International Journal of Industrial Engineering Computations 13 (2022) 457–472

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Bi-objective optimization of identical parallel machine scheduling with flexible maintenance and job release times

Yarong Chen[a,b], Zailin Guan[a], Chen Wang[b], Fuh-Der Chou[b] and Lei Yue[c*]

[a]*School of Mechanical Science and Engineering, Huazhong University of Science and Technology, China*
[b]*School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, Zhejiang, China*
[c]*School of Mechanical and Electrical Engineering, Guangzhou University, Guangzhou 510000, China*

C H R O N I C L E

Article history:
Received May 21 2022
Received in Revised Format July 31 2022
Accepted August 6 2022
Available online August, 6 2022

Keywords:
Identical parallel machine scheduling
Flexible maintenance
Bi-objective optimization
MIP
M-NSGA-II

A B S T R A C T

This paper investigates an identical parallel machine scheduling problem with flexible maintenance and job release times and attempts to optimize two objectives: the minimization of the makespan and total tardiness simultaneously. A mixed-integer programming (MIP) model for solving small-scale instances is presented first, and then a modified NSGA-II (M-NSGA-II) algorithm is constructed for solving medium- and large-scale instances by incorporating several strategies. These strategies include: (i) the proposal of a decoding method based on dynamic programming, (ii) the design of dynamic probability crossover and mutation operators, and (iii) the presentation of neighborhood search method. The parameters of the proposed algorithm are optimized by the Taguchi method. Three scales of problems, including 52 instances, are generated to compare the performance of different optimization methods. The computational results demonstrate that the M-NSGA-II algorithm obviously outperforms the original NSGA-II algorithm when solving medium- and large-scale instances, although the time taken to solve the instances is slightly longer.

1. Introduction

Parallel machine scheduling (PMS) is an important branch of production scheduling (Cheng & Sin, 1990). In the literature on PMS, most studies assume that machines are available at all times (Liu et al., 2020; Kim et al., 2019). However, in real-world manufacturing, machines need to be maintained and hence may become unavailable during a certain period, which makes it highly important to jointly schedule production and maintenance tasks to simultaneously improve system availability and system throughput. The problem of scheduling jobs with maintenance reduces to the problem often referred to in the literature as scheduling with machine availability constraints. Lee and Chen (2000) first researched a parallel machine scheduling problem (PMSP) with maintenance in which each machine must be maintained once in the planning horizon, and they proposed branch and bound (B&B) algorithms based on the column generation approach. To date, many studies on various PMS with different preventive maintenance (PM) strategies have been carried out. Unfortunately, most research focuses on single-objective optimization, especially the makespan, and few studies focus on multiple objective optimization. Berrichi et al. (2009) researched the PMSP with machine reliability and simultaneously minimized the makespan and machine system unavailability. Based on this research, Wang and Liu (2015) proposed a multi-objective PMSP with two kinds of resources (machines and moulds) and flexible PM activities on resources, to simultaneously minimize the makespan, the machine system unavailability, and the mould system unavailability. With the development of intelligent manufacturing, customer satisfaction plays an increasingly vital role in both the manufacturing and service industries. On-time delivery is becoming increasingly important. Therefore, the bi-objective to simultaneously minimize the makespan and total tardiness is studied in this paper.
In addition, most studies on the joint optimization of production scheduling and PM usually assume that jobs reach zero time.

* Corresponding author Tel: +86-180-6250-8408
E-mail: leileiyok@gzhu.edu.cn (L. Yue)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2022.8.003

However, in real-world manufacturing, it is very common for jobs to arrive at different times. Many PMSP studies consider job release times without considering integration optimization with PM (Kramer et al., 2020; Abedi et al., 2015; Nessah et al., 2008; Yalaoui & Chu, 2006). Cui and Lu (2017) studied the single machine scheduling problem with machine flexible maintenance and job release times. Chen et al. (2021) studied two identical parallel machines with flexible maintenance to minimize the makespan without considering the job's release time. To the best of our knowledge, the PMSP with PM and job release times has not been documented in the literature. In this research, we extended the number of machines from references (Cui & Lu, 2017; Chen et al., 2021) to multiple machines and considered job release times to simultaneously minimize the makespan and total tardiness. Our objective is to determine the assignment of jobs to the parallel machines, the decision of the maintenance activities, and the optimal sequence of jobs and PM on each machine. The MIP model and an M-NSGA-II are successfully customized to solve the considered problem.

The rest of the paper is organized as follows: the next section surveys the related literature. A problem description is presented in Section 3. The MIP model is proposed in Section 4. Section 5 provides details about the modified NSGA-II. The experimental results are demonstrated in Section 6, and finally, Section 7 includes conclusions and discusses future research.

2. Literature review

2.1. PMSP with preventive maintenance

PMSPs incorporated with PM can be divided into different models according to different criteria. The basic criterion is the number of maintenance tasks, including single maintenance (Xu et al., 2009; Wang & Wei, 2011) or multiple maintenance (Hashemian et al., 2014; Li et al., 2017; Xu et al., 2008). In addition, maintenance may be performed on one of the parallel machines or on all parallel machines. Some earlier studies assumed that maintenance is only performed on some parallel machines (Tan et al., 2011) or only once within the planning horizon (Lee & Chen, 2008; Yoo & Lee, 2016). Recent research has generally assumed that all parallel machines have multiple maintenance, and the related research is divided into three kinds: fixed interval maintenance, flexible maintenance, and variable maintenance in the form of maintenance activities. Fixed interval maintenance involves periodically performing fixed time maintenance according to a fixed time interval. Yoo and Lee (2016) studied the PMSP in which each machine requires maintenance activity once within a given time window. Li et al. (2017) studied the PMSP in which each machine is subject to periodic maintenance. Xu et al. (2009) considered two PMSPs in which one machine is periodically unavailable. Flexible maintenance is fixed time maintenance within a flexible time window or upper threshold. Xu et al. (2008) studied a PMSP with ε-almost periodic maintenance activities to minimize the makespan. Sun and Li (2010) studied a similar problem and assumed that the largest consecutive processing time for each machine cannot exceed an upper limit T. Chen et al. (2021) studied two identical PMSPs with flexible maintenance to minimize the makespan. Variable maintenance is a kind of maintenance in which the maintenance interval and time depend on the condition of the machines and/or jobs. Wang and Wei (2011) studied PMSPs with deteriorating maintenance activity, that is, when delaying maintenance increases the time required to perform it. Wu et al. (2020) considered the dispatching-dependent deterioration of machines and machine-health-dependent production rates and proposed a dynamic dispatching and PM model to minimize the weighted long-run average waiting costs of MTO systems. Moradi and Zandieh (2010) studied the joint production and maintenance problem of parallel machines, and they concluded that the availability of machines is related to the failure rate and repair rate.

2.2. Multi-objective optimization of PMSP

Instead of a single objective, multiple but conflicting objectives are often considered when a manager executes production scheduling. Thus, a Pareto front consisting of a set of non-dominated solutions is required so that the manager can choose one of the alternatives from the set while preparing the production plan. However, compared with a single objective, such as minimizing the makespan, multi-objective optimization of PMSPs is rarely studied. According to the number of objectives, this problem can be divided into two types: bi-objective and multi-objective. For bi-objective optimization, in the context of green manufacturing, minimizing the energy or power consumption and cost is an important objective; therefore, the objective is to minimize the power and the makespan (Wang et al., 2018; Anghinolfi et al., 2021) as well as the tardiness penalty and power cost (Fang & Lin, 2013). In addition, the bi-objective to minimize the makespan and the total weighted earliness and tardiness of jobs (Abedi et al., 2015) as well as the makespan and the system unavailability (Berrichi et al., 2009), Moradi and Zandieh (2010)) can also be found in the literature. For multiple objective optimizations, based on the literature (Berrichi et al., 2009), the bi-objective is extended to three; Wang and Liu (2015) proposed a multi-objective PMSP with two kinds of resources (machines and moulds) and with flexible preventive maintenance activities on resources, and presented an integrated optimization method with NSGA-II adaptation. Bandyopadhyay and Bhattacharya (2013) proposed a modified version of NSGA-II with a new mutation algorithm for a PMSP with three objectives. Cochran et al. (2003) proposed a two-stage multi-population genetic algorithm to solve PMSPs with multiple objectives.

2.3. Solution methodology of the PMSP

The solutions to PMSP generally include two decisions: one assigns jobs to machines, and the second sorts of the jobs assigned to the machine. These two decisions can be made in parallel, i.e., allocation first and then sequencing, or sequencing can be performed during assignment. The solution method for PMSP in the literature mainly includes the exact method, rule-based heuristics, the meta-heuristic algorithm and the intelligent evolutionary algorithm.

The exact method can obtain the optimal solution to the problem, but it is only suitable for small-scale instances. The exact methods proposed for solving the PMSP are mainly the B&B algorithm (Nessah et al., 2008; Yalaoui & Chu, 2006) and mathematical models (Wu & Wang, 2018; Naderi & Roshanaei, 2020). To improve the efficiency of the exact models, the lower bounds and the dominance properties have been incorporated into the B&B algorithm. In addition, Alhadi et al. (2020) presented a polynomial-time approximation scheme to generate an approximate Pareto Frontier that minimizes the maximum lateness and makespan. Rule-based heuristics are widely used in the real-world manufacturing industry (Lee, 2018). The rules proposed for the problem $\mathrm{Pm}//C_{max}$ include the LPT, MULTIFIT, COMBINE, and LISTFIT (Gupta & Ruiz-Torres, 2001); those proposed for the problem $\mathrm{Pm}//\sum T_j$ include the EDD, SPT, TPI, MDD, KPM, Minimum Slack and BHG (Biskup et al., 2008); and those proposed for the problem $\mathrm{Pm}//\sum w_j T_j^2$ are the QB and QBP procedures (Schaller & Valente, 2018). Considering the job's release time, Akturk and Ozdemir (2001) proposed some rules, namely, the ATC (apparent tardiness cost), X-RM (X-dispatch ATC), KZRM (Kanet and Zhou approach to ATC), and COVERT (weighted cost over time), for the problem $1/\mathrm{r}_j/\sum T_j$. Meta-heuristics and intelligent evolutionary algorithms are the focus of the existing research. Kayvanfar et al. (2017) proposed an intelligent water drop algorithm for PMSP with controllable processing times. Dipak and Gupta (2018) proposed an improved cuckoo search algorithm to minimize the makespan for identical PMSP. Abdeljaoued et al. (2020) proposed a simulated annealing meta-heuristic for PMS under resource constraints.

3. Problem statement

This paper investigates the problem that a set of jobs $J = \{J_1, \cdots, J_j \cdots, J_n\}$ is to be scheduled on a set of identical machines $M = \{M_1, \cdots, M_i \cdots, M_m\}$, and the objectives are to simultaneously minimize the makespan and total tardiness. The problem can be denoted as $\mathrm{Pm}/r_j, \mathrm{nr}, \mathrm{FPM}/(C_{max}, \sum T_j)$, where "Pm" represents the identical parallel machines, "r_j" represents jobs released at different times, "nr" represents jobs that are non-resumable, and "FPM" represents flexible preventive maintenance, which means that the continuous processing time of the machines cannot exceed the maintenance threshold UT. Additionally, we assume that $p_j \leq UT$ for all jobs. The maintenance time is denoted by mt. The Gantt chart of a schedule for this problem is shown in Fig. 1. As shown in Fig. 1, each schedule consists of jobs and PMs, and the number of jobs on machine M_i is n_i. $J_{i[k]}$ is the job processed in the k^{th} position of machine M_i, $r_{i[k]}$ is the release time of $J_{i[k]}$, $p_{i[k]}$ is the processing time of $J_{i[k]}$, and $C_{i[k]}$ is the completion time of $J_{i[k]}$. Assume $d_{i[k]}$ is the due date of $J_{i[k]}$; then, the tardiness of $J_{i[k]}$ is calculated by the equation $T_{i[k]} = max\{C_{i[k]} - d_{i[k]}, 0\}$. Let B_{ib} be the set of jobs of the b^{th} batch in machine M_i, i.e., $B_{ib} = \{J_{i[1]}, \cdots, J_{i[k_b]}\}$, and $|B_{ib}| = k_b$. Let TP_{ib} be the sum processing time of the jobs in batch B_{ib}, i.e., $TP_{ib} = \sum_{J_{i[j]} \in B_{ib}} p_{i[j]}$. PM_{ib} is the b^{th} PM of machine M_i, and $C_{i[n_i]}$ is the makespan of machine M_i.

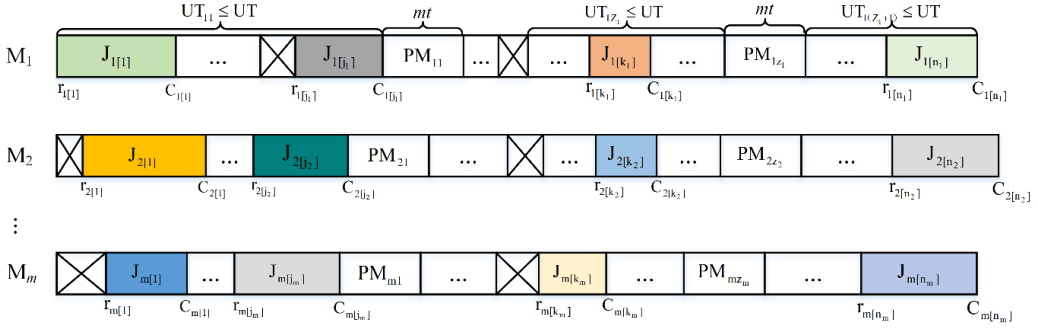

Fig. 1. A Gantt chart of a solution for the identical PMSP with m machines and n jobs

Theorem 1. Problem P: $\mathrm{Pm}/r_j, \mathrm{nr}, \mathrm{FPM}/(C_{max}, \sum T_j)$ is NP-Hard.

Proof. The sequencing problem of each machine in $\mathrm{Pm}/r_j, \mathrm{nr}, \mathrm{FPM}/(C_{max}, \sum T_j)$ is equivalent to the bin-packing problem. Because both the bin-packing problem and the $\mathrm{P2}//C_{max}$ problem have been proven to be NP-Hard, the problem $\mathrm{P2/nr}, \mathrm{FPM}/C_{max}$ is NP-Hard. In addition, the problem $1/r_j/\sum T_j$ is NP-Hard (Nessah et al. (2008)); thus, $\mathrm{Pm}/r_j/\sum T_j$ is NP-Hard. The problem $\mathrm{Pm}/r_j, \mathrm{nr}, \mathrm{FPM}/(C_{max}, \sum T_j)$ extends the number of identical parallel machines from 2 to m and considers the constraint of the job's release time and the objective of minimizing total tardiness; thus, problem $\mathrm{Pm}/r_j, \mathrm{nr}, \mathrm{FPM}/(C_{max}, \sum T_j)$ is NP-Hard.

Definition: Full loading (FL) means that the cumulative processing time of the jobs during two consecutive PMs, or the sum processing time of the jobs in a batch, is equal to the maintenance threshold UT. When $TP_{ib} = UT$, batch B_{ib} is fully loaded; otherwise, batch B_{ib} is not fully loaded, and the waste time is equal to $UT - TP_{ib}$.

4. MIP model

Based on the following assumptions, an MIP model for the problem studied in this paper is established: (1) each job has a release time; (2) the processing time, release time and due date of the job are known in advance; (3) the maintenance threshold and maintenance time are known in advance; (4) the time allocated to any machine for processing the same job is the same; (5) each machine can only process one job at any given time and shall maintain continuous processing, which shall not be

pre-empted and interrupted by other jobs; and (6) there is no job importance, that is, the weight is not considered. The input parameters and decision variables are introduced before developing the proposed model.

Input Parameters

J set of jobs
M set of machines
j, k job index from set J
i machine index from set M
p_j the processing time of job J_j
r_j the release time of job J_j
d_j the due date of job J_j
C_j the completion time of job J_j
ST_{ik} the start time at the k^{th} position of machine M_i
Q_{ik} the cumulative processing time at the k^{th} position of machine M_i
CT_{ik} the completion time of machine M_i in position k
UT threshold of the cumulative processing time that the machine can process continuously
mt time required for maintenance
L a sufficiently large positive integer, it is safe to set $L = \sum p_j + n * mt$

Decision variables

X_{ijk} equal to 1 if job J_j is processed at the k^{th} position of machine M_i, 0 otherwise
Y_{ik} equal to 1 if maintenance is conducted after the k^{th} position of machine M_i, 0 otherwise

Mathematical model

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

$$min. \quad TT = \sum_{j=1}^{n} T_i \tag{2}$$

$$\sum_{j=1}^{n} X_{ijk} \le 1, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 1,2,3,\dots,n \end{cases} \tag{3}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{n} X_{ijk} = 1, \forall\, j = 1,2,3,\dots,n \tag{4}$$

$$Y_{in} = 0, \forall\, i = 1,2,3,\dots,m \tag{5}$$

$$ST_{ik} \ge \sum_{j=1}^{n} r_j * X_{ijk}, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 2,3,\dots,n \end{cases} \tag{6}$$

$$ST_{ik} \ge CT_{i(k-1)} + mt * Y_{i(k-1)}, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 2,3,\dots,n \end{cases} \tag{7}$$

$$CT_{ik} \ge ST_{ik} + \sum_{j=1}^{n} p_j * X_{ijk}, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 2,3,\dots,n \end{cases} \tag{8}$$

$$Q_{i1} = \sum_{j=1}^{n} p_j * X_{ij1}, \forall\, i = 1,2,3,\dots,m \tag{9}$$

$$\begin{cases} Q_{i(k-1)} + \sum_{j=1}^{n} p_j * X_{ijk} \le Q_{ik} + L * Y_{i(k-1)} \\ \sum_{j=1}^{n} p_j * X_{ijk} \le Q_{ik} + mt * \left(1 - Y_{i(k-1)}\right) \end{cases}, \quad \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 2,3,\dots,n \end{cases} \tag{10}$$

$$Q_{ik} \le UT, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 1,2,3,\dots,n \end{cases} \tag{11}$$

$$\sum_{j=1}^{n} X_{ijk} \le \sum_{j=1}^{n} X_{ij(k-1)}, \forall \begin{cases} i = 1,2,3,\dots,m \\ k = 2,3,\dots,n \end{cases} \tag{12}$$

$$C_{max} \ge CT_{in}, \forall i = 1,2,3,\dots,n \tag{13}$$

$$C_j \ge CT_{ik} + L * \left(X_{ijk} - 1\right), \forall \begin{cases} i = 1,2,3,\dots,m \\ j = 1,2,3\dots,n \\ k = 1,2,3\dots,n \end{cases} \tag{14}$$

$$\begin{cases} T_j \ge C_j - d_j \\ T_j \ge 0 \end{cases}, \forall j = 1,2,3,\dots,n \tag{15}$$

In this model, the objectives are to minimize C_{max} and $\sum T_j$ as shown in Eq. (1) and Eq. (2). Constraints (3) and (4) ensure that each job can only be processed at one position of one machine and that each position of one machine can only be occupied by one job. Constraint (5) specifies that no PM is required after the last position of each machine. Constraints (6) and (7) limit the start time of each machine at each position. Constraint (8) defines the completion time of each machine at each position. Constraint (9) describes the cumulative processing time of each machine at the first position. Constraint (10) defines the cumulative processing time of each machine at each position. Constraint (11) ensures that the cumulative processing time of each machine at each position is not larger than UT. Constraint (12) ensures that the job is arranged in a more forward position.

Constraint (13) defines the maximum completion time. Constraint (14) defines the completion time of each job. Constraint (15) defines the tardiness of each job.

5. M-NSGA-II algorithm

For the proposed problem, an M-NSGA-II algorithm is presented by incorporating three strategies. (i) A decoding method based on dynamic programming (DP) is constructed. (ii) Dynamic probability crossover and mutation are designed. (iii) Neighbourhood search method is proposed. The general operation of the M-NSGA-II algorithm is shown in Algorithm 1.

Algorithm 1:M-NSGA-II algorithm
1 Create an initial population P_o of size $Popsize$ using a random rule
2 Decode the individual using the method based on DP, sort the population based on the fast-non-dominated sorting method
3 Create offspring population Q_o by applying the dynamic probability crossover and mutation operators as well as elitist operators
4 **While** stopping criteria are not verified, do
Create population $R_t = P_t \cup Q_t$ of size $2Popsize$ and construct the different F_i of R_t using the fast-non-dominated sorting procedure
Put $P_{t+1} = \emptyset$ and $i = 0$
While $|P_{t+1}| + |F_i| < Popsize$ do
$P_{t+1} = P_{t+1} \cup F_i$
$i = i + 1$
End While
Include in P_{t+1} the $(Popsize - |P_{t+1}|)$ of F_i according to the crowding procedure
Renew population P_{t+1} with neighbourhood search method
Create offspring population Q_{t+1} by selection, crossover, and mutation with dynamic probability
End While

5.1. Chromosome representation

According to the decision of the proposed problem, the presentation of the chromosome is a vector with length $n + m - 1$, where n is the number of jobs, and m is the number of machines. An example is $J_{1[1]}, \cdots, J_{1[n_1]}, 0, \cdots, 0, \cdots, 0, J_{m[1]}, \cdots, J_{m[n_m]}$, where the jobs allocated to different machines are distinguished by 0, and the string of numbers divided by 0 represents the sequence of the jobs processed on one machine.

5.2. Decoding method

For the solution $\pi = \{J_{1[1]}, \cdots, J_{1[n_1]}, 0, \cdots, 0, \cdots, 0, J_{m[1]}, \cdots, J_{m[n_m]}\}$, although the job allocation and sequence on each machine have been determined according to the position of 0 in the chromosome, the maintenance or the job-grouping batch decision still needs to be made. For the identical PMSP without considering the job's release time to minimize C_{max}, a near-optimal solution can be obtained by a maintenance decision made based on the FL principle as shown in Appendix 1. However, it is not true for the same problem considering the job release times. An example with five jobs on a single machine is used to explain, and the jobs data is shown in Table 1. Assume the solution $\pi = \{J_1, J_2, J_5, J_4, J_3\}$, $UT = 10$, and $mt = 2$. The Gantt chart of the solution obtained based on the FL and non-FL principle is shown in Fig. 2. The solution obtained without considering the FL is better than that obtained with the FL. The reason is that machine have to be idle for a period to be fully loaded. For the identical PMSP considering the job release times, the maintenance decision is a dynamic decision, and we propose a decoding method based on dynamic programming (DP) process (Zhou et al., 2018)). In each stage, only the different job-grouping batch schemes of a newly added job and the currently sorted job are compared, and the best optimal solution is selected as the final decoding result.

Table 1
Basic information for 5 jobs example

Job	J_1	J_2	J_3	J_4	J_5
r_j	1	1	9	0	9
p_j	2	2	7	5	5
d_j	3	4	28	17	16

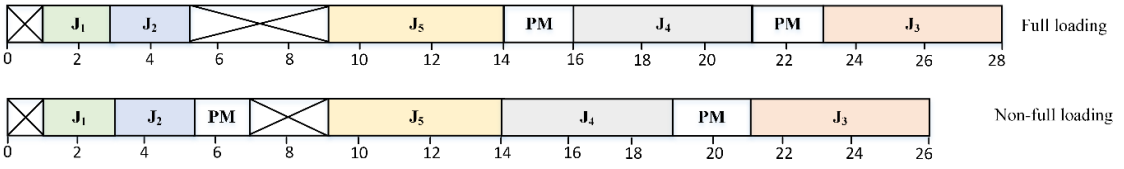

Fig. 2. Solution obtained by different decoding methods for the example

To decode the solution based on the DP method, the individual π is first divided into m segments, representing m machines, and then the solution is decoded for each machine by the Algorithm 2. When decoding the job sequence on machine M_i, the set of the first k jobs is $\Delta^k = \{J_{[1]}, \ldots, J_{[k]}\}$, and the last g jobs in the set Δ^k are $\nabla_g^k = \{J_{[k-g+1]}, J_{[k-g+2]}, \ldots, J_{[k]}\}$. $Z_0(\Delta^k)$ and $Z_1(\Delta^k)$ represent the makespan and total tardiness of k jobs in a given sequence, respectively, so $\Delta^0 = \{\emptyset\}, Z_0(\Delta^0) = 0, Z_1(\Delta^0) = 0$. The makespan and the total tardiness of k sequenced jobs are represented by the state function of the k^{th} stage as shown in Eq. (16); the calculation method for the makespan and total tardiness is shown in Eq. (17) and Eq. (18).

$$Z(\Delta^k) = \left(Z_0(\Delta^k), Z_1(\Delta^k)\right) = \min_{1 \le g \le k} \left\{ f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right), f_g^1\left(\nabla_g^k, Z_1(\Delta^{k-g})\right) \right\} \tag{16}$$

$$f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) = \begin{cases} \max\{Z_0(\Delta^0), r_{[k]}\} + p_{[k]} & g = 1 \text{ and } k = g \\ \max\{Z_0(\Delta^{k-g}) + mt, r_{[k]}\} + p_{[k]} & g = 1 \text{ and } k > g \\ \max\left\{ f_{g-1}^0\left(\nabla_{g-1}^{k-1}, Z_0(\Delta^{k-g})\right), r_{[k]} \right\} + p_{[k]} & g > 1 \text{ and } \sum_{\rho \in \nabla_g^k} p_\rho \le UT \\ \infty & g > 1 \text{ and } \sum_{\rho \in \nabla_g^k} p_\rho > UT \end{cases} \tag{17}$$

$$f_g^1\left(\nabla_g^k, Z_1(\Delta^{k-g})\right) = \begin{cases} Z_1(\Delta^{k-g}) + \max\left\{ f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) - d_{[k]}, 0 \right\} & g = 1 \\ f_{g-1}^1\left(\nabla_{g-1}^{k-1}, Z_1(\Delta^{k-g})\right) + \max\left\{ f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) - d_{[k]}, 0 \right\} & g > 1 \text{ and } \sum_{\rho \in \nabla_g^k} p_\rho \le UT \\ \infty & g > 1 \text{ and } \sum_{\rho \in \nabla_g^k} p_\rho > UT \end{cases} \tag{18}$$

Algorithm 2: DP decoding algorithm

0 Initialize the completion time set of machines $MC = \{\emptyset\}$, $\Delta^0 = \{\emptyset\}$, $Z_0(\Delta^0) = 0, Z_1(\Delta^0) = 0$, UT, and mt
1 For chromosome $h = 1$ to $Popsize$ **do**
2 **For** machine $i = 1$ to m **do**
3 **For** the machine's job position $k = 1$ to n_i, $\Delta^k = \{J_{[1]}, J_{[2]}, \ldots, J_{[k]}\}$
4 **For** $g = 1$ to k, $\nabla_g^k = \{J_{[k-g+1]}, J_{[k-g+2]}, \ldots, J_{[k]}\}$
// Compute the makespan and total tardiness
If $g = 1$
If $k = g$, then $Z_0(\Delta^k) = f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) = \max\{Z_0(\Delta^0), r_{[k]}\} + p_{[k]}$
Else if k > g, then
$$Z_0(\Delta^k) = f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) = \max\{Z_0(\Delta^{k-g}) + mt, r_{[k]}\} + p_{[k]}$$
$$Z_1(\Delta^k) = f_g^1\left(\nabla_g^k, Z_1(\Delta^{k-g})\right) = Z_1(\Delta^{k-g}) + \max\left\{ f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) - d_{[k]}, 0 \right\}$$
If $g > 1$
If $\sum_{\rho \in \nabla_g^k} p_\rho \le UT$, then
$$Z_0(\Delta^k) = f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) = \max\left\{ f_{g-1}^0\left(\nabla_{g-1}^{k-1}, Z_0(\Delta^{k-g})\right), r_{[k]} \right\} + p_{[k]}$$
$$Z_1(\Delta^k) = f_g^1\left(\nabla_g^k, Z_1(\Delta^{k-g})\right) = f_{g-1}^1\left(\nabla_{g-1}^{k-1}, Z_1(\Delta^{k-g})\right) + \max\left\{ f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) - d_{[k]}, 0 \right\}$$
Else if $\sum_{\rho \in \nabla_g^k} p_\rho > UT$, then
$Z_0(\Delta^k) = f_g^0\left(\nabla_g^k, Z_0(\Delta^{k-g})\right) = \infty$
$Z_1(\Delta^k) = f_g^1\left(\nabla_g^k, Z_1(\Delta^{k-g})\right) = \infty$
5 **End For**
6 **End For**
7 $C_i = Z_0(\Delta^k)$, $MC = MC \cup \{C_i\}$, $T_i = Z_1(\Delta^k)$
8 End For
9 $C_{max} = \max\{C_i, C_i \in MC\}$, $TT = \sum_{i=1}^m T_i$
10 End For

5.3. Neighbourhood Search operators

For the identical PMSP, it is easy to generate redundant solutions in the population because of the symmetry of the machines. For example, the solution $\pi_1 = \{J_4, J_3, J_6, 0, J_1, J_2, J_5\}$ is equal to the solution $\pi_2 = \{J_1, J_2, J_5, 0, J_4, J_3, J_6\}$. Too many redundant

solutions will lead to premature convergence of the algorithm. To improve the diversity of the population, we firstly need to identify the redundant solution, then to renew it with a neighbourhood search method. If chromosome q_1 and q_2 in the population is different, then the difference between the two chromosomes is $d(q_1, q_2) \succ 0$, otherwise $d(q_1, q_2) = 0$. If the difference between chromosome q_q and other chromosomes in the population is equal to 0, the solution corresponding to chromosome q_q is redundant solution. Because each chromosome is composed of m segments, firstly the lengths of the m segments are compared. If they are different, the chromosomes are different; otherwise, the difference is determined by comparing the genes. The Eq. (19) and Eq. (20) respectively give the calculation method of the difference of each segment and the difference of the whole chromosome between two chromosomes q_1 and q_2 with the same length.

$$d(q_{1i}, q_{2i'}) = \frac{\sum_{t=1}^{n_i} |g_{1it} - g_{2i't}|}{n_i} \tag{19}$$

$$d(q_1, q_2) = \frac{\sum_{i=1}^{m} d(q_{1i}, q_{2i'})}{n} \tag{20}$$

where q_{1i} and $q_{2i'}$ is gene segments on chromosomes q_1 and q_2, respectively; n_i is the length of the gene segment, and g_{1it} and $g_{1i't}$ is the locus in the gene segment.

Let N_1 represent the number of redundant solutions in the population, and dw represents the density of the redundant solutions; then, $dw = N_1/Popsize.$ If the dw is larger than the threshold, a neighbourhood search method is used to renew the individuals in the redundant solutions population. If the new solution is better than the original solution, it replaces the original solution; otherwise, the original solution is kept.

Because the two objectives are optimized, two kinds of four neighbourhood search operators are designed. The average makespan $\bar{C}$ of the maximum makespan $\mathrm{MaxC_{max}}$ and the minimum makespan $MinC_{max}$, the average $\sum \bar{T}$ of the maximum total tardiness $Max \sum T_j$, and the minimum total tardiness $Min \sum T_j$ in the redundant solution's population are calculated firstly, then the neighbourhood search operators are used.

5.3.1 Neighbourhood search guided by makespan improvement

If the objective of the k^{th} individual satisfies $C_k > \bar{C}$, one of the two M-neighbourhood search methods is randomly selected. The purpose of the M-neighbourhood search method is to improve the C_{max} of the solution, including M-Insert and M-Swap.

M-Insert: Select a batch with waste time p_{wt} from machine M_{minct} with the minimum completion time, search for a job with a processing time equal to or less than p_{wt} in machine M_{maxct} with the maximum completion time, and insert it into the selected batch on machine M_{minct}.

M-Swap: The job with a longer processing time selected from machine M_{maxct} is swapped with the job with a shorter processing time selected from machine M_{minct}. The difference in processing time between the swapped jobs should be less than the difference in the completion time of the two machines.

5.3.2 Neighbourhood search guided by total tardiness improvement

If the objective of the k^{th} individual satisfies $\sum T_k > \sum \bar{T}$, one of the two T-neighbourhood search methods is randomly selected. The purpose of the T-neighbourhood search method is to improve the $\sum T_j$ of the solution, including T-Insert and T-Swap.

T-Insert: Select the job with the shortest processing time from among the delayed jobs and insert them one by one into the position before the job with a larger due date time.

T-Swap. Select the job with the longest processing time from among the delayed jobs and swap its position with the job whose subsequent due date is smaller than the job one by one.

5.4. Genetic operators

5.4.1 Crossover operator

A fixed crossover probability $p_c = 0.5$ is set for the NSGA-II algorithm, and a dynamic crossover probability $\mathrm{p_c'} = 1.5 - \frac{2\mathrm{e}^{-\frac{\mathrm{g}}{\mathrm{G}}}}{1+\mathrm{e}^{-\frac{\mathrm{g}}{\mathrm{G}}}} \times \mathrm{p_c}$ is designed for the M-NSGA-II algorithm. g represents the number of iterations at present, and G represents the total number of iterations to be performed.

A two-point crossover method is adopted. A random chromosome p_x in the first Pareto front $\mathrm{F_1}$ of the population after fast non-dominated sorting and a random chromosome p_y in the other front are selected as parents. A random gene in $[1, n + m - 1]$ is selected from each parent for crossover, and the other genes are obtained by mapping. If the offspring is better than the parent p_y, it will be retained. Otherwise, the original parent is retained. An example of crossover is shown in Fig. 3.

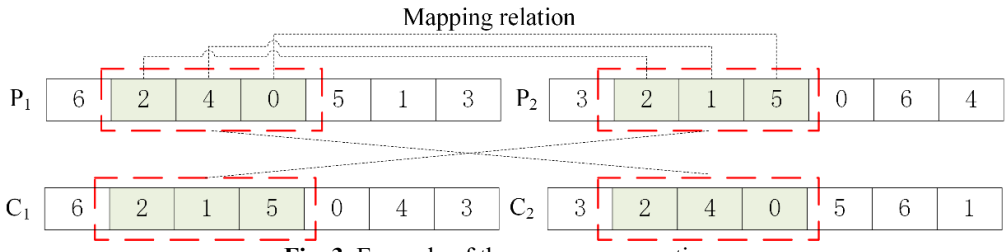

Fig. 3. Example of the crossover operation

5.4.2 Mutation operator

A fixed mutation probability $p_m = 0.05$ is set for the NSGA-II algorithm, and a dynamic mutation probability $p'_m = \frac{2e^{-\frac{g}{G}}}{1+e^{-\frac{g}{G}}} \times p_m$ is designed for the M-NSGA-II algorithm. Two methods, MO_1 and MO_2, are used for the mutation operation of the first 50% generation population and the last 50% generation population, respectively. MO_1 randomly selects an individual, selects a gene fragment of length $\lfloor n/m \rfloor$, and reverses the sequence of jobs in this gene fragment. MO_2 selects an individual and randomly selects two genes to exchange. If the randomly selected gene is 0, reselection is required. An example of the mutation operation is shown in Fig. 4.

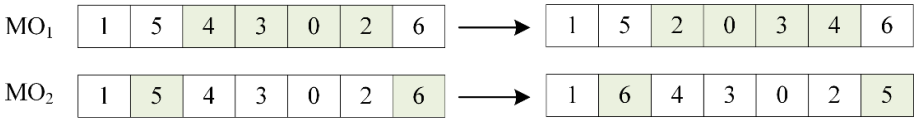

Fig. 4. Example of the mutation operation

6. Parameter tuning and computational experiments

The NSGA-II and the M-NSGA-II algorithms were written using the Dev C++ language, and the MIP model was constructed on the IBM ILOG CPLEX Optimization Studio Ver. 12.7.1 platform. The three algorithms' simulation experiments were all run on an Intel(R) Xeon(R) E5-2650 v4 @2.20 GHz CPU and 64 GB RAM workstation.

6.1. Performance measures

This paper compares different algorithms in terms of operation efficiency and effectiveness. The operation efficiency is directly expressed by the operating time. Four measures are used to compare the operation effectiveness or different aspects of the non-dominated fronts obtained by MIP, the NSGA-II algorithm and the M-NSGA-II algorithm, including the number of Pareto solutions N_d, the C metric, the inverted generational distance (IGD) and the distance Δ_{D1} between the different solutions. The C metric is used to show differences between two Pareto fronts (Wang & Liu, 2015). The $C(A,B)$ states the percentage of a solution in B dominated by at least one solution of A. It is calculated by Eq. (21):

$$C(A,B) = \frac{|\{b \in B | \exists a \in A: a < b\}|}{|B|} \tag{21}$$

where $a < b$ means that a dominates b or b is dominated by a. It can be seen from Equation (21) that if all individuals in solution set A dominate all individuals in solution set B, then $C(A,B) = 1$; otherwise, $C(A,B) = 0$. In this metric, $C(A,B) \neq C(B,A)$; thus, both $C(A,B)$ and $C(B,A)$ must be calculated, and A is better than B if $C(A,B) > C(B,A)$.

The IGD is the inverse mapping of the generational distance (Zhang et al., 2021), which is represented by the average distance from the individual in the Pareto solution set to the non-dominated solution set solved by the algorithm. The computing formula is shown in Eq. (22):

$$IGD(A,\Omega) = \frac{\sum_{x \in \Omega} \min_{a \in A} d_{xa}}{|\Omega|} \tag{22}$$

where d_{xa} is the Euclidean distance between solutions x and a, and Ω is the approximate Pareto front, and is acquired by merging the Pareto solutions obtained by all algorithms. The smaller the $IGD(A,\Omega)$ is, the better of the algorithm.

Δ_{D1} is used in the literature (Wang et al., 2018), and it is defined by Eq. (23):

$$\Delta_{D1}=\sum_{z=1}^{|A|-1}\frac{|d_z-\bar{d}|}{|A|-1} \tag{23}$$

where d_z is the Euclidean distance between two consecutive solutions in the obtained non-dominated set A, and $\bar{d}$ is the average of all distances d_z. The smaller the Δ_{D1} is, the better of the algorithm.

6.2. Parameter tuning

Parameters largely affect the performance of the M-NSGA-II algorithm. Therefore, it is necessary to tune the parameters using appropriate methods. The Taguchi method is a strong stochastic technique from among the design of experiments (DOE) methods. It aims to improve quality and reduce cost when designing a process or production, and its main idea is to combine different factors and their degrees using vertical arrays and factorial designs. Then, the analysis is conducted with fewer numbers of experiments (Wang et al., 2018). The parameters that affect the performance of the M-NSGA-II algorithm include the population number Popsize, the number of iterations Gensize, the crossover rate p_c, and the mutation rate p_m. The levels of these four parameters were determined according to the preliminary experiments as shown in Table 2.

Table 2
M-NSGA-II algorithm parameters and levels

Parameters	Small	Medium	Large
Popsize	50,75,100	100,200,300	200,300,400
Gensize	50,75,100	100,150,200	150,200,250
p_c	0.5,0.7,0.9	0.5,0.7,0.9	0.5,0.7,0.9
p_m	0.01,0.05,0.10	0.05,0.1,0.15	0.1,0.15,0.2

A set of 13 typical problem instances with three scales (small, medium, and large) were generated to perform the parameter tuning experiments, and a decoding method based on FL was adopted. The signal-to-noise (S/N) ratio of the parameters was analysed to determine the parameters of the M-NSGA-II algorithm. Using Popsize, MaxGen, p_c and p_m as control factors where each factor has 3 levels, the number of orthogonal experiments is $L_9(3^4)$, the objective function is used as the response variable, and the number of experiments is 13×9×10=1170. The results for the small-scale problem instances are shown in Table 3. Because there are two objectives, the same data normalization method used in reference (Yue et al., 2019) is applied. Through normalization of the S/N at different levels of different response variables, the comprehensive S/N ratio at the final parameter level was obtained, and Fig. 5 shows the mean S/N for the small-scale problems.

Table 3
Orthogonal results for the small-scale instances

No.	Popsize	Gensize	p_c	p_m	m2n6(10,5)		m2n8(10,5)		m2n10(10,5)		$\overline{C_{max}}$	$\overline{TT}$
					C_{max}	TT	C_{max}	TT	C_{max}	TT		
1	50	50	0.5	0.01	24.017	7.983	30.880	9.640	38.708	25.483	31.202	14.369
2	50	75	0.7	0.05	24.033	7.567	30.733	9.200	38.992	28.300	31.253	15.022
3	50	100	0.9	0.1	24.067	8.200	30.800	10.825	38.700	26.592	31.189	15.206
4	75	50	0.7	0.1	24.167	7.633	30.850	11.650	38.668	28.573	31.228	15.952
5	75	75	0.9	0.01	24.233	7.767	30.700	9.258	39.058	26.333	31.331	14.453
6	75	100	0.5	0.05	23.967	8.117	31.067	11.867	39.150	27.737	31.394	15.907
7	100	50	0.9	0.05	24.033	7.533	30.700	10.067	38.705	25.762	31.146	14.454
8	100	75	0.5	0.1	24.333	7.533	30.767	9.875	39.145	27.940	31.415	15.116
9	100	100	0.7	0.01	24.242	7.775	30.783	10.917	38.945	28.135	31.323	15.609

Due to the two objectives of the minimization functions, the level of the parameter at which the minimum value of the evaluation criteria is obtained is preferred. The parameters of NSGA-II for different sets of problem instances were also selected using the Taguchi method. The optimum level of parameters for each algorithm for each instance set is illustrated in Table 4.

Table 4
Parameter values for M-NSGA-II and NSGA-II

	M-NSGA-II				NSGA-II			
Instance	$Popsize$	$Gensize$	p_c	p_m	$Popsize$	$Gensize$	p_c	p_m
Small	50	50	0.9	0.05	50	50	0.9	0.05
Medium	200	200	0.9	0.15	200	200	0.9	0.15
Large	200	250	0.9	0.1	200	250	0.9	0.1

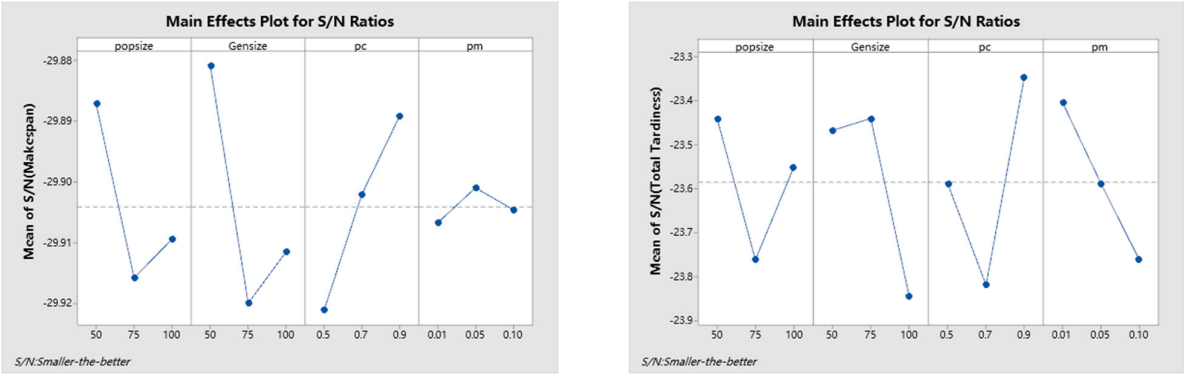

Fig. 5. The mean S/N results for the bi-objective of makespan and TT of small-scale problems

6.3. Computational experiments and discussion

6.3.1 Data generation

The experimental data in this paper are based on the data in the literature (Kramer et al. (2020), Nessah et al. (2008), Chen et al. (2021) and Li et al. (2019)). The experiments were conducted using the machine parameters, job parameters and PM activity parameters. The machine parameters include the number of machines m; the job parameters include the number of jobs n, processing time p_j, release time r_j and due date d_j; and the PM activity parameters include the threshold UT and maintenance time mt. The instances and the range of experimental parameter are shown in Table 5.

Table 5
Instances and experimental parameters

Instances	m	n	p_j	UT	mt
Small	2, 3	6,8,10	U[1,9]	10	2, 5
Medium	3, 5	15,20,25	U[1,9]	10, 15	5, 8
Large	5, 10	50,100	U[1,9]	15, 20	5, 8

The due date d_j satisfies the uniform distribution $U[r_j, r_j + \frac{(1+Q-C)\sum_{j=1}^{n} p_j}{m}]$, where $Q = [0.4,0.5]$ and $C = [0.4,0.3]$. The release time r_j of the job is set to obey a uniform distribution $U[0, k \times n \times \alpha/m]$, where $k = [2.02,3.03]$ and $\alpha = [0.4,0.5,0.6]$.

6.3.2 Comparison results for different decoding methods

To validate the performance of the decoding method based on DP, we solved the problem instances using the DP-based and FL-based decoding methods. For each instance, the method was run 10 times by considering randomness, and the average values of the performance metrics were recorded. The fronts obtained by the two methods mentioned above were combined, and all the non-dominated solutions were used to form set Ω to compute the IGD. The results are shown in Table 6.

Table 6
Results of the different decoding methods of the M-NSGA-II algorithm

Size	Instances	FL: A					DP: B				
	m, n (UT, mt)	t(s)	N_d	C(A,B)	IGD	Δ_{p1}	t(s)	N_d	C(B,A)	IGD	Δ_{p1}
Small	m2n6(10,2)	**2.20**	2	0.63	1.30	**0.00**	2.43	**2.3**	**0.78**	**0.20**	1.42
	m2n8(10,2)	**3.30**	**2.4**	0.57	0.95	1.15	4.50	2.3	**0.78**	**0.69**	**0.59**
	m2n10(10,2)	**3.99**	**2.4**	**0.62**	**1.71**	3.84	6.31	1.7	0.45	3.30	**0.50**
	m3n6(10,2)	**2.52**	2.1	0.92	0.19	**0.46**	2.77	**2.3**	**1.00**	**0.00**	0.46
	m3n8(10,2)	**3.16**	2.1	0.65	0.67	**0.17**	4.86	**2.3**	**0.85**	**0.46**	0.92
	m3n10(10,2)	**4.07**	**2.8**	0.49	**1.23**	**1.96**	5.89	2.6	**0.62**	1.34	2.54
	m2n6(10,5)	**2.52**	**3.0**	0.93	0.18	2.54	4.77	**3.0**	**0.97**	**0.03**	**2.41**
	m2n8(10,5)	**3.14**	3.0	**0.87**	**0.28**	1.14	5.49	**3.1**	0.58	1.18	**1.00**
	m2n10(10,5)	**3.44**	**3.5**	0.30	1.96	**2.07**	7.79	3.3	**0.55**	**0.76**	3.13
	m3n6(10,5)	**1.94**	**3.0**	0.83	0.36	1.90	2.65	**3.0**	**1.00**	**0.00**	**0.97**
	m3n8(10,5)	**2.83**	**3.5**	0.63	**0.57**	2.17	5.11	3.1	**0.75**	0.83	**1.44**
	m3n10(10,5)	**4.37**	**3.0**	0.48	1.40	**2.04**	7.59	**3.0**	**0.75**	**0.47**	2.31

Table 6
Results of the different decoding methods of the M-NSGA-II algorithm (Continued)

Size	Instances	FL: A					DP: B				
	m, n (UT, mt)	t(s)	N_d	C(A,B)	IGD	Δ_{D1}	t(s)	N_d	C(B,A)	IGD	Δ_{D1}
Medium	m3n15(10,5)	**107.97**	2.2	0.33	3.26	**3.07**	185.60	**2.8**	**0.48**	**1.27**	3.84
	m3n20(10,5)	**156.04**	**2.5**	**0.57**	2.49	**1.51**	261.42	2.3	0.25	**2.46**	2.11
	m3n25(10,5)	**190.18**	1.8	0.36	4.41	**0.22**	367.29	**2.4**	**0.45**	**1.92**	7.85
	m5n15(10,5)	**112.02**	2	0.45	**1.16**	3.90	151.67	**2.3**	**0.52**	1.32	**3.56**
	m5n20(10,5)	**166.34**	**2**	0.30	3.54	**1.89**	219.32	1.9	**0.75**	**0.48**	2.71
	m5n25(10,5)	**196.62**	2.4	0.26	4.37	2.77	318.34	**2.6**	**0.65**	**1.29**	**2.63**
	m3n15(10,8)	**134.90**	**2.2**	0.37	2.14	**2.36**	172.75	2.1	**0.62**	**1.69**	2.76
	m3n20(10,8)	**182.52**	**2.4**	0.20	5.88	4.15	248.46	1.7	**0.74**	**2.95**	**1.94**
	m3n25(10,8)	**194.37**	**2.5**	0.12	8.04	6.52	326.17	2	**0.68**	**5.04**	**2.92**
	m5n15(10,8)	**115.44**	2	0.50	1.30	**2.30**	124.12	**2.2**	**0.67**	**0.87**	2.88
	m5n20(10,8)	**160.18**	1.9	0.28	4.50	1.94	203.88	**2.8**	**0.60**	**0.95**	**1.82**
	m5n25(10,8)	**213.20**	**2.7**	0.28	5.16	**3.39**	249.11	2.1	**0.50**	**2.69**	10.22
	m3n15(15,5)	**108.43**	**2.7**	0.35	1.77	3.15	160.02	2.6	**0.61**	**1.48**	**1.81**
	m3n20(15,5)	**153.16**	2	0.34	1.86	**3.24**	207.99	**2.7**	**0.60**	**1.39**	4.60
	m3n25(15,5)	**163.92**	**2.7**	0.23	5.54	10.15	350.90	2.5	**0.75**	**1.18**	**1.63**
	m5n15(15,5)	122.81	**2.1**	0.47	1.38	**1.99**	**122.67**	1.6	**0.83**	**0.42**	3.14
	m5n20(15,5)	**149.96**	**2.3**	0.32	1.55	1.34	220.55	1.8	**0.55**	**0.93**	**1.08**
	m5n25(15,5)	**207.96**	**2.5**	0.50	**1.68**	2.27	302.78	2	**0.55**	2.38	**0.49**
	m3n15(15,8)	**33.90**	**3.2**	0.53	1.53	**3.53**	78.46	**3.2**	**0.56**	**1.18**	3.69
	m3n20(15,8)	**43.72**	3.5	0.37	4.75	3.51	117.30	**3.8**	**0.48**	**3.22**	**3.49**
	m3n25(15,8)	**50.59**	3.1	0.39	4.69	8.92	170.31	**3.4**	**0.47**	**3.78**	**2.66**
	m5n15(15,8)	**23.44**	**2.8**	**0.73**	**0.68**	2.59	58.59	2.7	0.23	1.76	**0.85**
	m5n20(15,8)	**39.32**	**3.2**	0.43	3.74	**1.87**	109.28	3.1	**0.58**	**0.93**	3.93
	m5n25(15,8)	**49.01**	3.2	0.42	3.54	3.28	132.92	**3.3**	**0.60**	**2.72**	**2.29**
Large	m5n50(15,5)	**408.28**	2.3	0.40	**13.07**	**4.44**	656.13	**3.7**	**0.50**	13.72	7.59
	m5n100(15,5)	**717.37**	**4.5**	0.38	70.35	**12.63**	1762.36	3.4	**0.45**	**61.54**	26.97
	m10n50(15,5)	**413.37**	1.8	0.33	13.80	12.60	535.59	**2**	**0.50**	**2.21**	**6.15**
	m10n100(15,5)	**809.03**	**2.7**	0.38	29.97	**6.33**	1222.25	2.2	**0.59**	**21.71**	21.09
	m5n50(15,8)	**350.95**	2.3	0.23	16.11	**11.10**	663.09	**3.4**	**0.40**	**10.83**	16.26
	m5n100(15,8)	**674.02**	**3.5**	0.26	142.49	48.29	1749.16	2.8	**0.55**	**37.95**	**23.92**
	m10n50(15,8)	**439.74**	**2.0**	0.32	12.52	**4.00**	562.49	1.8	**0.48**	**4.39**	11.97
	m10n100(15,8)	**757.55**	2.1	0.35	67.22	**3.86**	1205.37	**3.8**	**0.49**	**25.90**	23.03
	m5n50(20,5)	**419.26**	**3.2**	0.42	17.77	4.04	679.66	2.4	**0.45**	**5.37**	**3.92**
	m5n100(20,5)	**709.87**	3.6	0.27	112.63	17.48	1797.70	**4.3**	**0.67**	**21.01**	**15.57**
	m10n50(20,5)	**385.11**	1.6	0.30	5.64	**0.00**	597.70	**2.1**	**0.60**	**3.88**	0.67
	m10n100(20,5)	**788.78**	**3**	0.40	28.40	**3.49**	1237.99	2.2	**0.48**	**20.64**	7.12
	m5n50(20,8)	**249.44**	**3.7**	0.25	9.52	5.77	660.86	2.5	**0.62**	**6.00**	**3.61**
	m5n100(20,8)	**520.18**	**4.6**	0.26	145.98	29.14	1772.32	3.7	**0.70**	**16.28**	**24.92**
	m10n50(20,8)	**177.79**	**3.3**	0.33	6.86	**1.38**	548.57	2.6	**0.56**	**2.27**	3.15
	m10n100(20,8)	**485.64**	**4**	0.25	67.18	16.72	1225.39	3.4	**0.70**	**13.09**	**7.35**

The results show that the DP-based decoding method performs better than the FL-based decoding method in terms of the C metric and IGD, especially for large-scale problem instances composed of many jobs. There is no significant difference between the two decoding methods in terms of the Δ_{D1} metric. The N_d metric obtained by the FL-based decoding method is slightly larger than that obtained by the DP-based decoding method. This result shows that the solution set obtained by the DP-based decoding method is more diversity and better approximates the Pareto front. In this paper, the N_d metric obtained by the NSGA-II and M-NSGA-II algorithms is limited since minimizing the makespan indirectly leads to minimizing the total tardiness, which is why the N_d metric of the FL-based decoding method slightly outperforms the DP-based decoding method. The DP-based decoding method searches for the job-grouping solution from the first position to the current position when adding a job at each stage to make maintenance decisions to obtain the best solution, which also limits the number of Pareto solutions. Thus, the DP-based decoding method requires more time than the FL-based decoding method.

6.3.3 Comparison results for different neighbourhood search methods

To validate the performance of the proposed neighbourhood search method, the M-NSGA-II algorithm with and without the neighbourhood search method was used to solve the instances. The results are shown in Fig. 6. The results show that the M-NSGA-II method with neighbourhood search can obtain better non-dominated solutions compared to the method without neighbourhood search in terms of the C metric and IGD. This advantage becomes more evident as the size of the problem increases. However, there is no significant difference between the two methods in terms of the Δ_{D1} metric and N_d. Since neighbourhood search adds further exploration and searching of the solution space, the time required is longer than that without neighbourhood search. These results fully show that the use of neighbourhood search can improve the search quality of the solution, improve the distribution and quality of the solution, and requires a longer time to obtain a better solution.

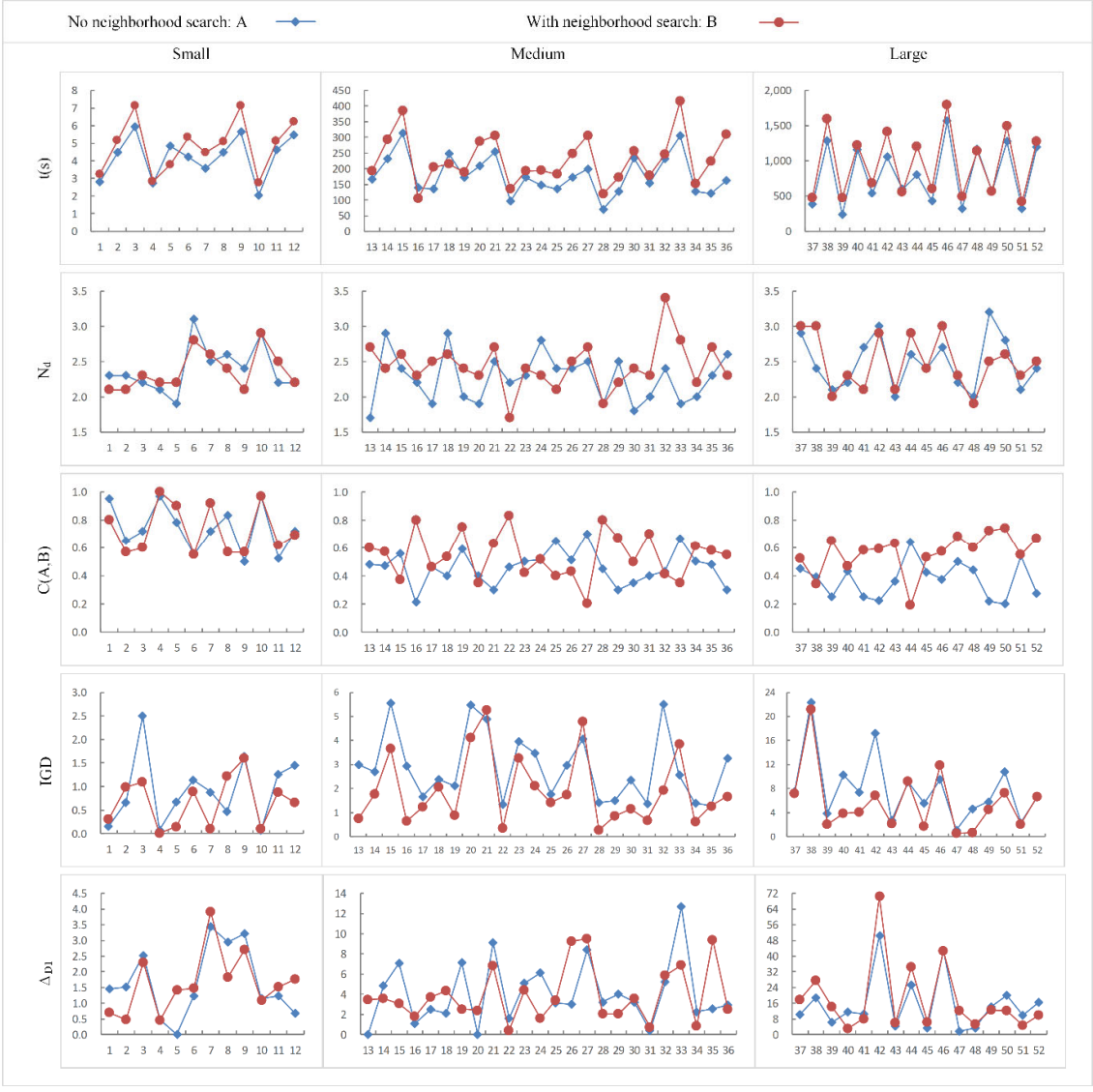

Fig. 6. Results for five performance metrics with and without neighbourhood search method

6.3.4 Comparison results for different algorithms

To compare the performance of MIP, M-NSGA-II, and NSGA-II, all three methods were used to solve the small-scale instances, and the M-NSGA-II and NSGA-II algorithms were used to solve the medium- and large-scale instances. The results are shown in Table 7 and Table 8.

Table 7
Performance results for different algorithms (small-scale problem instances)

Size	Instance	MIP: A						NSGA-II: B						M-NSGA-II: C					
		t(s)	N_d	C(A,B)	C(A,C)	IGD	Δ_{p1}	t(s)	N_d	C(B,A)	C(B,C)	IGD	Δ_{p1}	t(s)	N_d	C(C,A)	C(C,B)	IGD	Δ_{p1}
Small	m2n6 (10,2)	5.3	**2.2**	**1**	**1**	**0**	**0.30**	**0.88**	2.1	0.72	0.80	0.84	1.35	3.62	**2.2**	0.62	0.63	0.72	0.75
	m2n8 (10,2)	32.4	2.4	**1**	**1**	**0**	2.20	**1.64**	**2.6**	0.45	0.71	1.14	**1.49**	4.68	2.5	0.37	0.58	1.32	2.08
	m2n10 (10,2)	4096.0	**2.9**	**1**	**1**	**0**	**3.02**	**2.76**	2.6	0.28	0.73	2.56	4.64	6.28	2.4	0.15	0.48	3.13	6.19
	m3n6 (10,2)	7.2	**2.4**	**1**	**1**	**0**	0.55	**0.82**	2.4	0.95	1.00	0.05	0.55	2.01	2.1	0.79	0.79	0.41	**0.46**
	m3n8 (10,2)	49.7	**2.7**	**1**	**1**	**0**	1.55	**1.91**	2.4	0.40	0.52	1.15	1.57	3.53	2.2	0.59	0.68	0.96	**0.62**
	m3n10 (10,2)	3449.5	**3.4**	**1**	**1**	**0**	1.62	**2.90**	2.8	0.35	0.58	1.38	**1.34**	6.31	2.5	0.22	0.44	1.97	1.62
	m2n6 (10,5)	10.6	**3**	**1**	**1**	**0**	2.56	**0.91**	2.7	0.77	0.75	1.06	**1.27**	4.46	2.6	0.73	0.70	0.61	3.91
	m2n8 (10,5)	55.0	**2.8**	**1**	**1**	**0**	**1.18**	**1.65**	2.4	0.42	0.67	1.26	1.84	5.08	2.4	0.45	0.63	1.12	1.82
	m2n10 (10,5)	13264.7	**3.4**	**1**	**1**	**0**	**2.29**	**2.76**	3.1	0.32	0.73	2.00	2.33	7.10	2.1	0.22	0.40	4.03	2.71
	m3n6 (10,5)	25.5	3	**1**	**1**	**0**	**1.08**	**0.82**	**3.1**	0.97	1.00	0.03	1.13	2.76	2.9	0.93	0.92	0.15	1.09
	m3n8 (10,5)	210.0	**3.4**	**1**	**1**	**0**	1.70	**1.80**	3	0.58	0.67	1.06	2.17	5.10	2.5	0.36	0.47	1.69	**1.51**
	m3n10 (10,5)	15269.2	**3.2**	**1**	**1**	**0**	2.45	**3.22**	2.5	0.35	0.57	2.77	2.19	6.19	2.2	0.22	0.45	3.30	**1.76**

Table 8

Performance results for different algorithms (medium- and large-scale problem instances)

Size	Instance	NSGA-II: A					M-NSGA-II: B				
		t(s)	N_d	C(A, B)	IGD	Δ_{D1}	t(s)	N_d	C (B, A)	IGD	Δ_{D1}
Medium	m3n15(10,5)	**47.64**	**2.3**	0.25	3.47	**2.00**	206.14	2.2	**0.65**	**0.68**	3.67
	m3n20(10,5)	**71.72**	**2.4**	0.13	6.78	**5.12**	338.95	2.3	**0.50**	**3.27**	9.46
	m3n25(10,5)	**89.84**	2.6	0.30	6.88	12.19	365.60	**2.9**	**0.57**	**4.27**	**5.93**
	m5n15(10,5)	**47.49**	**2.2**	0.47	**1.08**	4.69	94.94	**2.2**	**0.57**	1.87	**1.69**
	m5n20(10,5)	**68.28**	**2.3**	**0.58**	**1.54**	**1.52**	151.99	**2.3**	0.26	2.52	5.31
	m5n25(10,5)	**96.35**	**3.1**	0.22	4.73	**2.68**	269.25	2.6	**0.49**	**2.52**	5.95
	m3n15(10,8)	**45.01**	**2.5**	0.28	4.67	5.82	162.42	2.4	**0.63**	**4.12**	**2.22**
	m3n20(10,8)	**78.37**	2.4	0.03	8.95	3.64	320.89	**2.5**	**0.89**	**0.61**	**2.73**
	m3n25(10,8)	**84.82**	**2.6**	0.25	12.08	7.87	437.04	2.5	**0.55**	**2.91**	**6.35**
	m5n15(10,8)	**52.80**	**2.6**	0.45	**0.76**	**1.77**	144.50	1.8	**0.64**	0.82	2.82
	m5n20(10,8)	**75.58**	**2.8**	0.32	**2.93**	**4.87**	189.48	**2.8**	**0.40**	4.14	7.31
	m5n25(10,8)	**96.21**	2.2	0.18	9.34	**2.33**	278.11	**2.3**	**0.61**	**3.06**	5.30
	m3n15(15,5)	**38.69**	**2.1**	0.30	2.40	8.20	145.51	1.8	**0.68**	**1.12**	**2.48**
	m3n20(15,5)	**66.65**	**2.3**	0.26	2.89	**1.02**	204.21	2.2	**0.53**	**1.53**	4.07
	m3n25(15,5)	**88.78**	**4**	0.21	7.24	**3.57**	248.33	2.7	**0.55**	5.79	6.44
	m5n15(15,5)	**47.68**	1.9	0.63	**0.68**	**2.61**	107.42	1.7	**0.66**	1.38	13.30
	m5n20(15,5)	**67.78**	2.6	0.44	**1.29**	**3.26**	189.53	**2.2**	**0.53**	2.15	3.71
	m5n25(15,5)	**100.24**	2.2	0.28	2.69	**0.80**	253.04	**2.5**	**0.68**	**1.72**	5.01
	m3n15(15,8)	**42.94**	**2.3**	0.00	4.61	4.04	176.97	**2.3**	**0.92**	**0.12**	**0.74**
	m3n20(15,8)	**64.78**	3.1	0.27	5.12	**4.21**	245.88	**3.4**	**0.63**	**2.41**	5.89
	m3n25(15,8)	**89.45**	**2.9**	0.05	9.56	**5.11**	415.90	2.8	**0.81**	**4.14**	6.91
	m5n15(15,8)	**51.59**	**2.2**	0.63	1.54	2.66	151.53	**2.2**	**0.67**	**0.70**	**0.82**
	m5n20(15,8)	**74.89**	2.2	**0.43**	**2.20**	**4.52**	223.01	**2.7**	0.43	2.48	9.35
	m5n25(15,8)	**98.04**	2.1	0.30	**3.22**	4.24	308.37	**2.3**	**0.52**	3.52	**2.50**
Large	m5n50(15,5)	**182.18**	**3.4**	0.03	58.42	**4.88**	396.15	2.7	**0.57**	**22.03**	18.88
	m5n100(15,5)	**441.01**	**3.5**	0.00	285.12	5.78	1591.75	2.2	**0.34**	**269.86**	**3.75**
	m10n50(15,5)	**203.41**	**2.6**	0.10	21.68	4.00	387.27	1.8	**0.61**	**6.31**	**3.85**
	m10n100(15,5)	**462.61**	**2.3**	0.00	190.68	**7.05**	1044.38	2.0	**0.58**	**69.14**	43.41
	m5n50(15,8)	**196.90**	**3.4**	0.05	56.04	23.14	543.69	2.6	**0.45**	**36.85**	**19.83**
	m5n100(15,8)	**394.61**	3.5	0.02	540.18	**44.27**	1241.68	**3.7**	**0.46**	**174.98**	59.58
	m10n50(15,8)	**237.12**	1.9	0.08	13.37	5.28	413.20	**2.4**	**0.23**	**6.24**	**5.16**
	m10n100(15,8)	**514.55**	**3.1**	0.00	174.53	**4.79**	858.93	2.5	**0.51**	**105.91**	19.10
	m5n50(20,5)	**197.17**	**3.0**	0.03	32.86	**2.12**	568.50	2.5	**0.47**	**14.32**	11.03
	m5n100(20,5)	**408.87**	**4.6**	0.00	325.77	27.53	1688.12	2.9	**0.46**	**276.07**	**11.42**
	m10n50(20,5)	**214.53**	2.5	0.03	10.22	**4.91**	321.24	**2.6**	**0.46**	**6.02**	7.74
	m10n100(20,5)	**473.33**	**3.3**	0.00	152.84	8.95	1105.91	1.8	**0.51**	**62.49**	**0.00**
	m5n50(20,8)	**207.05**	**3.3**	0.17	36.43	22.51	560.19	2.5	**0.49**	**18.64**	**12.50**
	m5n100(20,8)	**416.37**	**3.5**	0.00	302.44	**10.38**	1270.85	2.6	**0.29**	**239.52**	12.06
	m10n50(20,8)	**222.82**	2.1	0.10	10.00	**3.23**	316.07	**2.3**	**0.45**	**4.78**	4.77
	m10n100(20,8)	**512.82**	**2.8**	0.00	158.64	**4.25**	1191.68	2.5	**0.33**	**82.77**	9.83

As seen in Table 7, the MIP method can obtain the optimal Pareto solution for all 12 instances, although the time needed for solving the problem greatly increased with the increase in the size of the problem. The NSGA-II algorithm slightly outperforms the M-NSGA-II algorithm in terms of the IGD, the C metric and N_d. There is no significant difference between the two methods in terms of Δ_{D1}. The NSGA-II algorithm is more efficient than the MIP and M-NSGA-II algorithms, and this advantage becomes more evident as the size of the problem increases. It can be seen from Table 8 that the M-NSGA-II algorithm outperforms the NSGA-II algorithm in terms of the IGD and C metrics for medium- and large-scale problem instances, and this advantage also becomes more evident as the size of the problem increases. However, there is no significant difference between the two methods in terms of N_d and Δ_{D1}. The efficiency of the NSGA-II algorithm is better than that of M-NSGA-II because M-NSGA-II uses the DP-based decoding method and the neighbourhood search method.

6.3.5 Influence of PM parameters

To judge the influence of the PM parameters on the NSGA-II and M-NSGA-II algorithms, their four metrics were analysed using variance analysis (the confidence interval was 95%), and the results are shown in Table 9.

As shown in Table 9, for the different UT and mt, all the indicators for the M-NSGA-II algorithm are not significantly different. In contrast, for the NSGA-II algorithm, the N_d indicator for small instances and the IGD indicator for medium instances are significantly different for parameter mt. This result shows that the M-NSGA-II algorithm is stable and effective for all instances.

Table 9
Significance level of the different algorithms

NSGA-II

Size	Small				Medium				Large			
Parameter	N_d	IGD	C(A,B)	Δ_{D1}	N_d	IGD	C(A,B)	Δ_{D1}	N_d	IGD	C(A,B)	Δ_{D1}
UT	/	/	/	/	0.955	0.016	0.563	0.442	0.463	0.171	0.780	0.752
mt	**0.018**	0.548	0.902	0.509	0.955	**0.005**	0.162	0.798	0.4015	0.345	0.212	0.288

M-NSGA-II

Size	Small				Medium				Large			
Parameter	N_d	IGD	C(A,B)	Δ_{D1}	N_d	IGD	C(A,B)	Δ_{D1}	N_d	IGD	C(A,B)	Δ_{D1}
UT	/	/	/	/	1	0.569	0.201	0.872	0.884	0.938	0.527	0.115
mt	0.330	0.256	0.899	0.857	0.123	0.976	0.120	0.368	0.059	0.740	0.111	0.499

7. Conclusions

In this paper, an identical PMSP with machine flexible maintenance and job release times has been investigated to minimize the makespan and total tardiness simultaneously. A MIP model was first established to obtain the exact Pareto fronts for small-scale instances. To tackle the medium- and large-scale instances, an M-NSGA-II algorithm was constructed with three strategies. After tuning the M-NSGA-II parameters using the DOE and the Taguchi method, computational experiments were conducted. The results show that the M-NSGA-II algorithm obviously outperforms the original NSGA-II algorithm when solving medium- and large-scale instances, although the time required to solve these instances is slightly longer.

In future research, more realistic parallel machine manufacturing environments such as unrelated parallel machines should be investigated. The maintenance period depends on the machine's reliability, and the objective is to minimize the energy and power costs. Furthermore, considering the development of Big Data and Intelligent Manufacturing, parallel machine dynamic scheduling and intelligent scheduling methods based on reinforcement learning should be developed.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.51705370).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Abdeljaoued, M. A., Saadani, N. E. H., & Bahroun, Z. (2020). Heuristic and metaheuristic approaches for parallel machine scheduling under resource constraints. *Operational Research, 20*, 2109-2132.

Abedi, M., Seidgar, H., Fazlollahtabar, H., & Bijani, R. (2015). Bi-objective optimisation for scheduling the identical parallel batch-processing machines with arbitrary job sizes, unequal job release times and capacity limits. *International Journal of Production Research, 53*(6),1680-1711.

Akturk, M. S., & Ozdemir, D. (2001). A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research,135*(2),394-412.

Alhadi, G., Kacem, I., Laroche, P., & Osman, I. M. (2020). Approximation algorithms for minimizing the maximum lateness and makespan on parallel machines. *Annals of Operations Research, 285*(1-2),369-395.

Anghinolfi, D., Paolucci, M., & Ronco, R. (2021). A bi-objective heuristic approach for green identical parallel machine scheduling. *European Journal of Operational Research, 289*(2),416-434.

Bandyopadhyay, S., & Bhattacharya, R. (2013). Solving multi-objective parallel machine scheduling problem by a modified NSGA-II. *Applied Mathematical Modelling, 37*(10-11),6718-6729.

Berrichi, A., Amodeo, L., Yalaoui, F., Châtelet,E., & Mezghiche, M. (2009). Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. *Journal of Intelligent Manufacturing, 20*(4),389-400.

Biskup, D., Herrmann, J., & Gupta, J. N.D. (2008). Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics,115*(1),134-142.

Chen, Y., Huang, P., Huang, C., Huang, S., & Chou, F. (2021). Makespan minimization for scheduling on two identical parallel machines with flexible maintenance and nonresumable jobs. *Journal of Industrial and Production Engineering, 38*(1),1-14.

Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research, 47*(3),271-292.

Cochran, J. K., Horng, S., & Fowler, J.W. (2003). A multi-population genetic algorithm to solve multi-objective scheduling

problems for parallel machines. *Computers & Operations Research, 30*(7),1087-1102.
Cui, W., & Lu, Z. (2017). Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates. *Computers & Operations Research,80*,11-22.
Dipak, L., & Gupta., J.N.D. (2018). An Improved Cuckoo Search Algorithm for Scheduling Jobs on Identical Parallel Machines. *Computers & Industrial Engineering,126*, 348-360.
Fang, K. T., & Lin, B. (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers & Industrial Engineering, 64*(1),224-234.
Gupta, J.N.D., & A.J. Ruiz-Torres. (2001). A LISTFIT heuristic for minimizing makespan on identical parallel machines. *Production Planning & Control,12*(1), 28-36.
Hashemian, N., Diallo, C., & Vizvari, B. (2014). Makespan minimization for parallel machines scheduling with multiple availability constraints. *Annals of Operations Research, 213*,173-186.
Kayvanfar, V., Zandieh, M., & Teymourian, E. (2017). An intelligent water drop algorithm to identical parallel machine scheduling with controllable processing times: a just-in-time approach. *Computational and Applied Mathematics, 36*,159-184.
Kim, J. G., Song, S., & Jeong, B. (2020). Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times. *International Journal of Production Research*, *58*(6), 1628-1643. DOI:10.1080/00207543.2019.1672900.
Kramer, A., Dell'Amico, M., Feillet, D., & Iori, M. (2020). Scheduling jobs with release dates on identical parallel machines by minimizing the total weighted completion time. *Computers & Operations Research*, *123*, 105018.
Lee, C. H. (2018). A dispatching rule and a random iterated greedy metaheuristic for identical parallel machine scheduling to minimize total tardiness. *International Journal of Production Research, 56*(6), 2292-2308.
Lee, C.Y., & Chen, Z.L. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics, 47*,145-165.
Li, G., Liu, M., Sethi, S.P., & Xu, D. (2017). Parallel-machine scheduling with machine-dependent maintenance periodic recycles. *International Journal of Production Economics,186*, 1-7.
Li, K., Xiao, W., & Yang, S. (2019). Minimizing total tardiness on two uniform parallel machines considering a cost constraint. *Expert Systems With Applications,123*,143-153.
Liu, X., Chu, F., Zheng, F., Chu, C., & Liu, M. (2020). Parallel machine scheduling with stochastic release times and processing times. *International Journal of Production Research,59*(20),6327-6346.
Moradi, E., & Zandieh, M. (2010). Minimizing the makespan and the system unavailability in parallel machine scheduling problem: a similarity-based genetic algorithm. *International Journal of Advanced Manufacturing Technology, 51*(5-8),829-840.
Naderi, B., & Roshanaei, V. (2020). Branch-Relax-and-Check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. *European Journal of Operational Research, 286*,811-827.
Nessah, R., Yalaoui, F., & Chu, C. (2008). A branch and bound algorithm to minimize total weighted completion time on identical parallel machines with job release date. *Computers & Operations Research, 35*(4),1176-1190.
Schaller, J., & Valente, J. M. S. (2018). Efficient heuristics for minimizing weighted sum of squared tardiness on identical parallel machines. *Computers & Industrial Engineering, 119*,146-156.
Tan, Z., Yong, C., & An, Z. (2011). Parallel machines scheduling with machine maintenance for minsum criteria. *European Journal of Operational Research, 212*(2),287-292.
Wang, J. B., & Wei, C.M. (2011). Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties. *Applied Mathematics and Computation, 217*(20),8093-8099.
Wang, S., & Cui, W. (2020). Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time. *International Journal of Production Research, 59*(15),4579-4592.
Wang, S., & Liu, M. (2015). Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. *Journal of Manufacturing Systems,37*,182-192.
Wang, S., Wang, X., Yu, J., Ma, S., & Liu, M. (2018). Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of Cleaner Production,193*,424-440.
Wu, L., & Wang, S. (2018). Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. *International Journal of Production Economics, 201*,26-40.
Wu, C., Yao, Y., Dauzère-Pérès, S., & Yu, C. (2020). Dynamic dispatching and preventive maintenance for parallel machines with dispatching-dependent deterioration. *Computers & Operations Research,113*,104779.
Xu, D., Sun, K., & Li, H. (2008). Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan. *Computers and Operations Research,35*(4),1344-1349.
Xu, D., Cheng, Z., Yin, Y., & Li, H. (2009). Makespan minimization for two parallel machines scheduling with a periodic availability constraint. *Computers & Operations Research,36*(6),1809-1812.
Yalaoui, F., & Chu, C. (2006). New exact method to solve the $Pm/r_j/\sum C_j$ schedule problem. *International Journal of Production Economics,100*(1),168-179.
Yue, L., Guan, Z., Zhang, L., Ullah, S., & Cui, Y. (2019). Multi objective lotsizing and scheduling with material constraints in flexible parallel lines using a Pareto based guided artificial bee colony algorithm. *Computers & Industrial Engineering, 128*,659-680.
Yoo, J., & Lee, I.S. (2016). Parallel machine scheduling with maintenance activities. *Computers & Industrial Engineering,*

101,361-371.
Zhang, L., Deng, Q., Zhao, Y., Fan, Q., Liu, X., & Gong, G. (2021). Joint optimization of demand-side operational utility and manufacture-side energy consumption in a distributed parallel machine environment. *Computers & Industrial Engineering, 164* (2022),107863.
Zhou, H., Pang, J., Chen, P., & Chou, F. (2018). A modified particle swarm optimization algorithm for a batch-processing machine scheduling problem with arbitrary release times and non-identical job sizes. *Computers & Industrial Engineering*, *123*,67-81.

Appendix 1. Decoding algorithm based on FL

Algorithm 3：Full loading decoding algorithm

0 Initialize the cumulative processing time of each machine $Q_1 = \cdots = Q_m = 0$, the completion time set of jobs $JC = \{\emptyset\}$, the tardiness set of jobs $JT = \{\emptyset\}$, and the completion time set of machines $MC = \{\emptyset\}$, UT, mt, etc.
1 For chromosome $h = 1$ to $Popsize$ **do**
2 **For** machine $i = 1$ to m **do**
3 **For** the machine's job position $k = 1$ to n_i, select job J_{ik}
4 If $(Q_i + p_{ik} < UT)$ then
$C_{ik} = \max(C_{i(k-1)}, r_{ik}) + p_{ik}, Q_i = Q_i + p_{ik}$
5 If $(Q_i + p_{ik} = UT)$ then
$C_{ik} = \max(C_{i(k-1)}, r_{ik}) + p_{ik} + mt, Q_i = 0$
6 If $(Q_i + p_{ik} > UT)$ then
$C_{ik} = \max(C_{i(k-1)}, r_{ik}) + p_{ik} + mt, Q_i = p_{ik}$
7 $T_{ik} = \max\{C_{ik} - d_{ik}, 0\}, JC = JC \cup \{C_{ik}\}, \ JT = JT \cup \{T_{ik}\}$
8 **End For**
9 $C_i = \max\{C_{ik}\}, MC = MC \cup \{C_i\}, T_i = \sum T_{ik}$
10 **End For**
11 $C_{max} = \max\{C_i, C_i \in MC\}, TT = \sum_{i=1}^{m} T_i$
12 End For