Special Issue Article

Institution of MECHANICAL ENGINEERS
Journal of ENGINEERING MANUFACTURE

Proc IMechE Part B:
J Engineering Manufacture
1–12

DOI: 10.1177/0954405420928691
journals.sagepub.com/home/pib
SAGE

A Tabu-GA-based parallel machine scheduling with restrained tool resources

Shilong Wang[1,2], Haixu Zou[1] and Sibao Wang[1,2]

Abstract

In flexible manufacturing systems, disordered parallel machine scheduling may cause interruptions in the production line; thus, a reasonable scheduling plan benefits profits of manufacturing. In the literature, the operation sequence, the operation assignment, the tool scheduling, and the tool switching problems have been studied with unlimited resources. However, in practical manufacturing process, a scheduling problem along with these problems with restrained resources should be studied to make scheduling solutions more performable. There are a number of operations processed on a group of parallel machines, and each operation requires a set of tools where the tool copy number and tool service life are limited. The objective of this problem is to minimize the make-span with restrained resources. This article studies a parallel machine scheduling problem combining operation scheduling, tool scheduling and restrained resources, which benefit the real industry. A Tabu-Genetic Algorithm is proposed to find the optimal solution. In small size problem, the proposed method can achieve similar results whereas the mixed integer programming approach can obtain the exact solution. However, in large size problem, the latter cannot obtain the solutions within acceptable times, whereas the proposed method can find solutions within reasonable times. This algorithm performs better when compared to Tabu search algorithm and genetic algorithm in selecting local and global optima.

Keywords

Operation assignment, operation sequence, tool scheduling, mixed integer programming, Tabu-genetic algorithm

Date received: 23 December 2019; accepted: 26 April 2020

Introduction

In flexible manufacturing systems (FMS), different operations should be processed on different machines with various tools required. An appropriate parallel machine scheduling plan can significantly contribute to the modern industry. Parallel machine scheduling problems (PMSPs) have attracted much attention from both academia and industry for many years. Pinedo[1] states that PMSP are worth greatly discussing to develop the practical techniques in theory. According to Tang and Denardo,[2] Roh and Kim,[3] processing on parallel machines have been used in many real-world environments.

The PMSP is a jobshop schedule issue, including operation sequencing, operation assignment and the tool scheduling problem. The operation sequencing problem is to determine an operation sequence in order to minimize the idle time. According to Paiva and Carvalho,[4] the operation assignment problem is to determine the machines that will process each operation. According to Ozpeynirci et al.,[5] the tool scheduling problem involves scheduling the switching and loading of tools for a fixed operation sequence. The aim of tool scheduling problem is to change the tools on machines to make sure that the next operation in the sequence can be processed.

A large number of studies have paid attention to the PMSP considering each problem presented above because an unreasonable scheduling plan can cause extra machine, idle time, extra resource cost, profits loss, and ultimately, lower productivity. Fanjul-Peyro et al.,[6] Villa et al.,[7] and Beezo et al.[8] all present mathematical programming models and heuristic algorithms for PMSPs. Ozpeynirci et al.[5] provide a constraint programming approach considering the resource

[1]College of Mechanical Engineering, Chongqing University, Chongqing, China
[2]State Key Laboratory of Mechanical Transmissions, Chongqing University, Chongqing, China

Corresponding author:
Sibao Wang, College of Mechanical Engineering, Chongqing University, Chongqing 400044, China.
Email: wangsibaocqu@cqu.edu.cn

management issues. Jahromi et al.[9] generalize that three policies can be used when tool roles are considered in dynamic scheduling system.

Several studies combine the PMSP and the resource allocation problems that occur mostly in FMS. Novas and Henning[10] provide a review of FMS with constrained resources. Xu et al.[11] consider a variation of the classical single machine scheduling problem with tool changes. Moreover, the PMSP can be easily solved to optimality when compared to several algorithms. Later Zhang et al.[12] introduce and compare several decomposition strategies. Beezo et al.[8] solve the PMSP by sequencing a set of operations with specified processing times and tool requirement on a set of identical parallel machines with tool constraints. Chen et al.[13] determine the scheduling of a constant number of operations, and handle the scheduling of remaining operations by formulating it as a 0-1 integer programming model. In above studies, tool scheduling problems have been combined into the PMSP to make the jobshop schedule issue more practical. Therefore, the tool scheduling problem also becomes one research field, and an optimal tool scheduling plan can reduce the tool resource cost, which benefits the practical manufacturing process.

A small number of studies consider the combined PMSP with more complex constraints to make it similar to practical processing of manufacturing. Li et al.[14] propose an energy-saving method for permutation flow line scheduling problem. Kumar et al.[15] consider simultaneous loading and part and tool scheduling for FMS, and they proposed a modified genetic algorithm (MGA) method. Woo and Kim[16] consider the PMSP with time-dependent deterioration and multiple rate-modifying activities (RMAs). Zheng and Ling[17] consider the PMSP with additional resource constraints. They proposed a two-stage adaptive fruit fly optimization algorithm to solve the problem. Cheng and Huang[18] formulate an unrelated PMSP as a mixed integer programming (MIP) model and develop a modified genetic algorithm. Reddy et al.[19] consider simultaneous scheduling of machines, tools and many other resources to generate optimal sequences by using a symbiotic organisms search (SOS) algorithm. In conclusion, different kinds of algorithms are improved, and fusion algorithms are also proposed to solve these hybrid scheduling optimization problems. Some good results are gained, but optimization objectives are not comprehensive enough to consider operation sequence, operation assignment, tool loading, and tool scheduling simultaneously.

Recent studies have solved the PMSP considering loading and scheduling problems sequentially. Zeballos[20] proposes a constraint programming model that takes into account machine loading, scheduling, and tool allocation. Zeballos et al.[21] consider a variety of constraints found in the PMSP, such as machine eligibility and upper limits on costs. Recent research direction is making these constraints of PMSP more complex and developing a more practical scheduling model with restrained resources.

From the above studies, it can be understood that operation sequence, operation assignment, tool loading, and tool scheduling are studied either separately or sequentially, due to the increased complexity of the NP-hard problem with more variables. But in practical manufacturing systems, the PMSP should consider all these problems together and propose a more effective and performable solution. In the literature, according to Gökgür et al.,[22] there are several ways to solve these combined machine scheduling problems, which are categorized generally into two methods: (1) mathematic models, such as MIP model; and (2) heuristic approach algorithms, such as Tabu search algorithm and genetic algorithm (GA). They all have their own merits and demerits. The two methods are combined to solve combined machine scheduling problems efficiently. Morillo et al.[23] provide a new model and meta-heuristic approach for the resource-constrained scheduling problem. Lin et al.[24] use a mathematical model and a novel multi-objective heuristic algorithm to deal with scheduling problems considering variable machining parameters. Liu et al.[25] use MIP model and GA in ultra-flexible jobshop (uFJS) to improve the energy efficiency. And Liu et al.[26] also proposed a heuristic-based two-stage approach to solve energy-efficient integration of process planning and scheduling. However, major constraints such as tool copy number and tool service life should be well considered in the models so as to reduce tool resources cost.

In this article, therefore, the operation sequence, the operation assignment, and the tool scheduling problems will be combined into one model of the PMSP. A uniform PMSP with constraints such as tool service life, tool conflicts, and the number of tool copies is studied. There are numbers of operations; each one must be processed on one machine, and requires a set of slots for tool loading on the machine in advance. The number of tools available in the system is limited for reasons such as economic restrictions. Due to the complex constraints and the difficulty of NP-hard problem, mathematic models and fusion heuristic approach algorithms are employed to solve this combined PMSP mentioned above. And the robustness, efficiency, and accuracy of each method are compared based on the experiments with different scales of problem size.

The article is organized as follows: Section "Problem definition" presents the definition of this problem. Sections "Mathematic model" and "Tabu-GA search algorithm" describe the mathematic models based on

MIP and heuristic approach algorithms (Tabu-GA), respectively. In "Conclusion," concluding remarks are presented.

Problem definition

Traditional machine scheduling problem defines that n workpieces should be processed on m parallel machines. The processing sequences of each operation are arranged so as to minimize the production make-span time $Cmax$, where the operation number of each workpiece, the operation sequences of each workpiece, and the processing time of each operation are fixed. There is no order constraint between the processes of each operation, and each machine can only process one operation at the same time.

The PMSP differs from other methods on the following points:

- Each workpiece need only one operation, and the operation can be assigned to exactly one machine with no processing stop.
- The processing sequences are unfixed.
- There is no part movement between machines, hence no buffers are required.
- Every operation can be assigned to all machines, but the processing time of one operation on different machines may vary.
- The tool scheduling time from one machine to another is ignored.
- There are c tool types, and r_k tool copies of type k are available. Due to economic restrictions, the number of tool copies available may be smaller than the number of machines, which means $r_k \leqslant m$.
- Each operation consumes at most one copy of each tool.

A more reasonable PMSP should contain tool service life constraints and tool resource completion. The problems define as follow:

- There are competitive relationships of tool use between parallel machines at the same time.
- The tool cannot be used beyond its tool service life as its service life is limited.

The assumptions about the PMSPs in this article are given below:

- The tools that are loaded on machines cannot be scheduled until one operation processing is stopped.
- Tools are delivered to cutting tool magazine (CTM) or other required machines immediately when processed operation is completed, which means the delivery time is negligible.
- Tool magazines of the machines are initially empty.
- Each tool occupies one tool slot.
- Each tool is subject to wear, and has a finite tool service life.
- All operations are available in time zero.
- The tool magazine capacity of the machines is limited.
- Automated guided vehicles (AGVs) are used to convey tools and operations to machine. Each AGV is fitted with a manipulator arm which is designed to load and unload tools or operations. And the loading, unloading, and waiting time of tools are fixed and have little effect on completion time.
- An operation must be processed by at most one machine with tools required at the same time.
- When the same tool copy is required by two machines, there is a tool conflicts problem.
- The processing time include the waiting time of tool buffer, which means the waiting time is not considered in this model.
- Tool wear is proportional to tool service time, and the processing time of each tool copy cannot exceed its tool service life.
- Tool service life varies among different tool types, but different tool copies of the same tool type have the same tool service life.

Along with other PMSPs, operation sequences and tool usage priority are also considered to minimize the cost and make-span of all workpieces. Additional practical constraints, such as limited tool service life and competitive relationships of the tool usage, are also studied. Therefore, this problem can be studied by several sub-objectives, such as reducing process conflicts, making full advantages of each tool service life, minimizing completion time, and so on.

Mathematic model

In this section, the models considering those vital elements will be established to obtain the closed-form solutions by the MIP method. Then, an example is used to illustrate the method.

Model

First, the indices, parameters, and decision variables used in the MIP model are defined.

- Indices:

i, q: processing operation of different workpiece, $i, q = 1,...,n$
j: different parallel machine, $j = 1,....,m$
k: type of different tool, $k = 1,...,t$
h_k: different tool copy of the same tool type k, $h_k = 1,...,r_k$

- Parameters:

p_{ij}: time machine j process operation i need
r_k: tool inventory quantity of tool type k

$l(i)$: set of tools that machines process operation i required
R_{kh}: tool service time of copy h of tool k
ST: tool resource competition table that different operations require the same tool type when processing at the same time

- Decision variables:

$Cmax$: completion time of the last workpiece
C_i: completion time of operation i on different machines
S_{ij}: start time of operation i when processed on machine j
W_{ikh}: 1, if operation i processed by tool type k copy number h; 0, otherwise
U_{iq}: 1, if the start time of operation i precedes the start time of operation q when they both require the same tool type; 0, otherwise
X_{ij}: 1, if operation i is processed on machine j; 0, otherwise
Y_{iqj}: 1, if the completion time of operation i precedes the start time of operation q on machine j; 0, otherwise
The objective function and the constraints of MIP model are given below

$$\text{Minimize } Cmax \tag{1}$$

s.t.

$$C\max \geqslant C_i \quad \forall i \tag{2}$$

$$C_i = \sum_j (\mathrm{S}_{ij} + \mathrm{p}_{ij}\mathrm{X}_{ij}) \quad \forall i \tag{3}$$

$$S_{ij} \leqslant M(\mathrm{X}_{ij}) \quad \forall i,j \tag{4}$$

$$\sum_j X_{ij} = 1 \quad \forall i \tag{5}$$

$$\sum_j \sum_i (\mathrm{W}_{ikh}\mathrm{p}_{ij}\mathrm{X}_{ij}) \leqslant \mathrm{R}_{kh} \quad k \in l(i)$$

$$\sum_j S_{ij} \geqslant \sum_j (\mathrm{S}_{qj} + \mathrm{p}_{qj}\mathrm{X}_{qj}) - M(3 - (\mathrm{W}_{ikh} + \mathrm{W}_{qkh}) - U_{qi}) \tag{6}$$

$$\forall (i,q) \in \mathrm{ST}, \quad \forall \mathrm{k} | \mathrm{k} \in l(\mathrm{i}) \cap l(\mathrm{q}), \quad \mathrm{h} = 1, ..., \mathrm{r_k} \quad r_k \geqslant 2 \tag{7}$$

$$U_{iq} + U_{qi} \leqslant 1 \quad \forall (\mathrm{i},\mathrm{q}) \in \mathrm{ST} \tag{8}$$

$$W_{ikh} + W_{qkh} \leqslant 2(\mathrm{U}_{iq} + \mathrm{U}_{qi}) \quad \forall (\mathrm{i},\mathrm{q}) \in \mathrm{ST}, \quad \forall \mathrm{k} | \mathrm{k} \in l(\mathrm{i}) \cap l(\mathrm{q}), \quad \mathrm{h} = 2, ..., \mathrm{r_k} \tag{9}$$

$$S_{ij} \geqslant S_{qj} + p_{qj} - M(\mathrm{Y}_{iqj}) - \mathrm{M}(2 - \mathrm{X}_{qj} - \mathrm{X}_{ij}) - \mathrm{M}(2 - W_{ikh} - W_{qk'h'}) \quad \forall i \neq q, j \;\; k \in l(\mathrm{i}), \mathrm{k}' \in l(\mathrm{q}) \quad h, h' = 1, ..., \mathrm{r_k} \quad \mathrm{r_k} \geqslant 2 \tag{10}$$

$$X_{ij} + X_{qj} \geqslant 2(\mathrm{Y}_{iqj} + \mathrm{Y}_{qij}) \quad \forall i \neq q, j \tag{11}$$

$$X_{ij} + X_{qj} \leqslant Y_{iqj} + Y_{qij} + 1 \quad \forall i \neq q, j \tag{12}$$

$$\sum_{h=1}^{r_k} W_{ikh} = 1 \quad \forall i, k \in l(\mathrm{i}) \tag{13}$$

$$C\max \geqslant 0 \tag{14}$$

$$C_i \geqslant 0 \quad \forall i \tag{15}$$

$$S_{ij} \geqslant 0 \quad \forall i,j \tag{16}$$

$$X_{ij} \in \{0,1\} \quad \forall \mathrm{i}, \mathrm{j} \tag{17}$$

$$W_{ikh} \in \{0,1\} \quad \forall \mathrm{i}, \quad \forall k \in l(\mathrm{i}), \quad \mathrm{h} = 1, ..., \mathrm{r}_k \tag{18}$$

$$U_{iq} \in \{0,1\} \quad \forall i \neq q \tag{19}$$

$$Y_{iqj} \in \{0,1\} \quad \forall i \neq q, \forall q, j \tag{20}$$

The objective function (1) minimizes the completion time of production. Constraint (2) ensures that the completion time of production is the completion time of the last workpiece. Constraint (3) defines that the completion time of production is calculated by each operation, their starting time, and their processing time. Constraint (4) demands that the production item starts off only when tools required for processing are assigned. In constraint (5), an operation can be processed at only one machine at a time. Constraint (6) guarantees that tools are not used beyond their tool service life. Constraints (7)–(9) guarantee that two operations cannot be processed at the same time when they both need the same tool copy of one tool type. The two operations must be processed in sequence. Due to constraints (10)–(12), there is processing sequence relationship between two operations when they are processed on the same machine. The starting time of one operation must be greater or equal to the completion time of the another one. Finally, constraint (13) make sure that, at one time, a certain operation at a machine only use one copy of tools required for processing, and constraints (14)–(20) are set constraints.

Example results

An example with a small scale of data is given to illustrate the problem environment and to visualize the solution structure on a Gantt chart. The example defines that there are 10 operations to be processed on 3 uniform parallel machines with 8 different types of tools. Processing times of each operation on different machine, tool requirements of each operation, and the tool service time of each tool type are given in Tables 1–3, respectively.

The completion time of production is found to be 201 min. The results of scheduling, the processing times of tool copies, and the Gantt chart are given in Table 4, Table 5, and Figure 1, respectively. As can be seen in the Gantt chart, there are expected times with demand of factors that each tool copy should not be beyond its tool service life and the limited number of tool copies.

A new example, which does not consider the tool resource constraints (tool service life and tool copy numbers), is also carried out. The results of the new example show that the make-span is 193 (as shown in Figure 2) while the number of tool copy is more than 3.

Table 1. Processing times of operations on each machine.

Operation	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
M1	60	42	147	77	51	49	40	57	126	108
M2	138	122	101	132	13	144	112	138	77	28
M3	44	148	33	104	57	133	98	129	41	57

Table 2. Types of tools required by each operation.

Operation	1	2	3	4	5	6	7	8	9	10
Tools	2,8	1,4,8	2,3	2,7	1,2,6,8	3,6,7,8	4,5	1,3,7,8	3,5,	1,6

Table 3. Tool service life of each type of tools.

Tool	1	2	3	4	5	6	7	8
Life	100	150	90	45	45	50	110	155

Table 4. Expected scheduling results of the example.

Operation	Machine	Starting time	Completion time	Tool (copies) used
1	2	0	138	2(1),8(1)
2	1	106	148	1(2),4(2),8(2)
3	3	115	148	2(2),3(1)
4	3	0	104	2(2),7(1)
5	2	148	161	1(1),2(2),6(1),8(1)
6	1	57	106	3(2),6(2),7(2),8(2)
7	1	161	201	4(1),5(1)
8	1	0	57	1(2),3(1),7(2),8(2)
9	3	148	189	3(2),5(2)
10	2	173	201	1(1),6(1)

When results from the two examples are compared, it is clear that the deviation of the make-span from the two examples is less than 5 percent, which is acceptable. However, solutions of the new mathematic model need much more tool copies, which may cause a huge cost. Therefore, considering the constraints of tool service life and tool copy numbers, the optimal scheduling results obtained by MIP could be more efficient while the deviation of expected make-span is reasonable.

Although the scheduling problem can be solved by MIP, it will be impossible to solve the problem instances when there are much more operations. Therefore, for the large size scheduling problem, it is definitely necessary to propose an efficient algorithm which could find near-optimal solutions in a reasonable time.

Tabu-GA search algorithm

Tabu search is one kind of meta-heuristic method introduced by Glover.[27] This algorithm starts with an initial feasible solution and develops to another solution in the defined neighborhood of the current one at each iteration. The solution which has the diversity of

Table 5. Processing times of each tool copy.

Copy	Times	Copy	Time
1(1)	98	1(2)	99
2(1)	138	2(2)	150
3(1)	90	3(2)	90
4(1)	40	4(2)	42
5(1)	40	5(2)	41
6(1)	41	6(2)	49
7(1)	104	7(2)	106
8(1)	151	8(2)	148

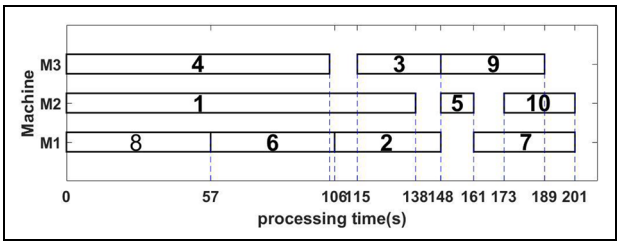

Figure 1. Gantt chart of the expected scheduling results.

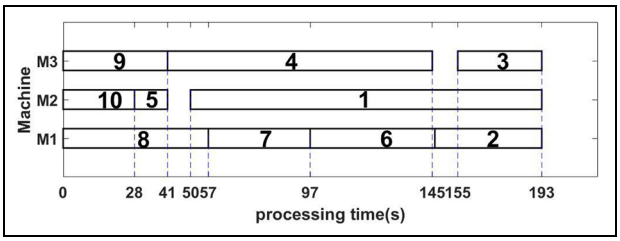

Figure 2. Gantt chart of the unconstrained scheduling results.

objective function is to be selected. It is forbidden to repeat a move for a specified number known as the Tabu tenure during the process of iterations. The Tabu tenure is a table that records the improved process, and is the best solution in a memory-based strategy; it can prevent the algorithm from getting the locally optimal solution.

The Tabu search algorithm has a good ability to move to the neighborhood intensive solution of current one but the algorithm jumps to a solution in a different neighborhood when the inner loop has terminated. Then a generation method of initial population using GA is critical for the diversification of solutions because the generation method of GA includes the optimal gene of last solution.

Detailed information about Tabu search and GA can also be found in Glover[27] and Golberg.[28] Sivaram et al.[29] also use the hybrid GA to explore the process of selecting optimal generation along with their weights and fusion function. However, this method suffers a setback of slow convergence. The convergence rate can be improved by merging Tabu search—a best local search—with the GA. Li and Gao[30] have proposed an algorithm which combines the global search and the local search by using GA and Tabu to solve flexible jobshop scheduling problem. The Tabu-GA is used to solve the PMSP in the article. The details of the algorithm are presented below.

Initial solution

The initial solution is found by a greedy heuristic algorithm that selects the operations according to a priority rule, similar to the one used by Roh and Kim.[3] The stepwise description is given below, and the pseudocode of greedy heuristic method is provided.

Let J_0: set of operations that have not been assigned to any machine; M_j: set of operations that have been assigned to machine j; L_0: set of tool copies available at one processing time, $L_0 = \{l_{kh}\}, l_{kh} = 1$, if the tool k copy h is free; otherwise, (g): set of tool copies used by all operation, $L_g = \{l_{kh}{}'\}, l_{kh}{}' = 1$, if the tool k copy h is used by any operation processed; otherwise $l_{kh}{}' = 0$; $T(g)$: set of tool service life of tool copies during Tabu move g ($g = 0$ when generate the initial solution), $T(g) = \{\omega_{kh}\}$, where ω_{kh} is the processing time of tool k copy h; it is made sure that the processing time of any tool copy is not beyond its tool service time, $\omega_{kh} \leqslant R_{kh}$.

Step 0: Set $J_0 = \{1,2,\ldots,n\}$, $M_j = \{\}$, $S_{ij} = 0, C_j = 0\ \forall i,j$ and $L_0 = \{l_{kh} = 1\}$, $L(0) = \{l'_{kh} = 0\}, T(0) = \{w_{kh} = 0\}$ $\forall k,h$.

Step 1: Find the machine which is available first, the machine with minimum S_{ij}. List all operations of J_0 and compare all priority values using the following equation

$$\lambda_{ij} = p_{ij} \times a_{ij} \times (b_{ij}{}^2 + 1) \tag{21}$$

where λ_{ij}: priority value of operation i on machine j; p_{ij}: processing time of operation i on machine j; a_{ij}: number of tools available that should load additionally to machine j before operation i started; b_{ij}: number of tools unavailable that should load additionally to machine j before operation i started.

As can be seen from the equation, the operation that has shorter processing time and that demand fewer additional tool copies has a lower priority value and is more important than any other operations. As an operation cannot start processing without tool copies required, the starting time of operation needs to be delayed until the tool copies become free. Hence, the completion time is affected by b_{ij}. The square of b_{ij} is used to increase the priority value of those operations since it plays an effective role.

List all priority values and select the operation–machine pair with the minimum priority value in the list. Make operation i to be processed on machine j.

Step 2: Move the tools that are required by operation i and that are not already loaded from total tool magazine or other machines to slots of machine j before operation i starts processing.

Update $L_0(l_{kh} = 0\ \forall k,h \in l(i))$ and $L(0)(l'_{kh} = 1\ \forall k,h \in l(i))$.

Step 3: Judge tools of $l(i)$ that are required by operation i processed on machine j. If one is not free, then delay the starting time of operation i on machine j until the tool becomes free, which means $S_{ij} \geqslant C_q$ (operation i has processed before operation q, and only when tools of elements $l(i)$ are free, the starting time of operation i is equal to the completion time of operation q, which means $S_{ij} = C_q = S_{qj} + p_{qj}$) and when operation i has completed, update $L_0(l_{kh} = 1\ \forall k,h \in l(i))$.

Step 4: Update $T(0)(w_{kh} = w_{kh} + p_{ij}\ \forall k,h \in l(i))$, J_0 and M_j ($J_0 = J_0$-$\{i\}$ and $M_j = M_j + \{i\}$). If $J_0 = \{\}$, then stop. Otherwise, go to **Step 1**.

Solution code and generation method

The sample solution representation of operations sequence is given below:

Machine 1: Operation8—Operation4—Operation6—Operation10 (which translate into code $M_1 = (8,4,6,10)$
Machine 2: Operation5—Operation9 (which translate into code $M_2 = (5,9)$
Machine 3: Operation3—Operation1—Operation2—Operation7 (which translate into code $M_3 = (3,1,2,7)$

The neighborhood of a solution is generated in two ways:

1. Switch two operations randomly without considering resource competition relationship.

2. Move an operation to another position.

Iteration and diversification phases

In the iteration phase of the algorithm, all the possible neighborhood of the current solution will be generated and compared with priority values. They will move to the optimal solution which is feasible. This allows us to search the local area of the current solution. When iterating to a certain number, the inner loop stops.

To avoid getting the local optimal solution, the search moves to diversification phases. In this phase, another different initial pollution is created by generation of GA.

As the efficiency of the algorithm is highly depended on their quality, the fitness value to evaluate its performance, which is similar to Zhou et al.,[31] is used. After the generation of new solution, fitness value of each move is calculated by F_g. The fewer the fitness value, the better the performance of Tabu search. Hence, solution with fewer fitness values has more chances to survive. Because the objective function is to minimize the make-span with constrained tool and tool service life, fitness value can be obtained using the following function

$$F_k = \alpha \cdot C_{\max}(\mathrm{k}) \times \mathrm{e}^{\beta \cdot l(\mathrm{k})} \tag{22}$$

where α and β are positive real numbers, $C_{\max}$ is the objective function value (make-span) of the Tabu move k, and $l(g)$ is the number of tool copies required by all operations. Considering the completion time and the number of tool copies, the $C_{\max}(g)$ is more important than $l(g)$ when the copies is under a specified number; but with the tool numbers increasing, processing cost increases. So α and β are so important that the weight coefficient are determined by processing conditions and the cost of tool copies.

1. Calculate selection probability of each Tabu search for next initial solution

$$P(g) = 1 - \frac{F(g)}{\sum_{i=1}^{N} F(g)} \quad g = 1, 2, 3, ..., N \tag{23}$$

2. Generate cut points, $S(g)$

$$\begin{aligned} &S(0) = 0 \\ &S(g) = \sum P(g), \qquad g = 1, 2, 3, ..., N \end{aligned} \tag{24}$$

3. Generate N random number uniformly distributed between 0 and 1, ξ_g for $g = 1,2,3,...,N$
4. For each ξ_g, equation $S(g-1) < \xi_g < S(g)$ gives Tabu move g to be selected
5. List the $S(g)$ selected and make the best one with highest selection probability and lowest fitness to be the next initial solution of Tabu search.

Algorithm flow description

First, parameters are defined as follows

N^i: number of the iteration phase

N^o: number of the diversification phase

N_1: maximum number of iteration phase

N_2: maximum number of the diversification phase

V: fitness values of all feasible solutions ($C_{\max}$(k), l(k), and F_k)

V^c: fitness values of the current solution

V^b: fitness values of the local optimal solution

V^*: fitness values of the feasible solution

tt_{ij}: time of the feasible solution that operation i is processed on machine j is Tabu, and the length is set to be $\left[\sqrt{N_1}\right]$

ts^c: Tabu status of the current solution, 1 if Tabu, otherwise 0

ts^b: Tabu status of the local optimal solution of current one

tl: Tabu list that record the Tabu tenure and Tabu status

Then detail description of this search algorithm is provided as follows

Step 0: Set $V^c = \infty$, $V^* = \infty$, and $N^0 = 0$, find an initial solution using greedy heuristic method described in section "Initial solution."

Step 1: Set $N^i = 1$, $N^0 = N^0 + 1$, $V = \{\}$, and $tl = \{\}$. Set the initial solution to be the current solution and consider its fitness values as V^i, which can obtained by calculating method of fitness values described in section "Algorithm flow description." Set $V^c = V^i$, $tl = tl + \{ts^c\}$. Set $V^* = V^i$ if $V^i < V^*$. Set $V = V + V^*$.

Step 2: Generate all neighborhood of the current solution using two ways described in section "Solution code and generation method."

Step 3: Select the local optimal solution due to the fitness values.

a) If $ts^b = 0$, go to Step 4.
b) If $ts^b = 1$

i. If $V^b < V^*$, $ts^b = 0, tl = tl - \{ts^b\}$ and then go to Step 4.
ii. If $V^b > V^*$, remove the best neighbor from the list and go to Step 3.

Step 4: Decrease the time that feasible solutions are Tabu by one $tt_{\mathrm{ij}} - 1$ and then consider local optimal solution as the current one. Set $V^c = V^b$, $tl = tl + \{ts^b\}$. Set $V^* = V^b$ if $V^b < V^*$ and set $V = V + V^*$,$N^i = N^i + 1$.

a) If $N^i \leqslant N_1$, go to Step 2.
b) If $N^i > N_1$

i. If $N^o < N_2$, $tt_{ij} = \{0\}$, create an another initial solution using method of calculating selection probability described in section "Algorithm

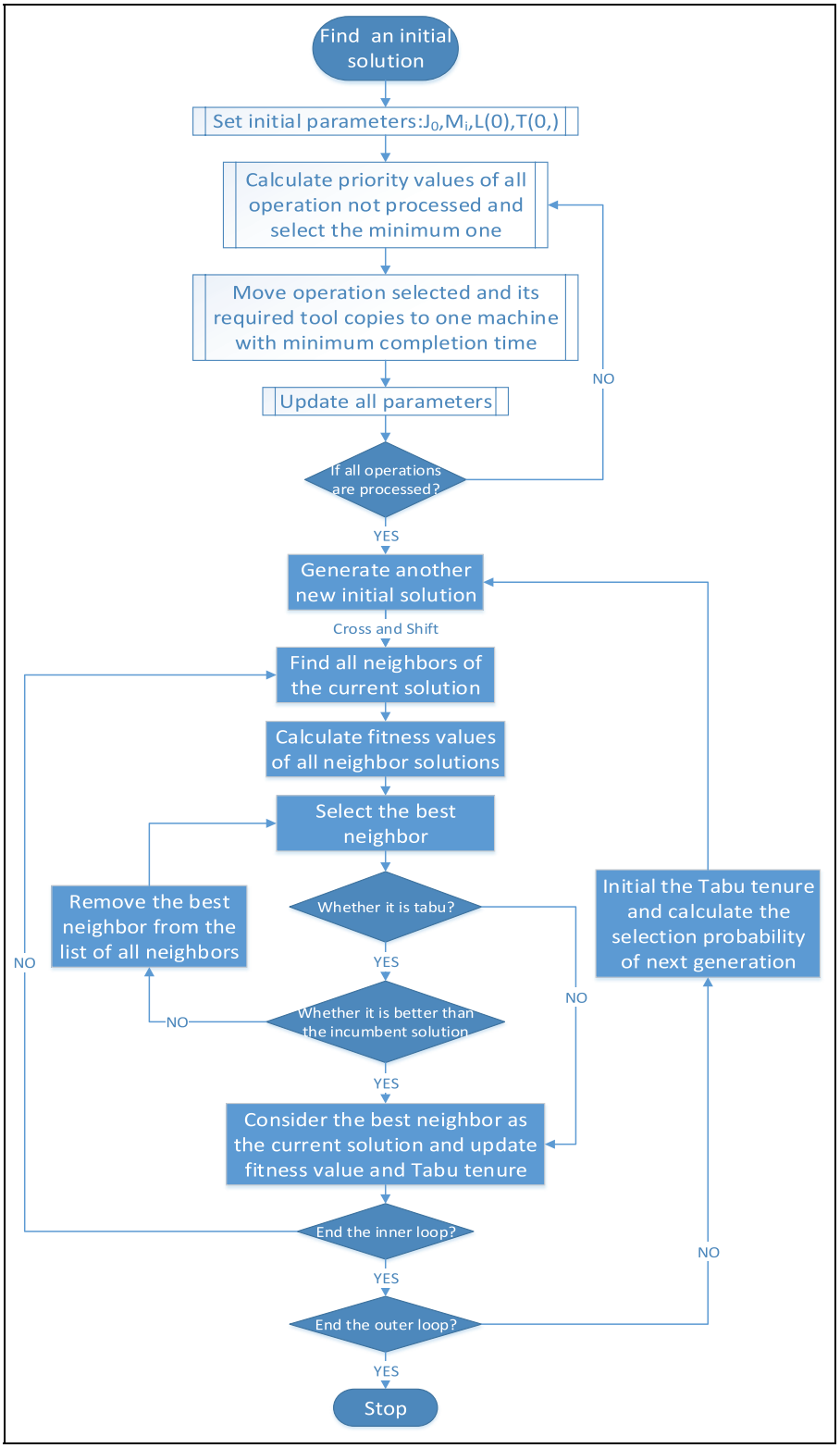

Figure 3. Tabu-GA search algorithm.

flow description," in which $C_{\max}(k)$,$l(k)$, and F_k can be found from V, and then go to Step 1.

ii. If $N^o = N_2$, stop.

The flow chart of algorithm is shown in Figure 3.

Comparison of experimental results

The example is defined in section "Mathematic model" and in some other cases similar to it. The number of iteration and diversification phase are set as 80 and 20, respectively. A total of 1600 iterations are performed

Table 6. Performance of MIP.

(n, m, t)	C	Computational Time (s)		
		Min	Average	Max
(8, 2, 5)	6	6.70	9.12	14.73
(8, 2, 8)	10	4.64	5.47	8.77
(10, 2, 5)	4	328.15	623.20	1442.02
(10, 2, 8)	10	578.60	789.81	1453.30
(10, 3, 5)	3	78.35	239.88	519.89
(10, 3, 8)	10	1053.80	2021.78	2989.76
(15, 2, 8)	0	–	–	–

Table 7. Performance of Tabu-GA.

(n, m, t)	C	Computational Time (s)		
		Min	Average	Max
(8, 2, 5)	9	67.75	107.06	141.61
(8, 2, 8)	7	81	102.41	129.47
(10, 2, 5)	10	147.76	208.71	247.04
(10, 2, 8)	3	58	151.47	220.9
(10, 3, 5)	10	219.70	291.93	342.40
(10, 3, 8)	6	166.02	227.00	302.07
(15, 2, 8)	3	566.58	635.58	721.91

for each problem instance. The Tabu-GA search is coded in the MATLAB® 2016(b) with tool boxes yalmip and lpsolve using Intel® Core™ i5-3470 CPU @ 3.20 GHz 3.60 GHz and 4 GB RAM.

There are several problem instances that n operations are processed on m machines with t kinds of tool types to be illustrated in this case. Compared with the results of MIP shown in Table 6 and in Table 7, the results are shown for the Tabu-GA search heuristic: the number operations, machines and tools (n, m, t); the number of optimal solutions found (C); the minimum, average, and maximum computational time in seconds (computational time).

Comparing Table 6 with Table 7, the computational time of each model with different sizes of problem instance are shown in Figure 4. From Figure 4, it can be concluded that the MIP model can gain optimal solutions in small size problem instances, but as the size increases, this model need much more computational time and it cannot gain optimal solutions within reasonable time. However, the efficiency of Tabu-GA model is better than the MIP model as the problem size increases.

Table 8 shows the range of make-span deviations from the best optimal solution which can be found in MIP (Deviation), and "Deviation" can be defined by equation (25). These figures are shown to demonstrate that these models can gain optimal solutions with low-level deviation ranges of the make-span within a reasonable time.

$$Deviation = \frac{C_T - C_B}{C_B} \times 100\% \tag{25}$$

where C_T is one solution of the Tabu-GA model, and C_B is the best optimal solution of the MIP model.

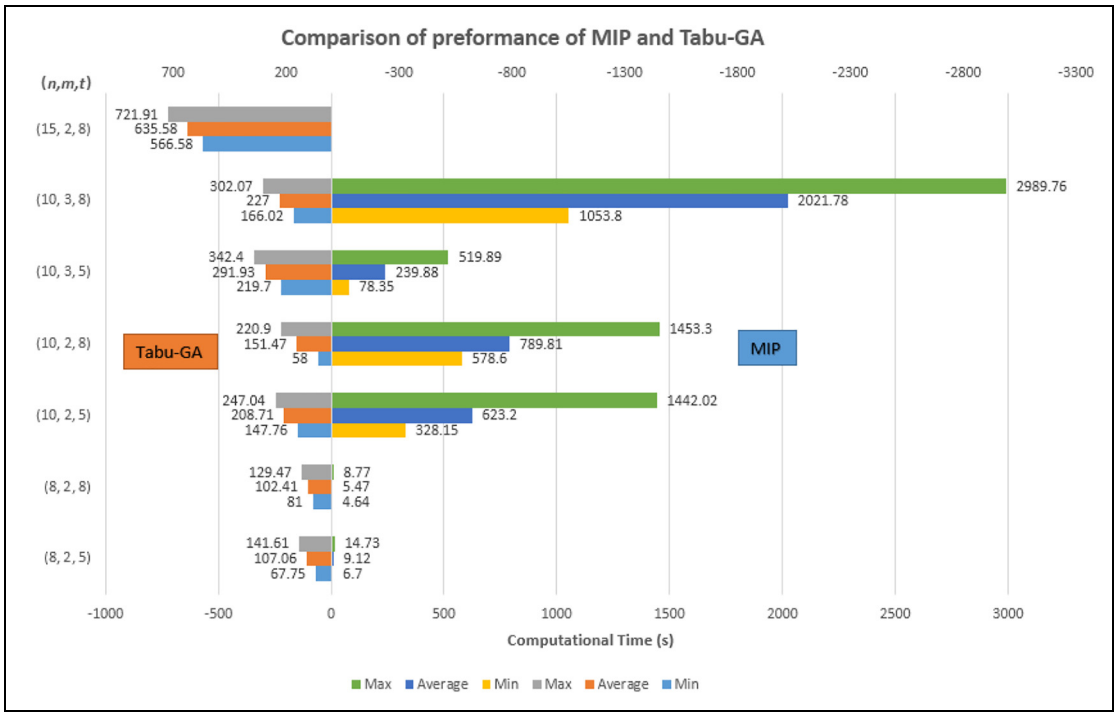

Figure 4. Tabu-GA search algorithm.

Table 8. Comparison of make-span deviation between MIP and Tabu-GA.

(n, m, t)	Deviation (%)
(8, 2, 5)	0 ~ 3.07 (average 0.31)
(8, 2, 8)	0 ~ 6.91 (average 3.19)
(10, 2, 5)	0 ~ 9.83 (average 4.31)
(10, 2, 8)	0 ~ 7.95 (average 3.47)
(10, 3, 5)	0 ~ 11.61 (average 5.74)
(10, 3, 8)	0 ~ 4.14 (average 2.26)
(15, 2, 8)	–

Tables 6–8 show that the Tabu-GA can search near-optimal solutions, and the problem size has no significant effect on the solution quality; however, the calculating times increase as the iteration numbers increase. MIP performs better when the number of operations is under a specified figure, but the Tabu-GA search can find more feasible solutions in reasonable times with the number of operations increasing.

Performance analysis

Compared with other algorithms, the superior performance of Tabu-GA search algorithm is demonstrated in Figure 5. The total generation number is 300, the population size is 30, and iteration process will be stopped to prevent extra computational cost when the optimal solution is found. The figure involves iterations and average make-span of all population in each generation, which represents the optimization performance of Tabu search algorithm, GA, and Tabu-GA search algorithm..

The results show the following:

a) The GA started from 478.4, and the optimal solution, which is 225, is found after generation 92
b) The Tabu search algorithm's initial solution is 826.2 and the optimal solution 206.5 is found in the 60th generation
c) The Tabu-GA started from initial solution 413.1, and the near-optimal solution 206.5 is found when in the 55th generation. After that, iteration process is stable, but the optimal solution 202.1 is found after generation 258, which is ignored in Figure 5 because the deviation is subtle that cannot obviously reflect on the comparison figure.

It can be concluded that:

a) The GA has a better performance in selecting initial solution;
b) Tabu search algorithm needs less computational cost to find the optimal solution;
c) Tabu-GA search is an effective fusion algorithm that leverages benefits of both Tabu search algorithm and GA, which can also find better optimal solution than Tabu search algorithm and GA after a number of iterations.

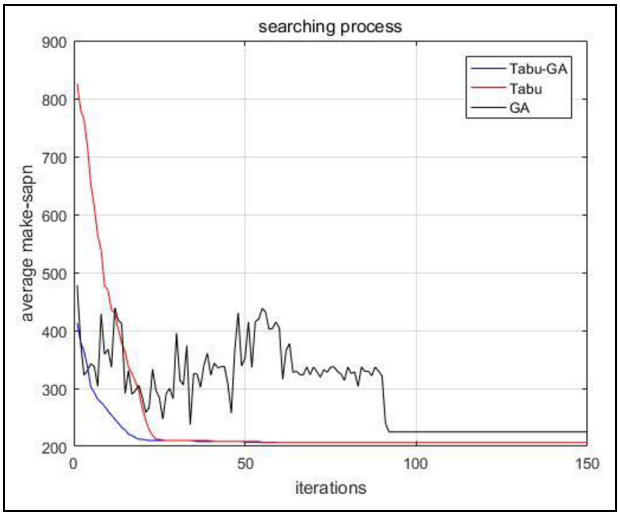

Figure 5. Performance comparison.

To sum up, GA can perform better in global search and provide a great flexibility to ensure the effectiveness and efficiency of the algorithm. However, this method suffers a setback of slow convergence. Tabu search can improve the convergence and local search capability. Furthermore, the mapping relationship between initial solution and neighborhood can also be improved by GA. The Tabu-GA algorithm is proposed to combine the advantages of Tabu and GA. It has a great flexibility, and the convergence is fast. Therefore, the Tabu-GA is better than both Tabu and GA in global or local optima searching. Tabu-GA is used in solving PMSPs defined in the article, and it shows a better optimization performance in optima search and strong convergence compared to Tabu search algorithm and GA.

Conclusion

In this article, the scheduling problem to minimize the make-span with restrained tool resources on uniform parallel machines is studied considering operation assignment, operation assignment, and tool scheduling simultaneously. A Tabu-GA search algorithm is proposed to search the near-optimal solutions within an acceptable time and deviation. In small size cases, the results obtained by the proposed method are similar to those obtained by MIP, which can find the exact solution. However, MIP cannot solve the large size problem due to the difficulty of problem as the problem size increases, while the proposed method still works. In addition, compared to Tabu search algorithm and GA, the proposed method performs better in solving scheduling problems.

During practical machining process, magazine of each machine tool (such as lathe and machining center) has limited tool type, and the tool number for each

type is also limited. Furthermore, tool service life will be varied with the machining process due to the effect of tool wear. In the proposed study, the limited tool resource and tool service life are considered in the PMSP. Therefore, the results generated by the proposed method are more realistic. In the future, tool loading problems such as switching time and transporting load, which are negligible in this study, will also be considered.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by National Key Technologies Research and Development Program of China (Grant No.2018AAA0101804), National Natural Science Foundation under Grant No.51635003, and Innovation Team Development Program of the Ministry of Education under Grant No.IRT 15R64.

ORCID iD


Sibao Wang https://orcid.org/0000-0003-2873-4631

References

1. Pinedo M. *Scheduling: theory, algorithms, and systems multi-coloring*. New York: Springer, 2008.
2. Tang CS and Denardo EV. Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. *Oper Res* 1988; 36(5): 767–777.
3. Roh HK and Kim YD. Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter. *Int J Prod Res* 1997; 35(11): 2989–3004.
4. Paiva GS and Carvalho MAM. Improved heuristic algorithms for the job sequencing and tool switching problem. *Comput Oper Res* 2017; 88: 209–219.
5. Ozpeynirci Selin Gokgr B and Hnich B. Parallel machine scheduling with tool loading. *Appl Math Model* 2016; 40(9–10): 5660–5671.
6. Fanjul-Peyro L, Perea F and Ruiz R. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *Europ J Oper Res* 2017; 260(2): 482–493.
7. Villa F, Vallada E and Fanjul-Peyro L. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Syst Appl* 2018; 93: 28–38.
8. Beezo AC, Cordeau JF, Laporte G, et al. Scheduling identical parallel machines with tooling constraints. *Europ J Oper Res* 2017; 257(3): 834–844.
9. Jahromi MHMA, Tavakkoli-Moghaddam R, Makui A, et al. A novel mathematical model for a scheduling problem of dynamic machine-tool selection and operation allocation in a flexible manufacturing system: a modified evolutionary algorithm. *Sci Iran Trans E, Ind Eng* 2017; 24(2): 765.
10. Novas JM and Henning GP. Integrated scheduling of resource-constrained flexible manufacturing systems using constraint programming. *Expert Syst Appl* 2014; 41(5): 2286–2299.
11. Xu D, Min L, Yin Y, et al. Scheduling tool changes and special jobs on a single machine to minimize makespan. *Omega* 2013; 41(2): 299–304.
12. Zhang X, Wang S, Yi L, et al. An integrated ant colony optimization algorithm to solve job allocating and tool scheduling problem. *Proc IMechE, Part B: Journal of Engineering Manufacture* 2018; 232(1): 172–182.
13. Chen L, Ye D and Zhang G. Parallel machine scheduling with speed-up resources. *Europ J Oper Res* 2018; 268(1): 101–112.
14. Li S, Liu F and Zhou X. Multi-objective energy-saving scheduling for a permutation flow line. *Proc IMechE, Part B: Journal of Engineering Manufacture* 2018; 232(5): 879–888.
15. Kumar N, Chandna P and Joshi D. Integrated scheduling of part and tool in a flexible manufacturing system using modified genetic algorithm. *Int J Syst Assur Eng Manag* 2017; 8(2): 1596–1607.
16. Woo YB and Kim BS. Matheuristic approaches for parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Comput Oper Res* 2018; 95: 97–112.
17. Zheng X and Wang L. A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. *Expert Syst Appl* 2016; 65: 28–39.
18. Cheng CY and Lu WH. Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. *J Manuf Syst* 2017; 42: 1–10.
19. Reddy NS, Ramamurthy DV, Prahlada Rao K, et al. A novel metaheuristic method for simultaneous scheduling of machines and tools in multi machine FMS. In: *International conference on recent trends in engineering, science & technology—(ICRTEST 2016)*, Hyderabad, India, 25–27 October 2016. New York: *IEEE*.
20. Zeballos LJ. A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robot Comput Integr Manuf* 2010; 26: 725–743.
21. Zeballos LJ, Quiroga OD and Henning GP. A constraint programming model for the scheduling of flexible manufacturing systems with machine and tool limitations. *Eng Appl Artif Intel* 2010; 23(2): 229–248.
22. Gökgür B, Hnich B and Özpeynirci S. Parallel machine scheduling with tool loading: a constraint programming approach. *Int J Prod Res* 2018; 56(16): 5541–5557.
23. Morillo Torres D, Barber F and Salido MA. A new model and metaheuristic approach for the energy-based resource-constrained scheduling problem. *Proc IMechE, Part B: Journal of Engineering Manufacture* 2019; 233(1): 293–305.
24. Lin W, Tian G, Li Z, et al. Flow shop scheduling with low carbon emission and variable machining parameters. *Proc IMechE, Part B: Journal of Engineering Manufacture* 2019; 233(5): 1561–1572.

25. Liu N, Zhang YF and Lu WF. Energy-efficient integration of process planning and scheduling in discrete parts manufacturing with a heuristic-based two-stage approach. *Int J Adv Manuf Technol* 2020; 106(5): 2415–2432.
26. Liu N, Zhang YF and Lu WF. Improving energy efficiency in discrete parts manufacturing system using an ultra-flexible job shop scheduling algorithm. *Int J Precis Eng Manuf Technol* 2019; 6(2): 349–365.
27. Glover F. Future paths for integer programming and links to artificial intelligence. *Comput Oper Res* 1986; 13(5): 533–549.
28. Golberg DE. *Genetic algorithms in search, optimization, and machine learning*. Boston, MA: Addion Wesley, 1989, p.36.
29. Sivaram M, Batri K, Amin Salih M, et al. Exploiting the local optima in genetic algorithm using Tabu search. *Indian J Sci Tech* 2019; 12(1): 1–13.
30. Li X and Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int J Prod Econ* 2016; 174: 93–110.
31. Zhou H, Feng Y and Han L. The hybrid heuristic genetic algorithm for job shop scheduling. *Comput Ind Eng* 2001; 40(3): 191–200.