} \left(E_f \times \left(1 - A_{i(j+1)f}\right) \times A_{ijf}\right), \quad \forall i \in I, j = 2,3.....(|J_i| - 1) \tag{4}$$

$$FT_{ij} = \sum_{f=1}^{F} \left(E_f \times \left(2 - A_{i(j-1)f}\right) \times A_{ijf}\right), \forall i \in I, j = |J_i| \tag{5}$$

$$FT_{ij} = \sum_{f=1}^{F} \left(E_f \times \left(2 - A_{i(j+1)f}\right) \times A_{ijf}\right), \forall i \in I, j = 1 \tag{6}$$

$$C_{ij} \le S_{i'f} + L\left(2 - A_{ijf} - A_{i'jf}\right) + L\left(1 - Z_{iji'jf}\right), \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall f \in F_{ij} \cap F_{i'f} \tag{7}$$

$$C_{i'j} \le S_{ij} + L\left(2 - A_{ijf} - A_{i'jf}\right) + L \times Z_{iji'jf}, \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall f \in F_{ij} \cap F_{i'f} \tag{8}$$

$$C_{ij} \le S_{i'j} + L\left(2 - X_{ijm} - X_{i'jm}\right) + L\left(1 - Y_{iji'jm}\right), \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall m \in M_{ij} \cap M_{i'j} \tag{9}$$

$$C_{i'j} \le S_{ij} + L\left(2 - X_{ijm} - X_{i'jm}\right) + L \times Y_{iji'jm} \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall m \in M_{ij} \cap M_{i'j} \tag{10}$$

$$S_{ij} + FT_{ij} + \sum_{m=1}^{M} \left(T_{ijm} \times X_{ijm}\right) \le C_{ij}, \forall i \in I, \forall j \in J_i \tag{11}$$

$$C_{ij} \le S_{i(j+1)}, \forall i \in I, j = 1,2.....(|J_i| - 1) \tag{12}$$

$$C_{ij} \le C_{\max}, \forall i \in I, \forall j \in J_i \tag{13}$$

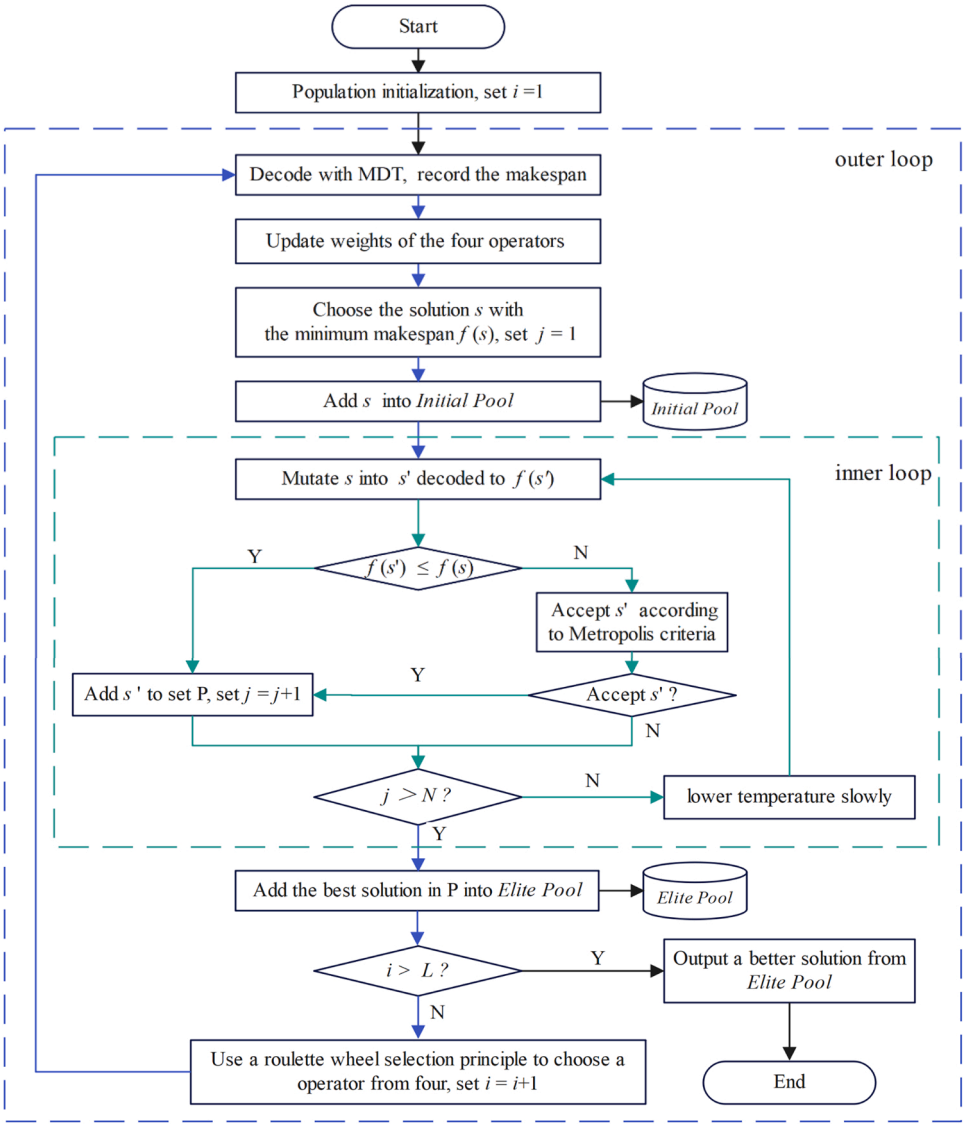


Fig. 5. The flowchart of IALNS-MDT.

$$A_{ijf} \le B_{ijf}, X_{ijm} \le W_{ijm}, Y_{ij\acute{i}\acute{j}m} \le X_{ijm}, Y_{ij\acute{i}\acute{j}m} \le X_{\acute{i}\acute{j}m}, \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall m \in M_{ij} \cap M_{i'j'}, \forall f \in F_{ij} \cap F_{i'j'} \tag{14}$$

$$Y_{ij\acute{i}\acute{j}m} + Y_{\acute{i}\acute{j}ijm} \le 1, Z_{ij\acute{i}\acute{j}f} + Z_{\acute{i}\acute{j}ijf} \le 1, \forall i \in I, \forall j \in J_i, \quad \forall i' \in I, \forall j \in J_{i'}, \forall m \in M_{ij} \cap M_{i'j'}, \forall f \in F_{ij} \cap F_{i'j'} \tag{15}$$

$$Z_{ij\acute{i}\acute{j}f} \le A_{\acute{i}\acute{j}f}, Z_{ij\acute{i}\acute{j}f} \le A_{ijf}, \quad \forall i \in I, \forall j \in J_i, \forall i' \in I, \forall j \in J_{i'}, \forall f \in F_{ij} \cap F_{i'j'} \tag{16}$$

$$A_{ijf}, X_{ijf}, Y_{ij\acute{i}\acute{j}m}, Z_{ij\acute{i}\acute{j}f} \epsilon \{0,1\}, \forall i \in I, \forall j \in J_i, \quad \forall i' \in I, \forall j \in J_{i'}, \forall m \in M_{ij} \cap M_{i'j'}, \forall f \in F_{ij} \cap F_{i'j'} \tag{17}$$

$$S_{ij}, C_{ij}, C_{\max} \ge 0, \quad \forall i \in I, \forall j \in J_i \tag{18}$$

The objective function aims to minimize the makespan as Eq. (1). Eqs. (2) and (3) guarantee that each operation can only be processed on one machine and fixed by one fixture-pallet at a time. Eqs. (4–6) are to compute the loading and unloading times of operationO_{ij}, which is different on the first and last operation of the workpiece. The first operation of the workpiece must be loaded first and the last operation of the workpiece must be unloaded after processing. The loading and unloading times depends on whether the current operation uses the same fixture as the subsequent operation or preceding operation. Eqs. (7)-(10) guarantee that each fixture-pallet and machine can only be used by one operation at a time. The sum of the start time, processing time, and loading and unloading time is no more than the completion time for each operation realized by Eq. (11). Eq. (12) guarantees that each workpiece is processed in order. Eq. (13) limits the makespan to no less than any completion time of all operations. Eqs. (14)-(16) describe the mutual relations among these decision variables. Eqs. (17) and (18) determine the basic variable types of some decision variables.

4. Improved ALNS with MDT

4.1. Solution approach

To solve the model in the last section, IALNS-MDT is proposed. The main four steps are listed as follows:

Step 1: Encoding and initialization.

Apply two-string encoding strategy and minimum machine load rule for initialization.

Step 2: Decoding with MDT.

Decode and allocate fixture-pallets for each operation according to the specific MDT rule.

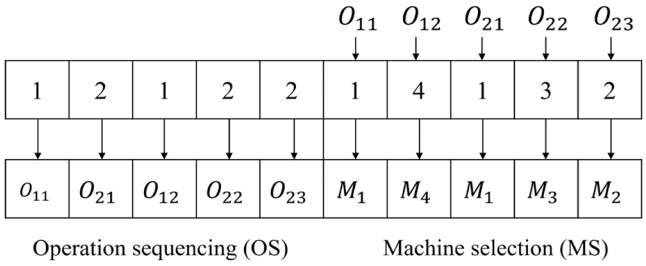


Fig. 6. Two-string encoding strategy based on OS, MS.

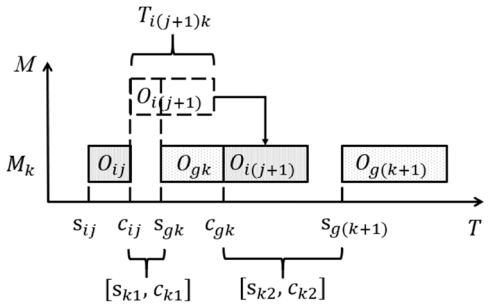


Fig. 7. First step of the decoding method.

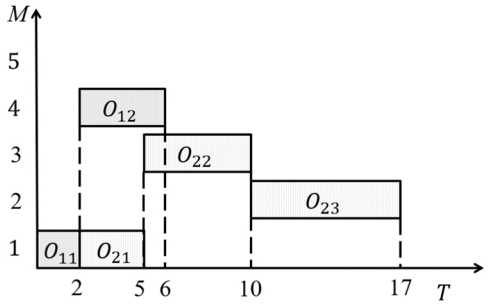


Fig. 8. The result of decoding the solution in Fig. 6.

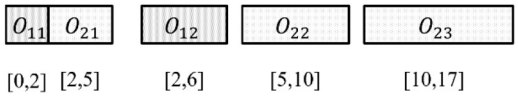


Fig. 9. The order of operations.

Step 3: Inner loop of local search.

Use SA to search for other solutions from the best solution of the last step and remember them.

Step 4: Four operators and selection.

One of four operators is selected by roulette wheel to generate a new population based on the dynamically adjusted weights.

Furthermore, the detailed flowchart of IALNS-MDT, is illustrated in Fig. 5, predominantly includes an outer loop (Step 1-Step 4) and an inner loop (Step 3). The outer loop dynamically selects one of the four different operators to perturb the machine selection (MS) and operation sequencing (OS) of the solution depending on roulette wheel selection principle. After that, decode them by MDT rule, obtaining the best solution for each generation which is stored in the *Initial Pool*. The termination condition of the outer loop is either reaching the specified maximum number of iterations$L$or when the best solution remains unchanged for a certain number of consecutive generations. The inner loop mainly utilizes the two-point mutation to perturb the fixture selection (FS) of the best solution from the last step and then precisely decode them. The acceptance probability of inferior mutated solutions is controlled by SA. The termination condition of the inner loop is that the number of acceptable solutions generated through perturbation reaches N. The best of N solutions is stored in the *Elite Pool*. Finally, after the outer loop terminates, the best solution in the *Elite Pool* is outputted as the final solution.

Table 5
The process of selecting a fixture-pallet for O_{21}.

Operation	O_{11}	O_{21}		
Machine	M_1	M_1		
Fixture	F_1	F_1	F_2	F_3
S_{ij}	0	2	2	2
Processing time	2	3	3	3
Loading and unloading time	$LF_1 = 5$	$UF_1 + LF_1 = 10$	$LF_2 = 10$	$LF_3 = 15$
Start time	0	7	7	7
Delay time	5	15	15	22
Workpiece occupied time	[0,7]	[7,20]	[7,20]	[7,25]
Machine occupied time	[7,0,0,0,0]	[20,0,0,0,0]	[20,0,0,0,0]	[25,0,0,0,0]
Fixture occupied time	[7,0,0]	[20,0,0]	[7,20,0]	[7,25,0]

4.2. Encoding and initialization

The two-string encoding strategy [44] is employed in this step, which consists of two strings: operation sequencing (OS) and machine selection (MS). One of the encoding results is shown in Fig. 6 from the example in Table 2. The length of each string is equal to the total number of operations of workpieces, which is 5 in the example. The number in OS represents the workpiece index, and the order it appears determines which operation this gene is. For example, the "2" in the fourth gene position represents O_{22}. MS are arranged in the sequence of operations, and the number in each gene position represents the machine selected by the corresponding operation.

According to the characteristics of FJSP and the encoding strategy, this paper adopts a minimum machine global load rule (GL), minimum machine local load rule (LL), and randomly generated rule (RL) [45] for initializing MS. OS is randomly generated. This method improves the quality of the initial solution for the next step of decoding.

4.3. Decoding with MDT

The decoding method based on MDT rule is mainly divided into the following two steps.

Step 1 Based on operation insertion method (OIM) [44], decode OS and MS.

All available time intervals on the machine$M_k$can be found as$[s_{kh}, c_{kh}]$(h=1,2…, representing the number of intervals). If$s_{kh} \geq c_{ij}$and$s_{kh} + T_{i(j+1)k} \leq c_{kh}$, the operation $O_{i(j+1)}$can begin to be processed ats_{kh}on theM_k. This process is illustrated in Fig. 7. And the final Gantt chart is shown in Fig. 8 by decoding the solution in Fig. 6.

Step 2 According to the MDT rule, arrange fixture-pallets for each operation and change the start and end time.

Firstly, order each operation by start time (see Fig. 9), taking the solution in Fig. 8 as an example.

Next, choose one fixture-pallet from its available fixture-pallets set for the first operationO_{11}. As shown in Table 5, update the occupied time for the workpiece, machine, and fixture-pallet after choosingF_1. Then, select a fixture-pallet for the second operation$O_{21}$from the three available fixture-pallets:F_1,F_2, andF_3. Choose the fixture-pallet among these three that minimizes the delay time. And then, follow the same steps to select fixture-pallets for all remaining operations. The number of available fixture-pallets for the initial operation determines the number of solutions$(s_1, s_2, s_3.)$. Choose the better solution as the optimal solution.

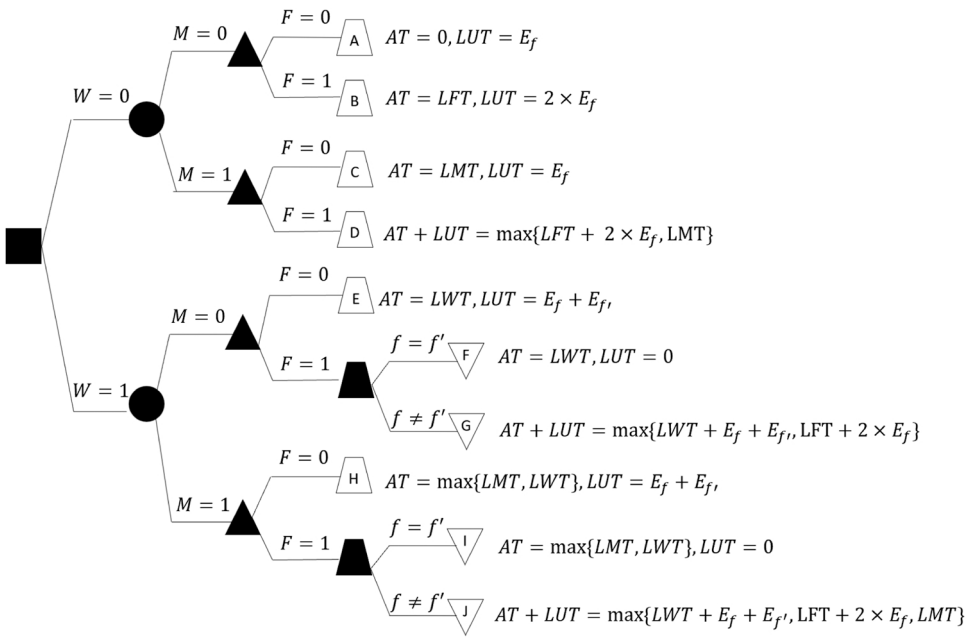


Fig. 10. The decision tree of AT and LUT.

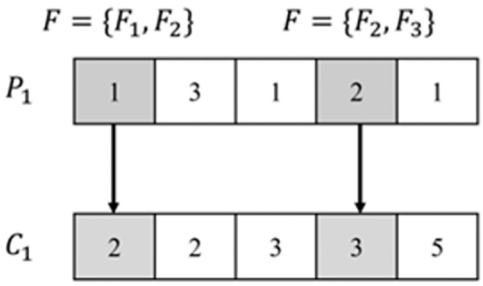


Fig. 11. The mutation strategy.

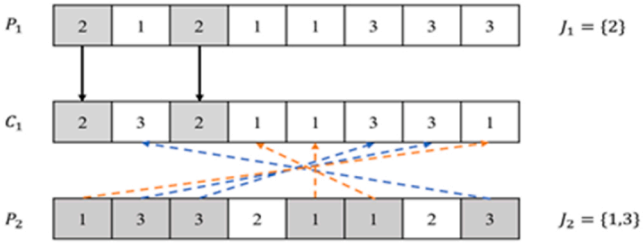


Fig. 13. Improved precedence operation crossover.

(1) The equation of Delay Time

The equation of Delay Time (DT) is the Eq. (19). Start Time (AT) is the practical start time of the operation. LUT is the loading and unloading time of the operation.$s_{ij}$is the start processing time of the operation obtained by Step 1.

$$\text{Delay Time}=\text{Start Time} + \text{Loading and unloading time-}s_{ij} \tag{19}$$

(2) The decision of Start Time and Loading and unloading time

Fig. 10 shows the decision tree of AT and LUT according to the use of the three resources: workpiece (W), machine (M), and fixture-pallet (F). When$W = 0$, it indicates that the current operation is the first one of the workpiece, whereas$W =$ 1indicates that the current workpiece has been processed. Similarly, $M = 0$and$F = 0$imply that the current operation is the first one to use the machine and the fixture-pallet, whereas$M =$ 1and$F = 1$indicate that the machine and fixture-pallet have been used for other operations before.

"f"is the fixture-pallet selected by the current operation, and "f'"is the fixture-pallet selected by the preceding operation of the current workpiece. "E_f"stands for the loading time or unloading time of the operation on the fixture-pallet f. There are three important variables: Last Workpiece Time (LWT), Last Machine Time (LMT), and Last Fixture-pallet Time (LFT), which respectively represent the preceding operation's end time of the current workpiece, current machine, and current fixture-pallet. LWT can also be considered as End Time (ET). The calculation equation is

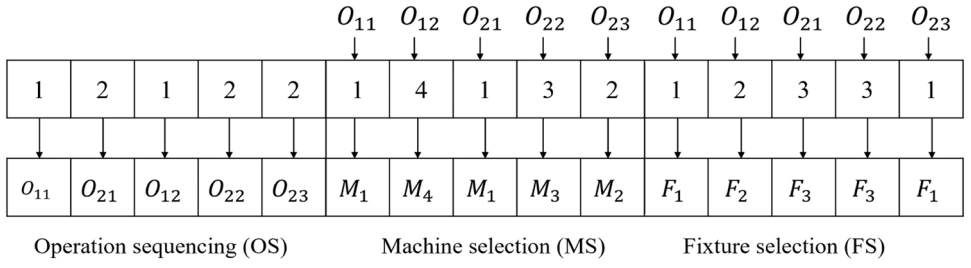


Fig. 12. Three-string encoding for decoding.

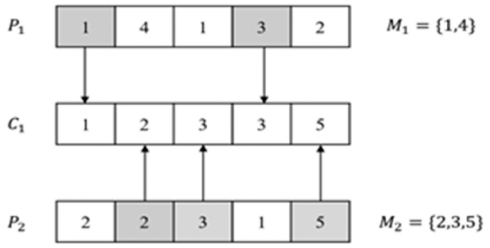


Fig. 14. Uniform crossover.

Fig. 15. A PAS in a Chinese engine manufacturing enterprise.

shown as follows, where "Processing Time (PT)" is the processing time of the operation:

$$LWT \setminus LMT \setminus LFT = Start\ Time + Loading\ and\ unloading\ time + Processing\ Time \tag{20}$$

(3) As shown in Fig. 10, according to the resource usage, the corresponding Start Time and Loading and unloading time can be determined in the following scenarios:

The current operation is the first one for the workpiece$W$:

A: At time zero, load the workpiece onto the fixture-pallet$f$.

B: Unload the other workpiece from the fixture-pallet$f$first, then load the current workpiece onto it and start processing.

C: After the preceding operation on the machine$m$is completed, load the current workpiece onto the fixture-pallet$f$and begin processing.

D: Unload the other workpiece from the fixture-pallet$f$first, then load the current workpiece onto the fixture-pallet$f$. Consider that processing can only start after the preceding operation on the machine$m$is completed.

(4) The current operation is not the first one for workpiece$W$:

E: After the preceding operation of the workpiece is completed, unload the workpiece from the fixture-palletf', then load the workpiece onto the fixture-pallet$f$and begin processing.

F: If the same fixture-pallet is used for the preceding and current operation of the workpiece, there is no need for loading and unloading. Processing can directly start after the preceding operation is completed.

G: If different fixture-pallets are used for the preceding and current operations of the workpiece, unload the existing workpiece from the fixture-pallet$f$, and then load the workpiece that was unloaded from the fixture-palletf'onto the fixture-pallet$f$.

H: After the preceding operation on the workpiece is completed, unload the workpiece from the fixture-pallet f', then load the workpiece onto the fixture-pallet$f$. Consider that processing can only start after the preceding operation on the machine$m$is completed.

I: If the same fixture-pallet is used for the preceding and current operations of the workpiece, there is no need for loading and unloading. However, processing can only begin after the preceding operation of the workpiece and on the machine$m$is completed.

J: If different fixture-pallets are used for the preceding and current operations of the workpiece, unload the existing workpiece from the fixture-pallet$f$, and then load the workpiece that was unloaded from the fixture-palletf'onto the fixture-pallet$f$. Consider that processing can only start after the preceding operation on the machine$m$is completed.

(4) Change the start and end time

After knowing the machine selection of each operation and operation sequencing from Step 1, the delay time is calculated to select the minimum delay time fixture for each operation, during which the Start Time and End Time of each operation are simultaneously calculated. The largest End Time is makespan. Therefore, the scheduling scheme is formed.

It's worth noting that the Start Time and End Time of each operation are not as completely accurate as the optimization model but can provide a good initial solution quickly for local search using SA in the next step. This will significantly improve the efficiency of local search because it is performed on the best solution obtained during this

Table 7
The algorithm parameters.

Notation	Meaning	Value
Population	Population scale in the algorithm.	20
Iterations	The number of iterations	1000
Limitation	The iteration stops when the optimal value is unchanged for limited consecutive generations	400
GL, LL, RL	The proportion of global, local, and random initial rule	0.5, 0.3, 0.2
p_1,p_2	Scoring different performances of operators	0.8, 1.5
T_0	The initial temperature of SA	200
N	Number of local searches for optimal solution under SA	10

Table 6
Three scales of PMF cases.

Scale	Case	Workpiece	Operation	Machine	Fixture
Small	PMF1	2	3	5	3
	PMF2	5	3	5	5
	PMF3	8	6	6	5
	PMF4	10	5	5	6
Medium	PMF5	15	10	8	13
	PMF6	15	9	8	10
	PMF7	15	9	10	12
	PMF8	10	15	15	11
	PMF9	12	15	8	13
Large	PMF10	20	10	15	23
	PMF11	20	14	18	20
	PMF12	30	15	20	22
	PMF13	40	15	30	20
	PMF14	50	9	35	30
	PMF15	60	10	35	35

Table 8
The comparison of initial encoding solutions.

Cases	Decoding with three strings	Decoding with MDT
PMF1	45	45
PMF2	90	83
PMF3	212	178
PMF4	352	332
PMF5	592	493
PMF6	681	576
PMF7	647	591
PMF8	889	675
PMF9	948	754
PMF10	1244	922
PMF11	1704	1284
PMF12	3343	2468
PMF13	3629	3144
PMF14	3820	3160
PMF15	4254	3504

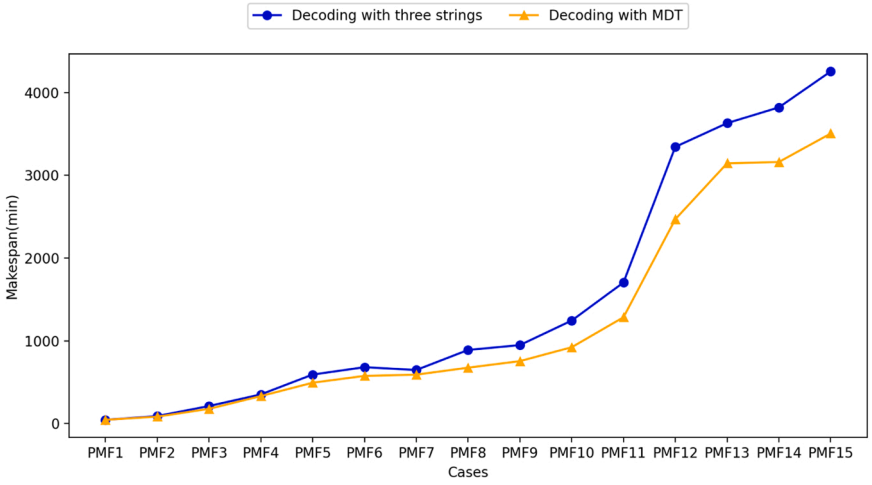


Fig. 16. Initial solutions of the two decoding methods for each case.

Table 9
The comparison of experimental results by solving PMF cases (a).

Cases	GUROBI			GA		TS		SA	
	T_1	*gap*	Opt_1	$\overline{T}_2$	$\overline{Opt}_2$	$\overline{T}_3$	$\overline{Opt}_3$	$\overline{T}_4$	$\overline{Opt}_4$
PMF1	0.03	0.00 %	42	2.8	43.7	0.3	61.0	0.7	61.0
PMF2	0.37	0.00 %	47	10.6	61.0	33.0	79.2	1.9	79.2
PMF3	900	5.88 %	102	70.5	163.6	98.8	173.2	60.5	190.4
PMF4	900	28.80 %	208	75.7	274.8	99.2	263.4	84.6	290.6
PMF5	1800	50.10 %	443	354.7	426.8	321.0	389.6	339.5	406.0
PMF6	1800	52.40 %	420	355.1	520.0	182.2	509.0	255.2	526.4
PMF7	1800	34.40 %	487	253.4	546.4	239.5	486.8	175.0	513.6
PMF8	1800	38.40 %	497	427.8	632.0	190.3	599.2	171.8	642.8
PMF9	1800	54.20 %	747	399.9	743.0	284.0	755.2	329.5	776.2
PMF10	3600	27.80 %	864	299.7	849.2	311.9	852.0	257.9	881.8
PMF11	3600	48.10 %	1393	414.2	1235.6	466.8	1302.8	544.9	1344.2
PMF12	3600	–	–	715.6	2494.8	892.3	2368.4	860.1	2405.0
PMF13	3600	–	–	1108.6	2813.6	957.3	2865.6	952.3	2974.4
PMF14	3600	–	–	798.2	2856.2	862.0	3001.8	971.4	3089.0
PMF15	3600	–	–	882.8	3402.8	983.9	3278.4	1043.3	3350.6

Table 10
The comparison of experimental results by solving PMF cases (b).

Cases	IALNS-MDT				Q_1	Q_2	Q_3	Q_4
	$\overline{T}_5$	*max*	*min*	$\overline{Opt}_5$				
PMF1	2.4	42	42	42.0	0.00 %	–3.89 %	–31.15 %	–31.15 %
PMF2	8.8	52	47	49.2	4.68 %	–19.34 %	–37.88 %	–37.88 %
PMF3	59.4	122	112	116.6	14.31 %	–28.73 %	–32.68 %	–38.76 %
PMF4	48.4	204	190	196.8	–5.38 %	–28.38 %	–25.28 %	–32.28 %
PMF5	318.3	393	374	382.0	–13.77 %	–10.50 %	–1.95 %	–5.91 %
PMF6	371.2	424	395	412.8	–1.71 %	–20.62 %	–18.90 %	–21.58 %
PMF7	242.2	455	432	442.6	–9.12 %	–19.00 %	–9.08 %	–13.82 %
PMF8	300.9	509	470	487.4	–1.93 %	–22.88 %	–18.66 %	–24.18 %
PMF9	341.1	675	652	664.4	–11.06 %	–10.58 %	–12.02 %	–14.40 %
PMF10	292.2	859	802	824.8	–4.54 %	–2.87 %	–3.19 %	–6.46 %
PMF11	306.4	1255	1186	1219.2	–12.48 %	–1.33 %	–6.42 %	–9.30 %
PMF12	636.1	2389	2313	2356.8	–	–5.53 %	–0.49 %	–2.00 %
PMF13	919.0	2839	2701	2770.2	–	–1.54 %	–3.33 %	–6.87 %
PMF14	546.8	2944	2771	2846.8	–	–0.33 %	–5.16 %	–7.84 %
PMF15	782.7	3264	3206	3239.0	–	–4.81 %	–1.20 %	–3.33 %

decoding step. And then the optimal solution is decoded precisely after local search to reduce the running time of the algorithm.

4.4. Inner loop of local search

In order to enhance the local search capability of the algorithm and compensate for the limitations of fixture-pallet selection when using the MDT rule for decoding, this paper incorporates the SA algorithm into IALNS and improves it. Since the standard SA operator may easily miss some better solutions, this paper performs multiple searches for each decoded optimal solution to increase the possibility of finding better solutions. Additionally, a memory function of SA to save the current best solution is added. The algorithm flow of SA is illustrated in the inner loop of Fig. 5, and the specific steps are described as follows:

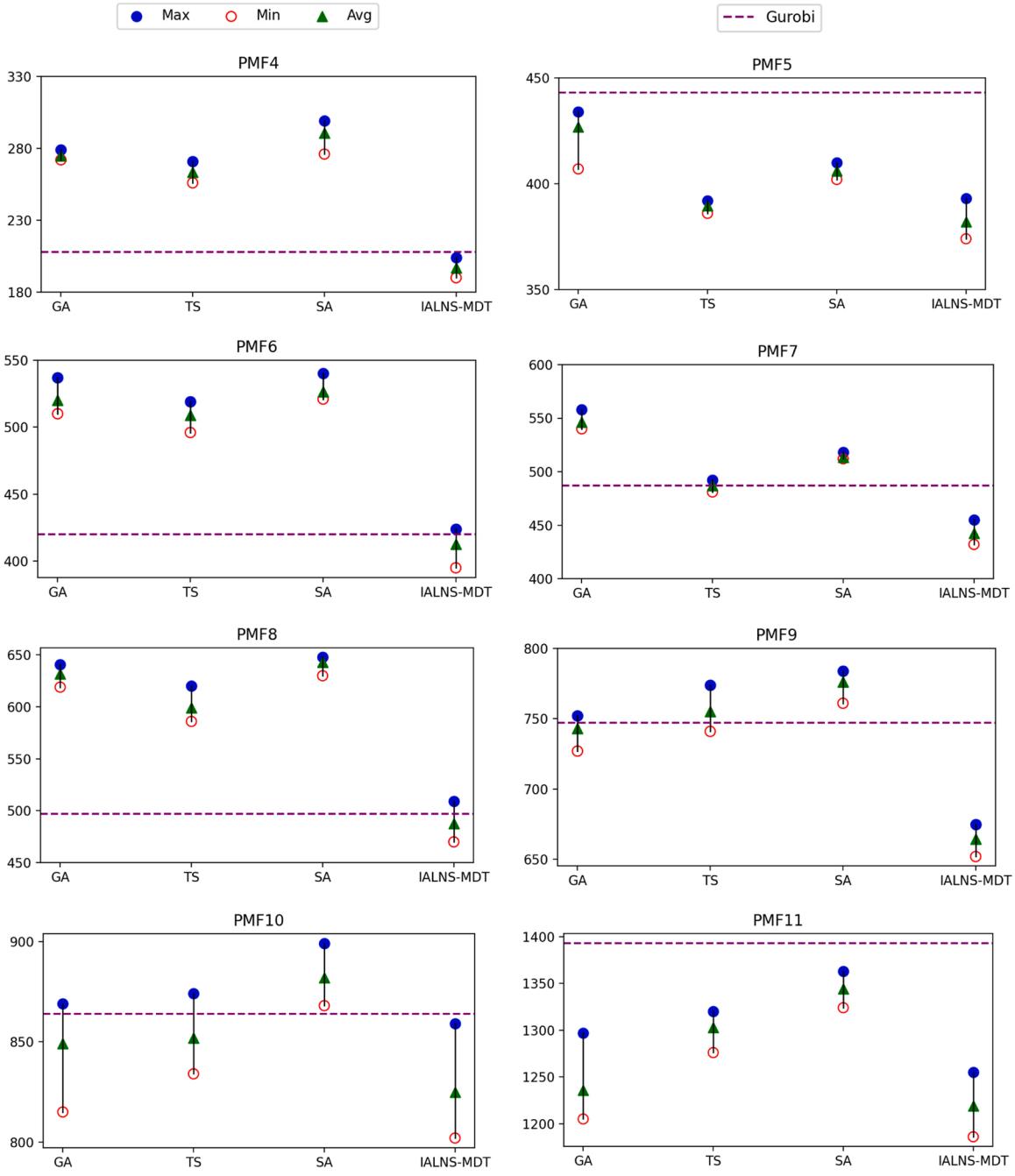


Fig. 17. The maximum, minimum, and average makespan for each algorithm.

Step 1: Set$j = 1$. The two-point mutation is carried out on the current solution$s$and obtains the solutions'. The mutation strategy is shown in Fig. 11.

Step 2: Decode the three strings (see Fig. 12) by finding the commonly available times of the fixture-pallet and the machine for operation insertion, which is similar to OIM and just change available times intervals on the machine to common times between the machine and fixture-pallet. Since FS is known (i.e.,$A_{ijf}$is known), the loading and unloading times can be calculated according to constraints (4) - (6). If the makespan $f(s')$ of the mutated solution is better than $f(s)$, then s' is put into the set$P$, and set$j = j + 1$. Otherwise, go to the next step.

Step 3: Determine whether the mutated solution s' can be accepted under Metropolis criteria. At temperature T_j, calculate the accepted probability$u$, and the equation of $u$is shown in Eq. (20) [46]. Generate a random number$r$, if $r < u$, then s' is stored in the set$P$, and set$j = j + 1$.

$$u = e^{\frac{-(f(s)-f(s))}{T_j}} \tag{20}$$

Step 4: Determine whether the number of solutions$j$in the set$P$reaches the set value N. If $j > N$, end the loop and store the optimal solution in P into the *Elite Pool*. Instead, reduce the temperature of SA and repeat Step 1-Step 3. The temperature reduction equation is shown in Eq. (21) [46]:

$$T_j = \frac{T_0}{\log(1+j)} \tag{21}$$

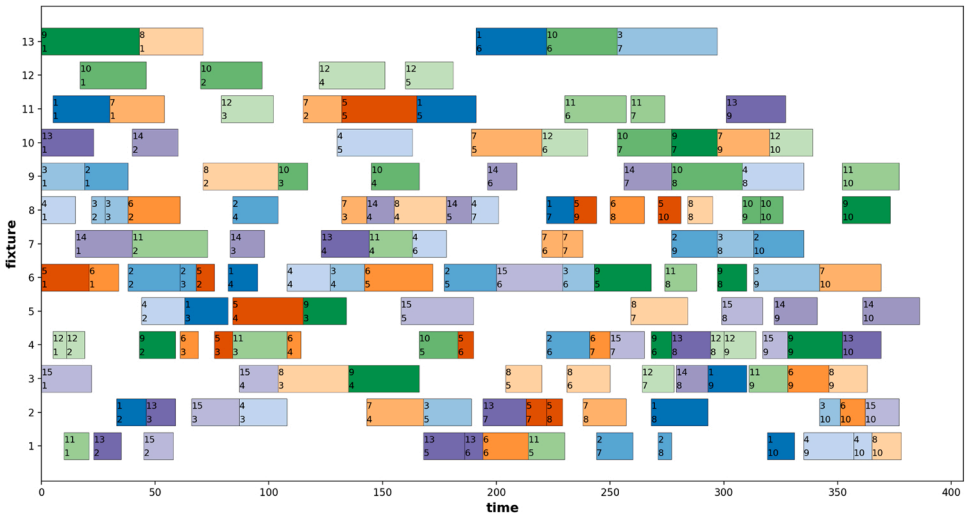


Fig. 18. The fixture-time Gantt chart of Case 1.

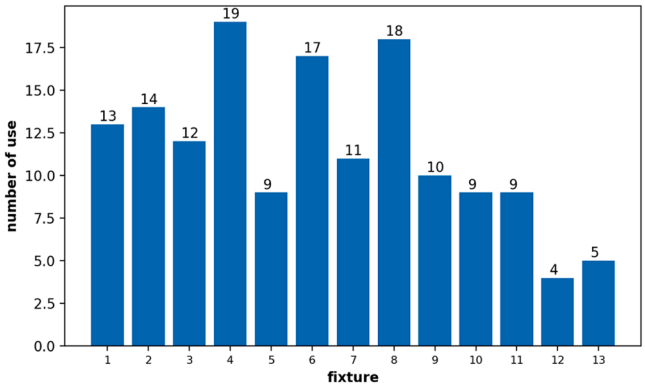


Fig. 19. The number of times each fixture-pallet used of Case 1.

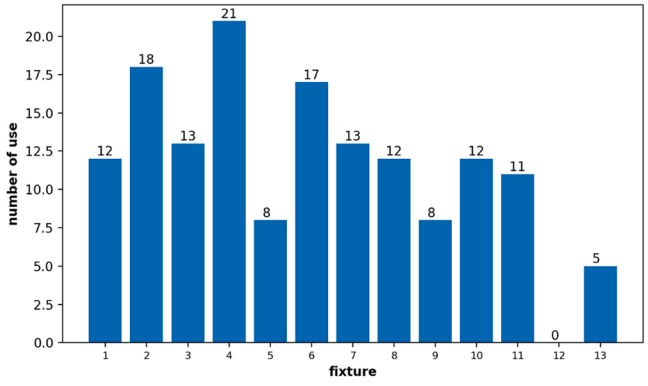


Fig. 21. The number of times each fixture-pallet used of Case 2.

4.5. Four operators and selection

In IALNS-MDT, four different operators are used to change the OS and MS of each individual, allowing the solution to explore different directions. Improved Precedence Operation Crossover (IPOX) (see Fig. 13) and two-point swap mutation are applied for OS. Uniform Crossover (UC) (see Fig. 14) and two-point mutation (see Fig. 11) are applied for MS.

Operator 1: Crossing with one of the solutions from *Initial Pool,* which is a set of best solutions after decoding with MDT.

Operator 2: Crossing with one of the solutions from *Elite Pool,* which

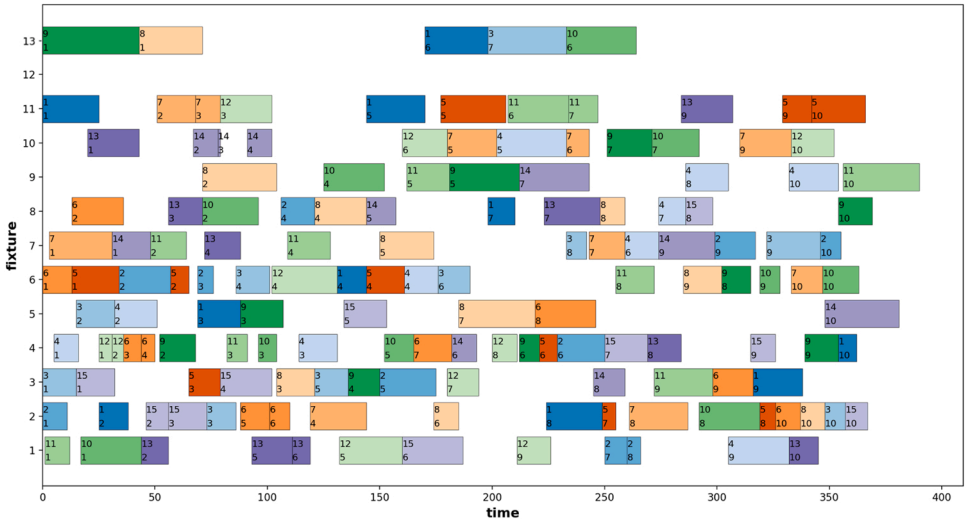


Fig. 20. The fixture-time Gantt chart of Case 2.

Table 11
The comparison of experimental results of three cases.

Case	Fixture-pallet resource	Makespan
1	All thirteen kinds of fixture-pallets	386
2	Except F12	390
3	Except F4	445

is a set of best solutions after the inner loop.

Operator 3: Crossing with one of the solutions from the current generation.

Operator 4: Crossing with one of the solutions, which is randomly generated.

In order to accelerate the population to evolve in a favorable direction during the initial iterations, roulette wheel selection (RWS) strategy for four crossover operators is used. After crossover, the individual is decoded with MDT, and then the scores of the operators are updated according to the performance of the four operators. The RWS method used in this paper is listed as follows:

Step 1: Update the scores of the operators based on the makespan after decoding.

Local optimum: If the optimal solution of the current generation is better than the preceding generation, the score of the corresponding operator plusp_1.

Global optimum: If the current generation's optimal solution is superior to the best solution in the Elite Pool, the score of the corresponding operator plus$p_2 (p_2 > p_1)$.

Step 2: Calculate the weight$w_j (j \in \{1,2,3,4\})$of each operator based on its corresponding score$\{s_1, s_2, s_3, s_4\}$. These weights represent the probability of each operator being selected.

$$w_j = \frac{s_j}{\sum_{i=1}^{4} s_i} \tag{22}$$

In conclusion, IALNS-MDT dynamically selects anyone from various operators to enhance the diversity of the neighboring field. Additionally, SA, which is applied in IALNS, avoids being trapped in a local optimum solution owing to premature convergence. The decoding method by MDT rule selects a fixture-pallet for minimizing the delay in completion time for each operation compared to the original solution when considering loading and unloading times and unavailable times, which reduces the complexity of the encoding and considers the mutual constraints among resources, thereby improving the quality of decoding.

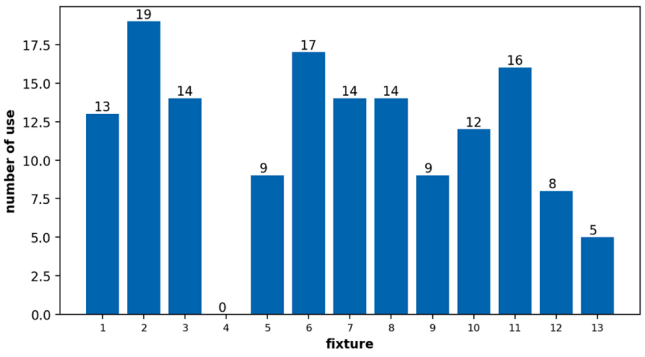


Fig. 23. The number of times each fixture-pallet was used in Case 3.

5. Case study

5.1. The design of experiments

All experiments were conducted in a desktop computer with an Intel (R) Core(TM) i5–10210 U CPU @ 1.60 GHz 2.11 GHz , Windows 10, and PyCharm 2023.2.4 (Professional Edition) python 3.11.5.

PAS has been adopted in one of the largest Chinese engine manufacturing enterprises. A PAS in the enterprise is taken as an example (see Fig. 15), which has thirteen fixture-pallets and six machines. In the PAS, pallets are shared among different machines due to their mobility and the same size as workstations, so the fixtures are not unloaded once it is attached to the pallets one to one. Each operation of each workpiece involves specific available fixture-pallets and machines. The loading times or unloading times of different workpieces on the same fixture-pallet are generally similar, typically ranging from 5 to 20 min, which depends on the size of the workpiece and the complexity of the fixture.

For the experiments, 15 PMF (workpiece, machine, and fixture-pallet) cases were generated by real data provided by the PAS in Fig. 15, including three scales: 4 small-scale cases, 5 medium-scale cases,

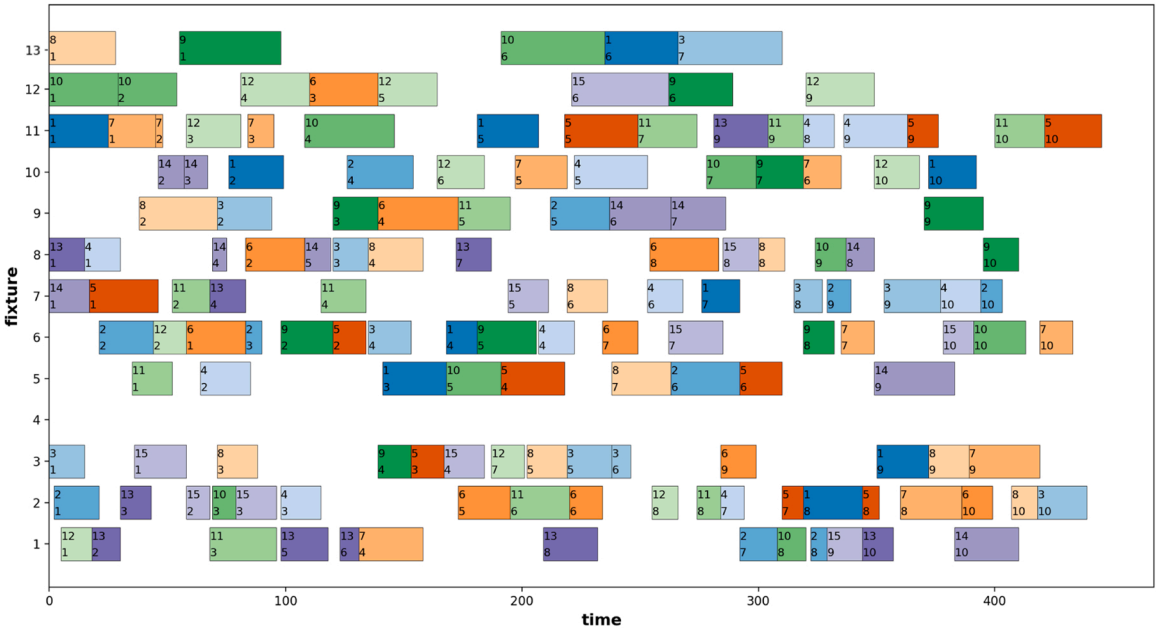


Fig. 22. The fixture-time Gantt chart of Case 3.

and 6 large-scale cases, as listed in Table 6, which respectively correspond to operation quantities of 0–100, 100–200, and above 200. The detailed data were generated as follows. DU (a, b) stands for discrete uniform distribution with range [a, b].

1) The loading and unloading time of workpieces (E_f) ~ DU (5,20)
2) Number of alternative machines for each operation ($|M_{ij}|$) ~ DU (1,5)
3) Number of alternative fixtures for each operation ($|F_{ij}|$) ~ DU (1,5)
4) The processing time for each operation depends on the real data, ranging from 1 to 200 min.

Set the algorithm parameters listed in Table 7, which are determined by a series of tests. In IALNS-MDT, the population should not be vast because each individual in the population needs to do a crossover. A relatively high number of iterations is required to ensure the algorithm's convergence. Additionally, the limitation is vital to reduce wasted time on useless iterations. The initial temperature$T_0$for SA is set to 200. At this temperature, the acceptance probability for a suboptimal solution with a difference of 10 is 96.59 %. After 500 iterations, the acceptance probability decreases to 73.48 %, which helps maintain diversity within the population.

In order to verify the performance of the proposed algorithm for solving FJSP-PAS and provide fixture-pallet allocation solutions for enterprises, the following numerical experiments are designed:

1. Analysis of the performance of MDT. The initial solutions generated by decoding with MDT are compared with those by decoding with three strings (OS, MS, and FS).
2. Analysis of the performance of IALNS-MDT. The PMF cases are solved using GUROBI, GA, Tabu search, SA, and IALNS-MDT respectively, to prove the correctness of the model as well as to verify the ability of IALNS-MDT to find the optimal solution.
3. Fixture-pallet allocation experiments. Use IALNS-MDT to solve PMF5. Analyze the fixture-pallet selection for each operation under the optimal solutions and provide a fixture-pallet allocation scheme for the enterprise.

5.2. Analysis of the performance of MDT

The efficiency of MDT rule can be verified by comparing the decoding with MDT and the decoding with three strings. The method is based on three strings (OS, MS, and FS) and then finds a time period during the available time of the fixture-pallet and the machine in which the operation is inserted by OIM.

By five repeated experiments for each case from PMF1 to PMF15, the comparative experimental data of the above two methods are shown in Table 8. It is demonstrated that the decoding method with MDT obtains a smaller makespan for the initial solution.

Fig. 16 is a supplement to Table 8. The solid dots indicate the initial solutions of "Decoding with three strings" and the solid triangles indicate the initial solutions of "Decoding with MDT". It can be seen that the initial solutions of the proposed decoding method in this paper are better than the other method, and the gap between the two increases as the size of the cases increases.

5.3. Analysis of the performance of IALNS-MDT

The aim is to verify the model's correctness and the performance of the proposed method by comparing the makespan of PMF cases among GUROBI, GA, Tabu search, SA, and IALNS-MDT.

The GUROBI is a high-powered mathematical programming solver commonly used in academia and industry, which was selected to solve the mixed-integer programming model established in this paper. The time limits for solving small-scale, medium-scale, and large-scale cases are set to 900 s, 1800 s, and 3600 s, respectively. If the GUROBI solver fails to find the theoretically global optimal solution within the specified time, the currently obtained best solution will be taken as the result.

Genetic Algorithm (GA) is one of the most commonly used metaheuristic methods for solving FJSP problems, but it is easy to be trapped in local optimal. Tabu search (TS) is a local search algorithm, behind which the main idea is to iteratively explore the solution space by moving from one solution to a neighboring solution while avoiding getting trapped in the local optimum. Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It allows the algorithm to escape the local optimum by occasionally accepting worse solutions early in the optimization process. As the temperature decreases, the algorithm transitions to a more greedy strategy, focusing on exploiting the neighborhood of the current solution to converge towards an optimal or near-optimal solution. For GA, the population size was set to 100, with 150 iterations, the crossover probability 0.7, and the mutation probability 0.2. For TS, the tabu object is the objective value in each neighborhood, the length of the tabu table is 5, and the number of iterations is 5000. If the number of iterations reaches 3000 or all neighborhood operations are tabu, the result is output. For SA, the number of iterations is 20,000 and the initial temperature is 200.

To measure the performance of the algorithms, four metrics,Q_1,Q_2,$Q_3$and$Q_4$are introduced, defined as shown in the Eq. (23). Here, $\overline{Opt}_2$,$\overline{Opt}_3$,$\overline{Opt}_4$and$\overline{Opt}_5$represent the averages of five times of GA, TS, SA, and IALNS-MDT for solving the same case, while$Opt_1$is the optimal solution obtained by GUROBI within the set time. If$Q_i$is negative, it indicates that IALNS-MDT performs better than other methods. The larger value of these metrics, the larger difference in the performance gap between methods. $\overline{T}_i(i = 1, 2, 3, 4, 5)$is the average time of solving cases. *max* and *min* indicate the maximum and minimum solutions of IALNS-MDT in each case.

$$Q_i = \frac{Opt_5 - Opt_i}{Opt_i} \times 100\%, i = 1, 2, 3, 4 \tag{23}$$

Table 9 and Table 10 show the comparison of experimental results by solving PMF cases. The specific analysis is as follows:

1) The 15 cases illustrate that the proposed algorithm can get better solutions than GUROBI, GA, TS, and SA in a shorter time in most cases.
2) For PMF1 and PMF2, GUROBI and IANLS-MDT all can obtain the optimal solution, confirming the correctness of the model and algorithm.
3) For PMF3, the solution obtained by IALNS-MDT is worse than GUROBI, which may be affected by the randomness of the algorithm.
4) Due to the fixture-pallet constraints, the problem's complexity increases, and even in small-scale cases, GUROBI cannot find the optimal solution for PMF3 and PMF4 within 900 s
5) For medium and large-scale cases, GUROBI cannot obtain the optimal solution within 1800 s and 3600 s. In PMF12-15, GUROBI even cannot find a feasible solution, which illustrates the complexity of the cases. In contrast, IALNS-MDT can get better solutions than the other four methods.
6) With the increase in the case size, the IALNS-MDT algorithm still maintains the advantage. Although the solving time is relatively increased, the solution is still superior to the other four methods.
7) The advantage of the IALNS-MDT algorithm in terms of solution time is crucial for enterprises, especially if they need to schedule orders quickly in some emergency situation.

In order to further compare the performance of GUROBI, GA, TS, SA, and IALNS-MDT, eight cases are selected to draw comparison figures (see Fig. 17), depicting the minimum, maximum, and average makespan of each algorithm in different cases. The dashed line represents the optimal solution of GUROBI in a given time. It highlights that the

proposed algorithm can achieve better solutions, and even the maximum solutions of the proposed algorithm are better than the minimum solutions of other algorithms in some cases.

5.4. Fixture-pallet allocation experiments

In the actual production process, the allocation of fixture-pallet resources is of paramount importance. Once fixture-pallets are assigned to a particular PAS, this allocation relationship typically remains unchanged. This means that if a critical fixture-pallet is not allocated to its corresponding PAS, it may result in an extended time of makespan. On the other hand, if a fixture-pallet is allocated to a PAS but has a low utilization rate, it can lead to a waste of resources.

Using PMF5 as an example, this experiment presents the Gantt chart (see Fig. 18) of fixture-time obtained by IALNS-MDT algorithm with makespan 386. In the Gantt chart, the top number represents the workpiece number and the bottom number represents the operation number. Fig. 19 shows the number of times each fixture-pallet is used. It can be observed thatF_{12}has the lowest frequency of use, only 4 times throughout the entire production process, while F4 is used up to 19 times.

1) Supposing$F_{12}$is no longer bound to this PAS, the makespan becomes 390 with the Gantt chart in Fig. 20 and the number of fixture-pallets used in Fig. 21. And it has little impact on the makespan compared to the initial plan. Therefore, in this case, the enterprise can reassign$F_{12}$to another PAS, potentially increasing its utilization rate.
2) In the other case, if$F_4$is removed from the PAS, the makespan would significantly increase to 445 (see Table 11). The Gantt chart (see Fig. 22) and the use times of fixture-pallets (see Fig. 23) are also presented. Obviously,$F_4$is greatly necessary for this case.

Fixture-pallet resources are highly valuable for enterprises. In order to be able to neither waste fixture-pallet resources in production but also make the existing fixture-pallet resources to the maximum extent to meet the production, the enterprise can obtain the fixture-pallet distribution plan by analyzing the historical order data and make appropriate adjustments according to the utilization rate.

6. Conclusions and future work

With flexible automation becoming a critical part of upgrading enterprises, PAS is widely applied, so the fixture and pallet resources cannot be ignored. How to allocate and schedule various resources to minimize makespan has become a vital consideration for enterprises.

This paper studies a new machine-fixture-pallet resources constrained flexible job shop scheduling problem considering loading and unloading times under a pallet automation system, which takes the different fixture-pallets available for each operation, different loading and unloading times of the workpiece, and fixture-pallets shared between all machines in the PAS into account. To solve the problem, a mixed-integer programming model is established, and IALNS-MDT algorithm is proposed, which enhances the quality of solutions by employing the decoding method with MDT rule and the local search method with SA. The correctness and effectiveness of IALNS-MDT are verified by solving PMF cases from a leading engine manufacturing enterprise in China. The comparative results show that the proposed method performs better in different scale cases. Furthermore, the importance of fixture-pallet resource allocation is emphasized and gives an example to illustrate, which inspires enterprises to make better resource allocation decisions.

However, there are several aspects of this study for future improvement. Firstly, the transportation time for fixture-pallet combinations within PAS or between multiple PAS is worth considering, especially for large factories where transportation can be time-consuming. Additionally, the energy consumption in the production process is also crucial, which can help enterprises obtain greater economic benefits. Finally, real production often faces various abnormal conditions, such as equipment breakdown and urgent orders inserted. Therefore, how to respond dynamically and timely is worthy of attention.

Conflict of interest

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

Data Availability

Data sharing is not applicable to this article as production secrets of the cooperated company are involved.

Acknowledgments

This work was funded by the National Natural Science Foundation of China (Grant No. 52275499), and the National Key Research and Development Program of China (No. 2022YFF0605700).

References

[1] Gania IP, Stachowiak A, Oleśków-Szłapka J. Flexible manufacturing systems: industry 4.0 solution. DEStech Trans Eng Technol Res 2018. https://doi.org/10.12783/dtetr/icpr2017/17583.
[2] Li, L., Jiang, Y. Fastems white paper on pallet automation application in China; 2018. Available from: https://www.fastemschina.com.
[3] Girish BS, Jawahar N. A particle swarm optimization algorithm for flexible job shop scheduling problem. 2009 IEEE Int. Conf. Autom. Sci. Eng. Bangalore, India: IEEE,; 2009. p. 298–303. https://doi.org/10.1109/COASE.2009.5234153.
[4] Wu X, Wu S. An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem. J Intell Manuf 2017;28:1441–57. https://doi.org/10.1007/s10845-015-1060-6.
[5] Roshanaei V, Azab A, ElMaraghy H. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int J Prod Res 2013;51:6247–74. https://doi.org/10.1080/00207543.2013.827806.
[6] Mahmud S, Chakrabortty RK, Abbasi A, Ryan MJ. Switching strategy-based hybrid evolutionary algorithms for job shop scheduling problems. J Intell Manuf 2022;33:1939–66. https://doi.org/10.1007/s10845-022-01940-1.
[7] Nouiri M, Bekrar A, Jemai A, Niar S, Ammari AC. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J Intell Manuf 2018;29:603–15. https://doi.org/10.1007/s10845-015-1039-3.
[8] Shen L, Dauzère-Pérès S, Neufeld JS. Solving the flexible job shop scheduling problem with sequence-dependent setup times. Eur J Oper Res 2018;265:503–16. https://doi.org/10.1016/j.ejor.2017.08.021.
[9] Yu J.-M., Doh H.-H., Kim J.-S., Lee D.-H., Nam S.-H. Scheduling for a Reconfigurable Manufacturing System with Multiple Process Plans and Limited Pallets/Fixtures 2012.
[10] Lee D-K, Shin J-H, Lee D-H. Operations scheduling for an advanced flexible manufacturing system with multi-fixturing pallets. Comput Ind Eng 2020;144:106496. https://doi.org/10.1016/j.cie.2020.106496.
[11] Chan FTS, Wong TC, Chan LY. Flexible job-shop scheduling problem under resource constraints. Int J Prod Res 2006;44:2071–89. https://doi.org/10.1080/00207540500386012.
[12] Wu X, Peng J, Xiao X, Wu S. An effective approach for the dual-resource flexible job shop scheduling problem considering loading and unloading. J Intell Manuf 2021;32:707–28. https://doi.org/10.1007/s10845-020-01697-5.
[13] Xu G, Jiang X, Sun D, Yang B, Deng R, Fu J. Flexible job-shop scheduling based on improved firefly algorithm. 2022 2nd Int. Conf. Comput. Control Robot. ICCCR. Shanghai, China: IEEE,; 2022. p. 90–7. https://doi.org/10.1109/ICCCR54399.2022.9790158.
[14] Thörnblad K, Strömberg AB, Patriksson M, Almgren T. Scheduling optimisation of a real flexible job shop including fixture availability and preventive maintenance. Eur J Ind Eng 2015;9:126. https://doi.org/10.1504/EJIE.2015.067451.
[15] Mati Y, Lahlou C, Dauzère-Pérès S. Modelling and solving a practical flexible job-shop scheduling problem with blocking constraints. Int J Prod Res 2011;49:2169–82. https://doi.org/10.1080/00207541003733775.
[16] Liao Q, Yang J, Zhou Y. Sustainable scheduling of an automatic pallet changer system by multi-objective evolutionary algorithm with first piece inspection. Sustainability 2019;11:1498. https://doi.org/10.3390/su11051498.
[17] Maccarthy B., Liu J. A new classification scheme for flexible manufacturing systems n.d.
[18] Godinho Filho M, Barco CF, Tavares Neto RF. Using genetic algorithms to solve scheduling problems on flexible manufacturing systems (FMS): a literature survey,

classification and analysis. Flex Serv Manuf J 2014;26:408–31. https://doi.org/10.1007/s10696-012-9143-6.
[19] Keung KW, Ip WH, Yuen D. An intelligent hierarchical workstation control model for FMS. J Mater Process Technol 2003;139:134–9. https://doi.org/10.1016/S0924-0136(03)00194-8.
[20] Turkcan A, Akturk MS, Storer RH. Due date and cost-based FMS loading, scheduling and tool management. Int J Prod Res 2007;45:1183–213. https://doi.org/10.1080/00207540600559955.
[21] Destouet C, Tlahig H, Bettayeb B, Mazari B. Flexible job shop scheduling problem under Industry 5.0: a survey on human reintegration, environmental consideration and resilience improvement. J Manuf Syst 2023;67:155–73. https://doi.org/10.1016/j.jmsy.2023.01.004.
[22] Fattahi P, Saidi Mehrabad M, Jolai F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J Intell Manuf 2007;18:331–42. https://doi.org/10.1007/s10845-007-0026-8.
[23] 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. https://doi.org/10.1016/j.ijpe.2016.01.016.
[24] Xie J, Gao L, Peng K, Li X, Li H. Review on flexible job shop scheduling. IET Collab Intell Manuf 2019;1:67–77. https://doi.org/10.1049/iet-cim.2018.0009.
[25] Gomes MC, Barbosa-Póvoa AP, Novais AQ. Optimal scheduling for flexible job shop operation. Int J Prod Res 2005;43:2323–53. https://doi.org/10.1080/00207540412331330101.
[26] Hu K, Zhang X, Gen M, Jo J. A new model for single machine scheduling with uncertain processing time. J Intell Manuf 2017;28:717–25. https://doi.org/10.1007/s10845-015-1033-9.
[27] Hajibabaei M, Behnamian J. Fuzzy cleaner production in assembly flexible job-shop scheduling with machine breakdown and batch transportation: Lagrangian relaxation. J Comb Optim 2023;45:112. https://doi.org/10.1007/s10878-023-01046-1.
[28] Soto C, Dorronsoro B, Fraire H, Cruz-Reyes L, Gomez-Santillan C, Rangel N. Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. Swarm Evol Comput 2020;53:100632. https://doi.org/10.1016/j.swevo.2019.100632.
[29] Juvin C, Houssin L, Lopez P. Logic-based benders decomposition for the preemptive flexible job-shop scheduling problem. Comput Oper Res 2023;152:106156. https://doi.org/10.1016/j.cor.2023.106156.
[30] Wang H, Peng T, Nassehi A, Tang R. A data-driven simulation-optimization framework for generating priority dispatching rules in dynamic job shop scheduling with uncertainties. J Manuf Syst 2023;70:288–308. https://doi.org/10.1016/j.jmsy.2023.08.001.
[31] Niu H, Wu W, Xing Z, Wang X, Zhang T. A novel multi-tasks chain scheduling algorithm based on capacity prediction to solve AGV dispatching problem in an intelligent manufacturing system. J Manuf Syst 2023;68:130–44. https://doi.org/10.1016/j.jmsy.2023.03.007.
[32] Cao J, Guan Z, Yue L, Ullah S, Sherwani RAK. A Bottleneck degree-based migrating birds optimization algorithm for the PCB production scheduling. IEEE Access 2020;8:209579–93. https://doi.org/10.1109/ACCESS.2020.3033002.
[33] Aydin ME, Fogarty TC. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. J Intell Manuf 2004;15:805–14. https://doi.org/10.1023/B:JIMS.0000042665.10086.cf.
[34] Zhang N, Shen Y, Du Y, Chen L, Zhang X. Counterfactual-attention multi-agent reinforcement learning for joint condition-based maintenance and production scheduling. J Manuf Syst 2023;71:70–81. https://doi.org/10.1016/j.jmsy.2023.08.011.
[35] Zhang L, Feng Y, Xiao Q, Xu Y, Li D, Yang D, et al. Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times. J Manuf Syst 2023;71:257–73. https://doi.org/10.1016/j.jmsy.2023.09.009.
[36] Ogunfowora O, Najjaran H. Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. J Manuf Syst 2023;70:244–63. https://doi.org/10.1016/j.jmsy.2023.07.014.
[37] Gao KZ, Suganthan PN, Pan QK, Chua TJ, Cai TX, Chong CS. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. J Intell Manuf 2016;27:363–74. https://doi.org/10.1007/s10845-014-0869-8.
[38] Xiong W, Fu D. A new immune multi-agent system for the flexible job shop scheduling problem. J Intell Manuf 2018;29:857–73. https://doi.org/10.1007/s10845-015-1137-2.
[39] Zhang W, Zheng Y, Ahmad R. The integrated process planning and scheduling of flexible job-shop-type remanufacturing systems using improved artificial bee colony algorithm. J Intell Manuf 2023;34:2963–88. https://doi.org/10.1007/s10845-022-01969-2.
[40] Sun X, Shen W, Vogel-Heuser B. A hybrid genetic algorithm for distributed hybrid blocking flowshop scheduling problem. J Manuf Syst 2023;71:390–405. https://doi.org/10.1016/j.jmsy.2023.09.017.
[41] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 2006;40:455–72. https://doi.org/10.1287/trsc.1050.0135.
[42] Rifai AP, Nguyen H-T, Dawal SZM. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl Soft Comput 2016;40:42–57. https://doi.org/10.1016/j.asoc.2015.11.034.
[43] Lusby RM, Schwierz M, Range TM, Larsen J. An adaptive large neighborhood search procedure applied to the dynamic patient admission scheduling problem. Artif Intell Med 2016;74:21–31. https://doi.org/10.1016/j.artmed.2016.10.002.
[44] Gao L, Zhang G, Wang X. Flexible Job-shop Scheduling Intelligent Algorithm and its Application. Huazhong University of Science & Technology Press; 2012.
[45] Zhang G, Gao L, Li P, Zhang C. Improved genetic algorithm for the flexible job-shop scheduling problem. J Mech Eng 2009;45:145. https://doi.org/10.3901/JME.2009.07.145.
[46] Gendreau M, Potvin J-Y, editors. Handbook of Metaheuristics, Vol. 272. Cham: Springer International Publishing; 2019. https://doi.org/10.1007/978-3-319-91086-4.