Expert Systems With Applications 175 (2021) 114843

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect

Like Zhang, Qianwang Deng *, Ruihang Lin, Guiliang Gong, Wenwu Han

State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha 410082, China

ARTICLE INFO

Keywords:
Unrelated parallel machine scheduling
Sequence and machine-dependent setup times
Workers resources
Learning effect

ABSTRACT

The existing papers on unrelated parallel machine scheduling problem with sequence and machine-dependent setup times (UPMSP-SMDST) ignore the worker resources and learning effect. Given the influence and potential of human factors and learning effect in real production systems to improve production efficiency and decrease production cost, we propose a UPMSP-SMDST with limited worker resources and learning effect (NUPMSP). In the NUPMSP, the workers have learning ability and are categorized to different skill levels, i.e., a worker's skill level for a machine is changing with his accumulating operation times on the same machine. A combinatorial evolutionary algorithm (CEA) which integrates a list scheduling (*LS*) heuristic, the shortest setup time first (*SST*) rule and an earliest completion time first (*ECT*) rule is presented to solve the NUPMSP. In the experimental phase, 72 benchmark instances of NUPMSP are constructed to test the performance of the CEA and facilitate future study. The Taguchi method is used to obtain the best combination of key parameters of the CEA. The effectiveness of the *LS*, *SST* and *ECT* is verified based on 15 benchmark instances. Extensive experiments conducted to compare the CEA with some well-known algorithms confirm that the proposed CEA is superior to these algorithms in terms of solving accuracy and efficiency.

1. Introduction

Unrelated parallel machine scheduling problem (UPMSP) with sequence and machine-dependent setup times (UPMSP-SMDST) is an extension of classical parallel machine scheduling problem, in which machines are non-identical and jobs' setup times depend on both sequence and machines. It is very difficult to tackle even if there is only one machine which is equivalent to the traveling salesman problem and has been proven to be NP-hard problem (Błażewicz, Ecker, Pesch, Schmidt, & Węglarz, 1996). However, due to the broad application in many real production systems such as truss industry (Rabadi, Moraga, & Al-Salem, 2006), printed writing board manufacturing industry (S.-W. Lin & Ying, 2014), photolithography workshop in Semiconductor manufacturing (Bitar, Dauzere-Peres, Yugma, & Roussel, 2016) and Heating, Ventilation and Air Conditioning factory (Perez-Gonzalez, Fernandez-Viagas, Zamora Garcia, & Framinan, 2019), the UPMSP-SMDST has become a research hotspot in the latest two decades.

Rabadi, et al. (2006) were among the first to consider UPMSP-SMDST. They proposed a new metaheuristic for randomized priority search to address it with the criterion of minimizing the makespan. Later, some single-objective algorithms also have been proposed to optimize the makespan of UPMSP-SMDST, including two stage ant colony optimization algorithm (Arnaout, Rabadi, & Musa, 2010), improved genetic algorithm(Vallada & Ruiz, 2011), variable neighborhood descent search algorithm (Fleszar, Charalambous, & Hindi, 2012), restricted simulated annealing algorithm(Ying, Lee, & Lin, 2012), modified hybrid artificial bee colony algorithm (S.-W. Lin & Ying, 2014), immune inspired algorithm (Marinho Diana, de Franca Filho, de Souza, & de Almeida Vitor, 2015), improved combinatorial benders decomposition (Abreu Gomes & Mateus, 2017), improved firefly algorithm (Ezugwu & Akutsah, 2018), novel constraint programming approach (Gedik, Kalathia, Egilmez, & Kirac, 2018) and enhanced symbiotic organisms search algorithm (Ezugwu, 2019). For the mathematical model of UPMSP-SMDST with makespan minimization, Avalos-Rosales, Angel-Bello, and Alvarez (2015) proposed several mixed integer programming models, which are much better than previous

* Corresponding author at: No. 2 South Lushan Road, Yuelu District, Changsha, Hunan, 410082, China.
E-mail addresses: likezhang@hnu.edu.cn (L. Zhang), deng_arbeit@hnu.edu.cn (Q. Deng), dredhund@hnu.edu.cn (R. Lin), gongguiliang@hnu.edu.cn (G. Gong), forrest123@hnu.edu.cn (W. Han).

https://doi.org/10.1016/j.eswa.2021.114843
Received 18 December 2019; Received in revised form 29 September 2020; Accepted 2 March 2021
Available online 10 March 2021

models in solving instance size. Bozorgirad and Logendran (2012) investigated an extension of UPMSP-SMDST, in which jobs were clustered in different groups, setup times were only considered between groups, and job ready times and machine availability times were considered as well. They constructed an improved tabu search algorithm for their problem with the objective of minimizing the sum of total weighted completion time and total weighted tardiness. Costa, Cappadonna, and Fichera (2013) studied the UPMSP-SMDST with limited multi-skilled workers who were responsible to perform setup activities and developed a hybrid genetic algorithm to solve the problem. A similar model was proposed by Bektur and Sarac (2019), who considered a common server instead of multi-skilled workers to perform sequence-dependent setup activity. They were the first to use the tabu search algorithm for their problem with minimum total weighted tardiness. Y.-K. Lin and Hsieh (2014) researched the UPMSP-SMDST with constraints that different jobs had different ready times. They proposed an iterated hybrid metaheuristic for their problem whose objective was the minimum total weighted tardiness. Rauchecker and Schryen (2019) tackled the UPMSP-SMDST considering machine eligibility constraints with the criterion of minimum total weighted completion time. They proposed an exact branch and price algorithm for their problem, in which a parallelization strategy was incorporated to speed up the computing efficiency. Perez-Gonzalez, et al. (2019) added the machine eligibility and job due dates restrictions into UPMSP-SMDST to minimize the total tardiness. Several adaptive methods based on similar problem and five constructive heuristics were presented for their problem.

Apart from the single-objective algorithms, some studies have focused on multi-objective algorithms for the UPMSP-SMDST. For example, Torabi, Sahebjamnia, Mansouri, and Bajestani (2013) studied a multi-objective UPMSP-SMDST with three performance criteria of minimum total weighted flow time, minimum total weighted tardiness and minimum total machine load variation. They proposed a multi-objective particle swarm optimization algorithm for their model, in which they considered secondary resource constraints for jobs, uncertainty in processing time and due date. Xue, Rui, Yu, Sang, and Liu (2019) investigated a bi-objective UPMSP-SMDST considering machines' processing speed, whose criteria were to minimize both makespan and total carbon emission. They proposed an estimation of distribution evolution memetic algorithm for addressing the problem. M. Wang and Pan (2019) proposed a bi-objective UPMSP-SMDST considering preventive maintenance and designed a novel imperialist competitive algorithm to minimize makespan and total tardiness.

Through the review of related work on UPMSP-SMDST above, we observe that few researchers consider the limited worker resources and no previous work has studied the UPMSP-SMDST taking limited worker resources and learning effect into consideration simultaneously. Possible reasons for this have been discussed by some researchers. Othman, Gouw, and Bhuiyan (2012) outlined some reasons for ignoring human resources, such as humans are hard to be modeled since they are too abstract and people are adaptive. Lei and Guo (2014) stated that some traditional scheduling problems assumed that workers were plentiful and the labor cost was lower than that of machines, which contradicts reality. In the real production systems, the worker resources are the key elements which cannot be ignored in a production system, and the machines would be in idle state when they are waiting for setup server/workers, indicating that a well schedule on worker resources can effectively improve the production efficiency (Bektur & Sarac, 2019; Costa, et al., 2013; Huang, Cai, & Zhang, 2010). Some researchers, such as Bautista, Alfaro-Pozo, and Batalla-Garcia (2015), Touat, Bouzidi-Hassini, Benbouzid-Sitayeb, and Benhamou (2017), G. Gong, Chiong, Deng, and Gong (2019), Sheikhalishahi, Eskandari, Mashayekhi, and Azadeh (2019), Yepes-Borrero, Villa, Perea, and Pablo Caballero-Villalobos (2020) and G. Gong, et al. (2020), emphasized that the human factors should be considered in the manufacturing systems for their potential to improve production efficiency and decrease production cost, especially in highly-manual mixed-model assembly systems (Ostermeier, 2019). Some scholars have further pointed out that both processing time of jobs (Anzanello & Fogliatto, 2010; Hamta, Ghomi, Jolai, & Shirazi, 2013; C. Wu, Lee, & Liou, 2013; R. Wu, Li, Guo, & Xu, 2018) and setup times (Exposito-Izquierdo, Angel-Bello, Melian-Batista, Alvarez, & Baez, 2019; J.-B. Wang, et al., 2009) can be reduced attributing to the learning effect of workers. D. Biskup (1999) have pointed out that, along with the accumulating repetitions on the same machine, the workers' skills will continuously improve and their operating speed to perform setup activities will increase. He further claimed that it is more reasonable to apply learning effect to the setup times if production process is fully automated (Dirk Biskup, 2008). The latest study about learning effect on setup times was presented by Exposito-Izquierdo et al. (2019), but the human resource constraints were ignored.

Inspired by these findings, we originally considere the worker resources and their learning effects simultaneously into UPMSP-SMDST. This is significant to bridge the theoretical gaps and guide practical production. We denote this new model as NUPMSP throughout this paper. The proposed NUPMSP simultaneously considers the energy consumption and processing time related factors, which means that the proposed model can provide guidance of production for the managers aiming at decreasing production cost and protecting the environment. Given the effectiveness of evolutionary algorithms in solving NP-hard scheduling problems (Gao, et al., 2019; Soares & Carvalho, 2020; X. Wu & Che, 2019) and the characteristics of the proposed model, a new combinatorial evolutionary algorithm (CEA) is designed to solve the NUPMSP with two optimization objectives, in which some effective operators including an initialization operator with three new methods, a mutation operator and a selection operator are presented to help the CEA to obtain a high-quality population. In the experiment phase, a total of 72 NUPMSP benchmark instances are constructed; the effectiveness of the proposed initialization operator is verified based on 15 constructed benchmark instances; extensive experiment carried out verify that the proposed NUPMSP is superior to the nondominated neighbor immune algorithm (NNIA) (M. Gong, Jiao, Du, & Bo, 2008) and the non-dominated genetic algorithm-II (NSGA-II) (Deb, Pratap, Agarwal, & Meyarivan, 2002).

2. Mathematical programming model of the proposed NUPMSP

The model of NUPMSP can be described as follows. There are a set of $n+1$ independent jobs $N=\{J_0, J_1, \ldots, J_n\}$ where J_0 denotes a dummy job, a set of w multi-skilled workers $W=\{W_1, W_2, \ldots, W_w\}$ and a set of m ($m > w$) unrelated parallel machines $M=\{M_1, M_2, \ldots, M_m\}$ whose process is fully-automated. Each job except J_0 has to be processed on exactly one machine out of the set M and its processing time varies with machines. Before a job is processed, a basic setup time for the job is required, which is sequence and machine dependent. For the factor of workers, their duty is to perform setup activities and they are featured with different initial skill levels for different machines and with the same learning rate. η_{lk} is used as a coefficient which reflects the initial skill level of worker l on machine k and a is defined as the learning effect. This means that the setup times are also worker dependent. Due to the learning effect, the processing efficiency of a worker will increase, but it will stop when it reaches a given highest skill level d (Azzouz, Ennigrou, & Ben Said, 2018). For the factor of green indicator, the total energy consumption (TEC) of machines under different states is considered.

The objectives of the investigated problem are to minimize the maximum completion time among all jobs (C_{max}) and the TEC. The TEC mainly includes two parts: the energy consumption in processing stage of each job and the energy consumption in standby state including the setup stage for each operation. The scheduling consists of three sub-problems: a) assigning a machine for each job; b) determining the job sequence on each machine; c) selecting a worker for the setup activity of each job before processing.

Some assumptions are put forward:

1) Preemption is not allowed;
2) Each machine cannot process more than one job at a time;
3) Each worker can perform at most one setup activity at a time;
4) A machine is not allowed to turn off before all jobs allocated to the machine are finished;
5) All jobs, workers and machines are available at time zero;
6) A dummy job J_0 has been processed on each machine at time zero;
7) The actual setup time of J_i, which is performed by W_l and processed immediately after J_g on M_k, is calculated by formula (1) adapted from Dirk Biskup (2008).

$$Ts_{gikl} = \max\left\{\eta_{lk} s_{gik} A_{ilk}^{a},\ d{\cdot}s_{gik}\right\} \tag{1}$$

Where A_{ilk} is the number of times of W_l operating M_k before the machine processing J_i, d is the limiting coefficient of workers and s_{gik} denotes the basic setup time of J_i which is immediately processed after J_g on M_k.

Indices:

- i, j, g: indices of jobs, $i, j, g = 0, 1, \ldots, n$;
- k: index of machines, $k = 1, 2, \ldots, m$;
- l, h: index of workers, $l, h = 1, 2, \ldots, w$ ($w < m$);
- r: sequential index of setup activities performed by the same worker, where $r \leq n$ is a positive integer;
- t, e: position index on each machine, where $t = 0$ denotes the dummy position to assign the dummy job;

Parameters:

- n: total number of jobs;
- w: total number of workers;
- m: total number of machines;
- p_{ik}: the processing time of J_i on M_k;
- η_{lk}: the initial coefficient of W_l on M_k;
- a: the learning effect of workers;
- d: the limiting coefficient of workers;
- C_i: the completion time of J_i;
- CM_k: the maximum completion time of jobs processed on M_k;
- CS_i: the completion time of the setup activity of J_i;
- A_{ktl}: the number of times of W_l operating M_k before the machine processing the job on its tth position;
- s_{ijk}: the basic setup time of J_j on M_k, immediately after J_i has been processed on M_k;
- Ts_{ijkl}: the actual setup time of J_j performed by W_l if J_j is processed immediately after J_i on M_k;
- PP_k: the processing power of M_k;
- SP_k: the standby power of M_k;
- B: a big enough number;

Decision variables:

$$x_{iktl} = \begin{cases} 1 \text{ if } J_i \text{ is assigned to position } t \text{ on } M_k \\ \text{and its setup activity is performed by } W_l \\ 0 \text{ otherwise} \end{cases}$$

$$z_{ilr} = \begin{cases} 1 \text{ if the setup activity of } J_i \text{ is processed} \\ \text{in the } rth \text{ position on worker } l \\ 0 \text{ otherwise} \end{cases}$$

The mathematical models for UPMSP-SMDST proposed by Avalos-Rosales, et al. (2015) and Abreu Gomes and Mateus (2017) provide a great reference for us to build the model. By referring to these models, the mathematical model of the proposed problem can be described as follows:

$$min\ C_{max} \tag{2}$$

$$min\ TEC = \sum_{k=1}^{m}\sum_{l=1}^{w}\sum_{i=1}^{n}\sum_{t=1}^{n} x_{iktl} p_{ik} PP_k + \sum_{k=1}^{m}\left[CM_k - \sum_{l=1}^{w}\sum_{i=1}^{n}\sum_{t=1}^{n} x_{iktl} p_{ik}\right] SP_k \tag{3}$$

Subject to:

$$\sum_{k=1}^{m}\sum_{l=1}^{w}\sum_{t=1}^{n} x_{iktl} = 1, \forall i = 1, 2, \ldots, n \tag{4}$$

$$\sum_{l=1}^{w}\sum_{i=1}^{n} x_{iktl} \leqslant 1, \forall t = 1, 2, \ldots, n; k = 1, 2, \ldots m \tag{5}$$

$$\sum_{l=1}^{w}\sum_{i=1}^{n} x_{ik(t+1)l} \leqslant \sum_{l=1}^{w}\sum_{i=1}^{n} x_{iktl}, \quad \forall t = 1, 2, \ldots, n-1; k = 1, 2, \ldots, m \tag{6}$$

$$\sum_{l=1}^{w}\sum_{r=1}^{n} z_{ilr} = 1, \forall i = 1, 2, \ldots, n \tag{7}$$

$$\sum_{i=1}^{n} z_{ilr} \leqslant 1, \forall l = 1, 2, \ldots, w; r = 1, 2, \ldots, n \tag{8}$$

$$\sum_{i=1}^{n} z_{ilr} \geqslant \sum_{i=1}^{n} z_{il(r+1)}, \forall l = 1, 2, \ldots, w; r = 1, 2, \ldots, n-1 \tag{9}$$

$$\sum_{r=1}^{n} z_{ilr} = \sum_{k=1}^{m}\sum_{t=1}^{n} x_{iktl}, \ \forall i = 1, 2, \ldots, n; l = 1, 2, \ldots, w \tag{10}$$

$$C_g - \mathrm{CS}_g \geqslant \sum_{k=1}^{m}\sum_{l=1}^{w}\sum_{t=1}^{n} x_{gktl} p_{gk}, \forall g = 1, 2, \ldots, n \tag{11}$$

$$A_{ktl} = \max\left\{\sum_{i=1}^{n}\sum_{e=1}^{t} x_{ikel},\ 1\right\}, \quad \forall k = 1, 2, \ldots, m; t = 1, 2, \ldots, n; l = 1, 2, \ldots, w \tag{12}$$

$$Ts_{jikl} \geqslant \max\left\{\eta_{lk} s_{jik} (A_{ktl})^{a},\ d{\cdot}s_{jik}\right\} + B\left(x_{jk(t-1)h} + x_{iktl} - 2\right), \quad \forall i, t = 1, 2, \ldots, n; j = 0, 1, \ldots, n; i \neq j; k = 1, 2, \ldots, m; h, l = 1, 2, \ldots, w \tag{13}$$

$$CS_g - C_i \geqslant Ts_{igkl} + B\left(x_{ik(t-1)h} + x_{gktl} - 2\right), \forall i = 0, 1, \ldots, n; \quad g, t = 1, 2, \ldots, n; i \neq g; k = 1, 2, \ldots, m; h, l = 1, 2, \ldots, w \tag{14}$$

$$CS_i - CS_g \geqslant Ts_{jikl} + B\left(z_{il(r+1)} + z_{glr} + x_{jk(t-1)h} + x_{iktl} - 4\right) \quad \forall i, g, t = 1, 2, \ldots, n; j = 0, 1, \ldots n; i \neq j; i \neq g; r = 1, 2, \ldots, n-1; k = 1, 2, \ldots, m; l, h = 1, 2, \ldots, w; \tag{15}$$

$$C_{max} \geqslant C_i, \ \forall i = 1, 2, \ldots, n \tag{16}$$

$$CM_k \geqslant C_j + B\left(\sum_{l=1}^{w}\sum_{t=1}^{n} x_{jktl} - 1\right), \quad \forall j = 1, 2, \ldots, n; k = 1, 2, \ldots, m \tag{17}$$

$$C_0 = 0, x_{0k0l} = 1, \ Ts_{ijkl} \geqslant 0, \forall i = 1, 2, \ldots n; \quad j = 0, 1, \ldots n; i \neq j; k = 1, 2, \ldots, m; l = 1, 2, \ldots, w \tag{18}$$

Equations (2)-(3) represent the objectives of minimizing makespan and *TEC* respectively. Constraint (4) ensures that each job is processed by only one machine with one position, and only one worker is selected to perform its setup activity. Constraint (5) ensures that at most one job is assigned to any position on a machine. Constraint (6) ensures that no vacant position exists before a filled position on a machine. Constraint (7) guarantees the setup activity of each job is performed at only one position on one worker. Constraint (8) ensures that at most one setup activity can be assigned to any position of any worker. Constraint (9) guarantees that there is no vacant position before a filled position of the

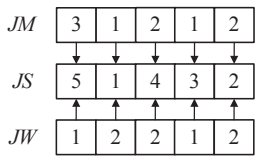

Fig. 1. An example of the solution representation.

same worker. Constraint (10) guarantees the relationship of z_{ilr} and x_{iktl}, which ensures that for any job J_i, only one worker preforms its setup activity. Constraint (11) ensures that the completion time of a job is no less than the completion time of its setup activity plus the processing time of the job. Constraint (12) calculates the value of A_{ktl} and defines its minimum value as 1 to ensure that $(A_{ktl})^a$ in constraint (13) is meaningful. Constraint (13) defines the lower bound of the actual setup time of J_i on M_k. Constraint (14) ensures that the gap between the completion time of the setup activity of a job and the completion time of the job's predecessor is no less than the setup time of the job. Constraint (15) guarantees the sequence constraint of setup activities processed by the same worker. Constraint (16) determines the makespan. Constraint (17) determines the maximum completion time of jobs assigned to each machine. Constraint (18) is sign constraints.

3. The proposed algorithm

In this section, we design a CEA for our NUPMSP. The main components of the CEA include four parts: solution representation, initialization, mutation and selection. The details of CEA are described as follows.

3.1. Solution representation

A feasible solution of CEA for NUPMSP should solve three subproblems: a) select a machine for each job; b) arrange the processing sequence of all jobs processed on the same machine; c) choose a suitable worker for each job's setup activity. Thereby, we propose an effective solution representation method with three vectors: job sequence (*JS*) vector, job-machine (*JM*) vector and job-worker (*JW*) vector. An example of this solution representation method is shown is Fig. 1, in which there are five jobs, three machines and two workers. The job, machine and worker sequences shown in Fig. 1 are as follows: (J_5, M_3, W_1), (J_1, M_1, W_2), (J_4, M_2, W_2), (J_3, M_1, W_1) and (J_2, M_2, W_2), where (J_5, M_3, W_1) means that J_5 is processed on M_3 and its setup activity is executed by W_1; others can be explained like this.

3.2. Initialization

To produce high quality initial solutions, we used three methods: a) list scheduling (*LS*) heuristic proposed by Davis and Jaffe (1981); b) shortest setup time first (*SST*) rule (Pindeo, 2002); c) a earliest completion time first (*ECT*) rule designed by us. The *LS* heuristic is used to select a suitable machine for each job, *SST* rule is introduced to optimize the job sequence on each machine, and *ECT* rule is designed to choose an appropriate worker for the setup activity of each job processed on each machine. A complete initial solution is generated through four steps as follows:

Step 1: Randomly generate a *JS* vector.
Step 2: For the *JS* vector, use the *LS* heuristic to select a machine for each job.
Step 3: Optimize the job sequence on each machine by *SST* rule.
Step 4: Carry out the *ECT* rule to determine a worker by which each job's setup activity is executed, to obtain the processing order shown as Fig. 1 and to compute the objectives.

Fig. 2 illustrates the steps about how to produce a high quality initial solution based on the given data in Tables 1 and 2. Fig. 3 is a Gant chart based on the result in Fig. 2.

3.2.1. LS heuristic

The *LS* heuristic is popular for assigning jobs effectively among several machines (X. Wu & Che, 2019). It depends on certain principle to select a machine for each job. Considering that the parallel machines are unrelated and one of the optimization objectives is makespan, we introduce a principle named processing efficiency to reflect the processing ability of each machine.

Principle 1: processing efficiency. We denote Te_{ik} as the processing efficiency of J_i on M_k, where $Te_{ik}=(\min_{1\le k\le m} p_{ik})/p_{ik}$.

Detailed steps of the *LS* heuristic used in this paper are depicted as follows:

Step 1: For each *JS* vector, set $CM_k = 0$ for each M_k ($k = 1, 2, \ldots, m$) and calculate Te_{ik} for each job on each machine according to the job sequence in *JS*.
Step 2: Create a list l_k for each M_k by sorting all the unassigned jobs in non-increasing order of the values of Te_{ik}.
Step 3: If $|JS|>0$, find $M_{\tilde{k}} := \arg min_{1\le k\le m} CM_k$ (randomly select a machine if $min_{1\le k\le m} CM_k$ exists on multiple machines) and find out the next unassigned job $J_{\tilde{i}}$ in $l_{\tilde{k}}$; otherwise, end the *LS* heuristic for this *JS* vector.
Step 4: If $Te_{\tilde{i}\tilde{k}} > 1/\sqrt{m}$, allocate $M_{\tilde{k}}$ to process $J_{\tilde{i}}$ and set $CM_{\tilde{k}} := CM_{\tilde{k}} + p_{\tilde{i}\tilde{k}}$; otherwise, assign the big value B to $C_{\tilde{k}}$ and return to step 3.
Step 5: Set $l_k := l_k \backslash \{J_{\tilde{i}}\}$ for $1 \le k \le m$ and set $JS := JS \backslash \{J_{\tilde{i}}\}$; then return to step 3.

3.2.2. SST rule

SST rule selects a job with the shortest setup time as the next job to be processed when the current job is finished, until all jobs have been assigned. It is similar to the nearest neighbor rule in Travelling Salesman Problem (*TSP*), which can effectively get a reasonable route. Due to the fact that both of the objectives can be minimized by reducing the total setup times and workers have the learning ability, the *SST* heuristic can

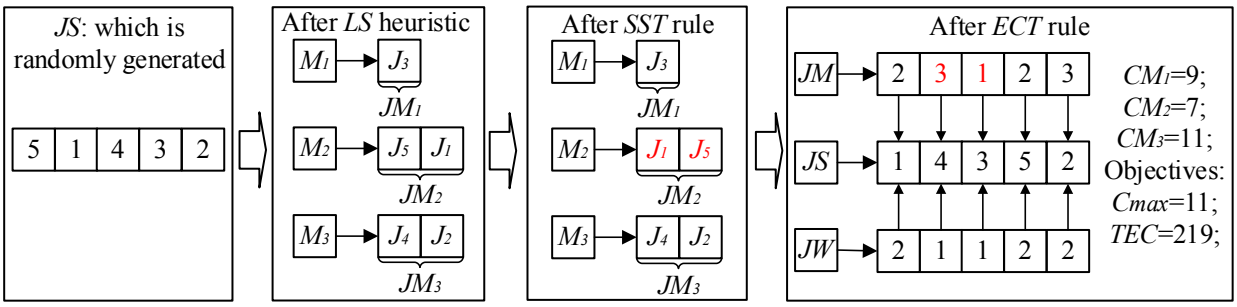

Fig. 2. Illustration of the Initialization procedure based on the data shown in Tables 1 and 2.

Table 1
Relevant data of an example.

M_k	p_{ik} $i=1$	$i=2$	$i=3$	$i=4$	$i=5$	η_{lk} $l=1$	$l=2$	a	d	PP_k	SP_k
$k=1$	4	5	2	8	3	0.75	1.5	−0.152	0.5	11	5
$k=2$	2	6	5	4	1	1.25	0.5			16	2
$k=3$	5	3	7	2	6	1	1			14	6

Table 2
The setup times of the example shown in Table 1.

	s_{ijk} $(k=1)$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	s_{ijk} $(k=2)$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	s_{ijk} $(k=3)$ $i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$j=1$	2	0	7	3	1	4	3	0	1	5	7	6	2	0	1	2	5	4
$j=2$	5	4	0	8	4	1	6	5	0	5	7	2	6	5	0	1	2	5
$j=3$	4	6	5	0	3	1	5	6	7	0	2	8	5	2	4	0	6	3
$j=4$	8	4	6	1	0	2	1	5	2	7	0	3	4	5	2	6	0	1
$j=5$	7	2	1	8	5	0	4	5	8	1	6	0	7	5	3	2	1	0

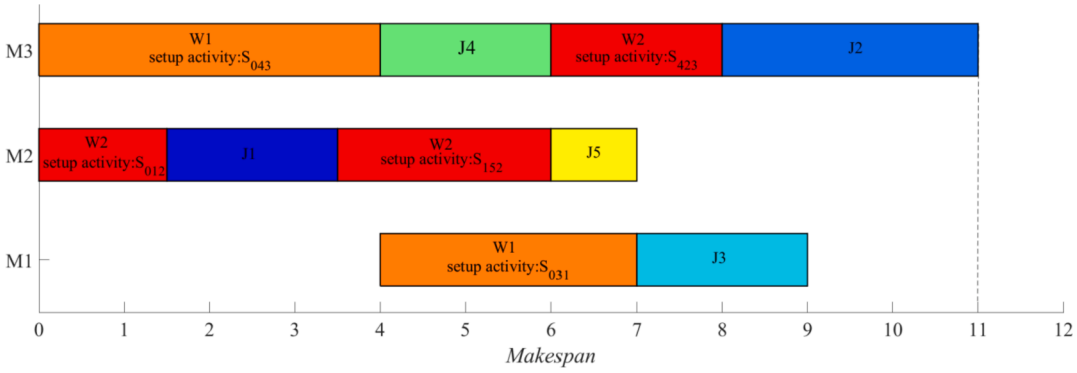

Fig. 3. The Gant chart of the solution in Fig. 2.

have a positive impact on our performance criteria. Detailed steps of the *SST* rule used in this paper are described as below:

Step 1: We denote JM_k ($k=1, 2, \ldots, m$) as the set of all jobs assigned to M_k, and assume a dummy job J_i ($i=0$) has been processed on M_k ($k=1, 2, \ldots, m$). Then, set $k=1$.

Step 2: If $k \leq m$, go to step 3; otherwise, end the *SST* rule.

Step 3: If $|JM_k|>0$, select a job $J_j \in JM_k$ with the minimum s_{ijk} and put it immediately after J_i. Then set $JM_k := JM_k \backslash \{J_j\}$ and $i:=j$. This means that J_j will be processed immediately after J_i and J_j will substitute J_i to choose a successor from JM_k. $i=0$ means J_j will be the first job processed on M_k.

Step 4: If $|JM_k|=0$, set $k:=k+1$ and return to step 2; otherwise, return to step 3.

3.2.3. ECT rule

After *LS* heuristic and *SST* rule, allocation of jobs to machines and jobs' sequence on each machine have been determined. To get a feasible solution shown as Fig. 1, we should select a worker for each job's setup activity and determine the processing order of all jobs. *ECT* rule is designed to choose an appropriate worker which can minimum the completion time of the current job, and to determine the scheduling sequence. Before detailing the steps of *ECT* rule, some parameters are introduced: RM_k denotes the ready time of M_k to process a new job; RW_l represents the ready time of W_l to perform a new setup activity; μ_k is the index of the current job just completed on M_k; CM_k represents the completion time of M_k. Detailed description of *ECT* rule is as follows:

Step 1: Input the results obtained by *SST* rule. We denote JM_k ($k=1, 2, \ldots, m$) as the set of all jobs assigned to M_k and define S as the index of job's position in a scheduling sequence.

Step 2: Set $RM_k=0$ and $\mu_k=0$ for $1 \leq k \leq m$, $RW_l=0$ for $1 \leq l \leq w$ and $S=1$.

Step 3: If $RM_k=B$ (B is a large enough number) holds for each $k \in \{1, 2, \ldots, m\}$, end the *ECT* rule and output the solution including its two objectives and CM_k for $1 \leq k \leq m$; otherwise, go to step 4.

Step 4: Find $M_{\tilde{k}} := \arg min_{1 \leqslant k \leqslant m} RM_k$ (randomly select a machine if $min_{1 \leq k \leq m} RM_k$ exists on multiple machines).

Step 5: If $|JM_{\tilde{k}}|>0$, find out the first job $J_{\tilde{i}}$ in $JM_{\tilde{k}}$ and set $JM_{\tilde{k}} := JM_{\tilde{k}} \backslash \{J_{\tilde{i}}\}$. Otherwise, set $CM_{\tilde{k}} := RM_{\tilde{k}}$, assign B to $RM_{\tilde{k}}$ and return to step 3.

Step 6: Calculate the completion time $C_{\tilde{i}}$ of $J_{\tilde{i}}$ under each worker by formula (19), then select a worker $W_{\tilde{l}}$ which can minimize the completion time of $J_{\tilde{i}}$.

Step 7: Update $RM_{\tilde{k}}$ and $RW_{\tilde{l}}$ by formulas (20) and (21) respectively, and set $u_{\tilde{k}} := \widetilde{i}$.

Step 8: Put $J_{\tilde{i}}$ in the position S and set $S:=S+1$. Then return to step 3.

$$C_{\tilde{i}} = \max\{RM_{\tilde{k}}, \mathrm{RM}_{\tilde{l}}\} + \max\left\{\eta_{\tilde{l}\tilde{k}} s_{\mu_{\tilde{k}}\tilde{i}\tilde{k}} \left(\sum_{j=0, j\neq i}^{n} y_{j\tilde{i}\tilde{k}\tilde{l}}\right)^{a},\ d \cdot s_{\mu_{\tilde{k}}\tilde{i}\tilde{k}}\right\} \tag{19}$$

$$RM_{\tilde{k}} = C_{\tilde{i}} \tag{20}$$

$$\mathrm{RW}_{\tilde{l}} = C_{\tilde{i}} - p_{\tilde{i}\tilde{k}} \tag{21}$$

3.3. Mutation

Due to the rigid initialization procedure, the searching space cannot contain all the solution space and is easy to fall into local optimum. Thus, two mutation operators are designed to improve the diversity. One is called machine-oriented mutation which is based on machine load

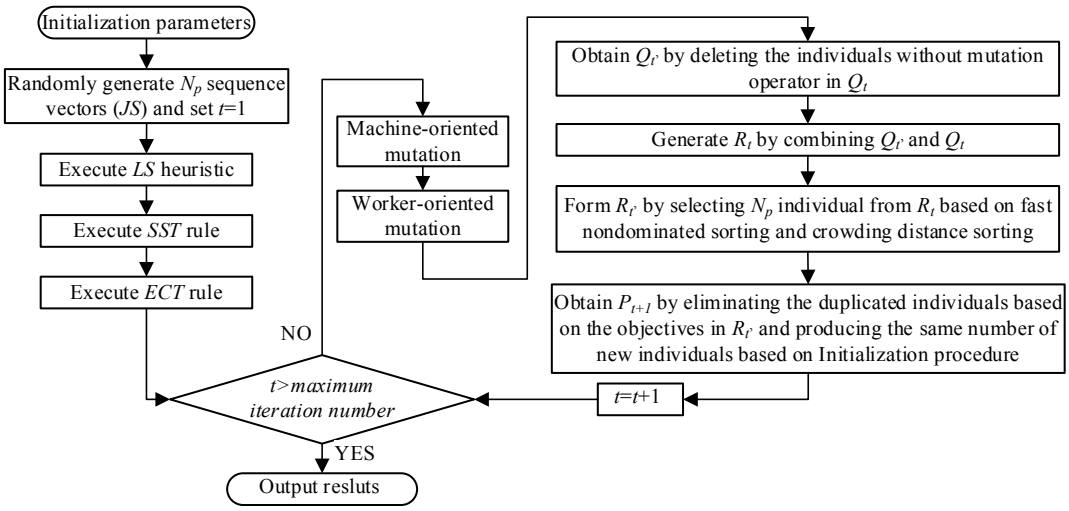

Fig. 4. The procedure of CEA.

balance and is similar to external insertion, the other is defined as worker-oriented mutation.

The detailed procedures of machine-oriented mutation and worker-oriented mutation are described in Algorithm 1 and Algorithm 2 respectively. It should be noted that the meanings of CM_k, JM_k, RM_k, RW_i, μ_k, S and B here refer to Section 3.2.3.

Algorithm 1: Machine-oriented mutation

1. **For** each individual *ind* **do**
2. Generate a random number between 0 and 1, denoted by R_{n1};
3. **If** $R_{n1} < p_{m1}$ **do** % p_{m1} is the machine-oriented mutation rate.
4. Calculate the fitness of each machine k in individual *ind* based on the equation $f(k)=CM_k/\sum_{k=1}^{m} CM_k$;
5. Select a machine denoted as M_k via the roulette wheel rule based on $f(k)$;
6. Randomly select a machine which is different from M_k and denote it as $M_{k'}$;
7. Randomly select a job from JM_k and reallocate the job to $JM_{k'}$;
8. Use the SST rule in Section 3.2.2 to optimize the job sequence in JM_k and $JM_{k'}$;
9. Use the ECT rule in Section 3.2.3 to reselect a worker for each job on each machine in individual *ind*;
10. **End If**
11. **End For**

Algorithm 2: Worker-oriented mutation

1. **For** each individual *ind* **do**
2. Generate a random number between 0 and 1, denoted by R_{n2};
3. **If** $R_{n2} < p_{m2}$ **do** % p_{m2} is the worker-oriented mutation rate.
4. Calculate the fitness of each machine in individual *ind* based on the equation $f(k)=CM_k/\sum_{k=1}^{m} CM_k$;
5. Select a machine denoted as M_k via the roulette wheel rule based on $f(k)$;
6. Select a machine, which is different from M_k and denote it as $M_{k'}$, via the roulette wheel rule based on the value $(1-f(k))/\sum_{k=1}^{m}(1-f(k))$ of each machine k;
7. Randomly select a job from JM_k and randomly reassign a worker whose initial skill level is in the top half of all the workers to this job. Randomly select a job from $JM_{k'}$ and randomly reassign a worker whose initial skill level is in the latter half of all the workers to this job;
8. Set $RM_k = 0$ and $\mu_k = 0$ for $1 \le k \le m$, and set $S = 1$;
9. **while** $RM_k = B$ is not hold for all machines **do**
10. Find $\tilde{k} := \arg min_{1\le k\le m} RM_k$ (randomly select a machine if $min_{1\le k\le m} RM_k$ exists on multiple machines);
11. **If** $|JM_{\tilde{k}}|>0$ **do**
12. Find out the first job $J_{\tilde{i}}$ in $JM_{\tilde{k}}$ and the worker $\tilde{l}$ corresponding to job $J_{\tilde{i}}$. Then put $J_{\tilde{i}}$ in the position S, update $RM_{\tilde{k}}$ and $RW_{\tilde{l}}$ by formulas (20) and (21) respectively, $JM_{\tilde{k}} := JM_{\tilde{k}} \backslash \{J_{\tilde{i}}\}$, $\mu_{\tilde{k}} := \tilde{l}$ and $S := S + 1$.
13. **Else**
14. Set $CM_{\tilde{k}} := RM_{\tilde{k}}$ and assign B to $RM_{\tilde{k}}$;
15. **End If**
16. **End while**
17. **End If**
18. **EndFor**

3.4. Selection

After mutation operator for the parent population, the offspring population is created. Then we should select individuals to compose a new population as the parent population of the next generation. In this paper, the selection operator is similar to elitist strategy used in NSGA-II. Suppose that P_t is tth generation's parent population and Q_t is the offspring population of P_t, where the size of both populations is N_p. To generate the next population P_{t+1} and conserve high-quality solutions at the same time, we should combine P_t and Q_t. However, some individuals without mutation operate exist double after this combination, which will reduce the diversity of the population. Here, an improved combination strategy is proposed and the detailed steps of the selection operator are described as follows:

Step 1: Delete the individuals without mutation operator in Q_t and denote the remaining individuals as the set of $Q_{t'}$.
Step 2: Generate a combined population $R_t = Q_t \cup Q_{t'}$.
Step 3: Use the fast nondominated sorting and crowding distance sorting to select N_p individuals from R_t, and denote the set of N_p individuals as $R_{t'}$.
Step 4: Use the Initialization procedure described in Section 3.2 to generate new individuals, whose number is equal to that of the duplicated individuals in $R_{t'}$, to substitute the duplicated individuals based on the objectives in $R_{t'}$.
Step 5: Form P_{t+1} by assigning $R_{t'}$ to P_{t+1}.

3.5. The procedure of CEA

According to the descriptions above, the overall procedure of CEA can be depicted as Fig. 4.

4. Experiments and results

In this section, three types of experiments are designed to evaluate the performance of CEA for NUPMSP. The first type of experiment is Taguchi analysis, which is used to select the suitable combination of critical parameters; the second type of experiment is carried out to

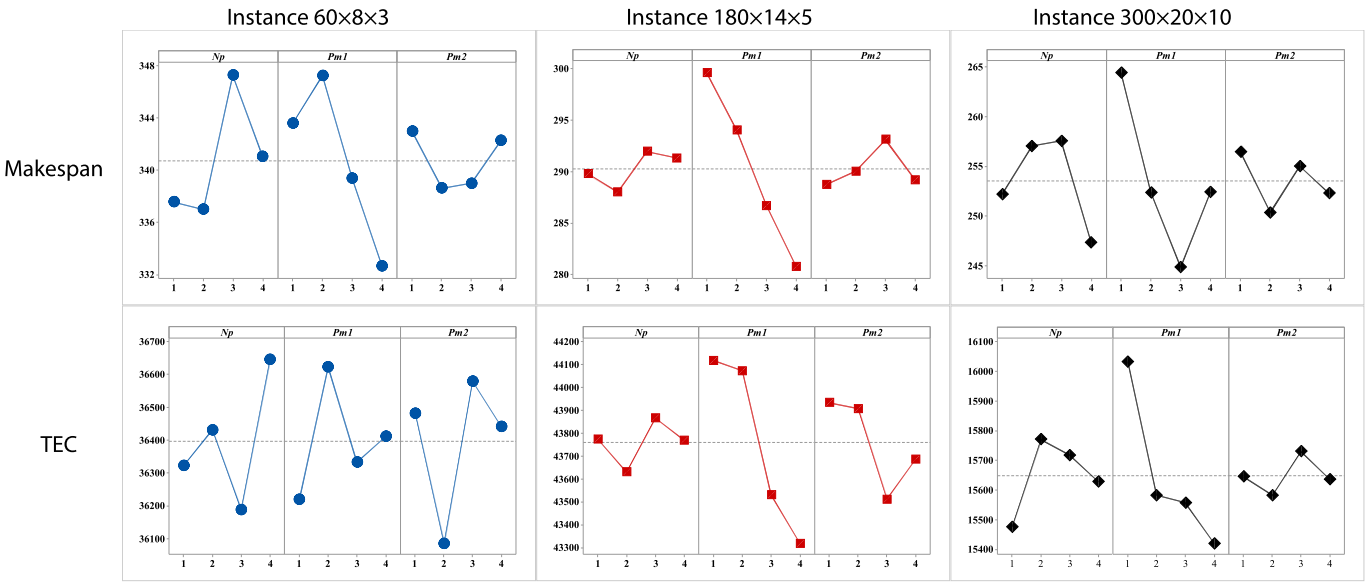

Fig. 5. Factor level trend of key parameters.

evaluate the validity of *LS* heuristic, *SST* rule and *ECT* rule; the last type of experiment is to test our CEA with other well-known algorithms, i.e., NNIA and NSGA-II. Note that all the algorithms here are coded in MATLAB R2016a and run on a computer with Inter Core i9-9900 k CPU at 3.60 GHz and 32 GB RAM.

4.1. Instances construction

In this section, 72 instances are constructed for testing and the source data of each instance are generated as follows. The processing time and the initial sequence dependent setup time of each job processed on each machine are integers and are both generated based on method proposed by Costa, et al. (2013). For the initial coefficient of each worker, η_{lk} is randomly generated from the set {0.5, 0.75, 1, 1.25, 1.5}. In this paper, the learning effect a of workers is the same and set $a = -0.152$ according to C. Wu, et al. (2013). The processing power and the idle power of each machine follow a uniform distribution [20, 40] and [10, 20] respectively. In the experiments, each instance is named as $n \times m \times w$, where n denotes the number of jobs, m denotes the number of machines and w denotes the number of workers. Inspired by Zhang, Deng, Gong, and Han (2019), the instances are divided into three scales: (a) small-scale instances where $n\in\{40, 60, 80, 100\}$, $m\in\{6, 8, 10\}$ and $w\in\{3, 4\}$; (b) middle-scale instances where $n\in\{140, 160, 180, 200\}$, $m\in\{12, 14, 16\}$ and $w\in\{5, 6\}$; (c) large-scale instances where $n\in\{250, 300, 350, 400\}$, $m\in\{18, 20, 22\}$ and $w\in\{8, 10\}$. All the data can be download via the link: https://github.com/likezhang1990/NUPMSP-instances.git.

4.2. Performance metrics

In this paper, three well-known multi-objective metrics are considered to evaluate the performance of all tested algorithms. They are the C metric, the hypervolume indicator (HV) and the inverted generational distance (IGD), which can be depicted as [1]-[3].

[1] The value of $C(A, B)$ represents the percentage of solutions in set B which are dominated by at least one solution in set A, i.e.,

$$C(A,B) = \frac{|\{x \in B | \exists y \in A : y \text{ dominates } x\}|}{|B|} \tag{22}$$

Therefore, the larger the $C(A, B)$ is, the better A is.

[2] The HV indicator is a performance metric for indicating the quality of a non-dominated approximation set. It can be defined as follows:

$$HV(ref, A) = \lambda\left(\bigcup_{f\in A}[f_1, ref_1] \times \cdots \times [f_m, ref_m]\right) \tag{23}$$

Where $HV(ref, A)$ represent the size of the space covered by set A, $R=(r_1, r_2, \ldots r_m)^T$ is reference point chosen from the target space, λ is the Lebesgue measure and m here refers to the number of targets. The higher $HV(ref, A)$ is, the better A is.

[3] The IGD of set A can be obtained by equation (24)

$$IGD(A, P^*) = \frac{1}{|P^*|}\sum_{x\in P^*} min_{y\in A}\, d(x, y) \tag{24}$$

Where $d(x, y)$ is the Euclidean distance between point $\times$ and y, and P^* is the Pareto optimal front. Here P^* is acquired by dealing with all the solutions obtained by all runs of all algorithms. The smaller $IGD(A, P^*)$ is, the closer A is to the Pareto optimal front. Besides, the objectives of all solutions are normalized by equation (25).

$$\tilde{f}(x) = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}},\ i = 1, 2 \tag{25}$$

f_i^{min} and f_i^{max} are the minimal and maximal value of ith objective in all solutions, respectively.

4.3. Analysis of experiments results

4.3.1. Parameter tuning

The proposed CEA has three key parameters, i.e., population size N_p, machine-oriented mutation probability p_{m1} and worker-oriented mutation probability p_{m2}. To select a suitable combination of these key parameters, the Taguchi method is carried out based on a small-scale instance $60 \times 8 \times 3$, a middle-scale instance $180 \times 14 \times 5$ and a large-scale instance $300 \times 20 \times 10$. Each key parameter is considered to have four factor levels, which are N_p={50, 100, 150, 200}, p_{m1}={0.1, 0.2, 0.3, 0.4} and p_{m2}={0.1, 0.2, 0.3, 0.4}. For each combination, the CEA is run independently five times and the mean value of each objective of the Pareto optimal front obtained by these runs is selected to analyze the effect of this combination.

Based on the experiment results, the trend of each factor level is depicted in Fig. 5. In Fig. 5, the first row and the second row denote the effect of N_p, p_{m1} and p_{m2} on makespan and TEC respectively.

From Fig. 5, we observe there is no distinct regularity between N_p and objectives. For the value of the optimization objectives, however,

Table 3
The comparison results of CEA and EA1.

Benchmark	*C*		*HV*		*IGD*		time/s	
	C(CEA, EA1)	*C*(EA1, CEA)	CEA	EA1	CEA	EA1	CEA	EA1
40 × 8 × 3	**1**	0	**0.176401**	0.057487	**0**	0.413059	28.07	**26.79**
60 × 6 × 4	**1**	0	**0.139488**	0.023493	**0**	0.773187	36.11	**35.12**
80 × 6 × 3	**1**	0	**0.192715**	0.001978	**0**	1.207774	47.75	**43.45**
80 × 10 × 4	**1**	0	**0.315037**	0.043207	**0**	0.815153	46.95	**41.32**
100 × 10 × 3	**1**	0	**0.264091**	0.002975	**0**	1.165858	52.85	**51.76**
140 × 14 × 5	**1**	0	**0.406864**	0.022778	**0**	1.018782	**69.51**	71.27
160 × 12 × 6	**1**	0	**0.273848**	0.011369	**0**	1.120342	81.94	**73.01**
180 × 12 × 5	**1**	0	**0.360152**	0.003832	**0**	1.241018	**80.92**	83.72
180 × 16 × 6	**1**	0	**0.348497**	0.002808	**0**	1.210553	89.50	**88.95**
200 × 16 × 5	**1**	0	**0.447582**	0.004167	**0**	1.250573	**97.81**	101.28
250 × 20 × 8	**1**	0	**0.370035**	0.000648	**0**	1.305765	119.84	**116.65**
300 × 18 × 10	**1**	0	**0.372183**	0.001314	**0**	1.265421	157.07	**145.07**
350 × 18 × 8	**1**	0	**0.447837**	0.004798	**0**	1.272546	179.14	**163.15**
350 × 22 × 10	**1**	0	**0.428368**	0.001705	**0**	1.309948	190.08	**167.50**
400 × 22 × 8	**1**	0	**0.483262**	0.001755	**0**	1.323147	223.86	**184.51**

Note: for each instance, significantly better results are marked in bold.

Table 4
The comparison results of CEA and EA2.

Benchmark	*C*		*HV*		*IGD*		time/s	
	C(CEA, EA2)	*C*(EA2, CEA)	CEA	EA2	CEA	EA2	CEA	EA2
40 × 8 × 3	**1**	0	**0.120065**	0.088115	**0.002606**	0.167001	28.07	**26.57**
60 × 6 × 4	**1**	0	**0.057447**	0.015560	**0**	0.508432	36.11	**31.62**
80 × 6 × 3	**1**	0	**0.032564**	0.007041	**0**	0.694190	47.75	**41.98**
80 × 10 × 4	**1**	0	**0.104024**	0.014373	**0**	0.707594	46.95	**41.51**
100 × 10 × 3	**1**	0	**0.067853**	0.010064	**0**	0.746158	52.85	**45.35**
140 × 14 × 5	**1**	0	**0.152455**	0.015612	**0**	0.850961	69.51	**64.43**
160 × 12 × 6	**1**	0	**0.144147**	0.010009	**0**	0.994372	81.94	**72.58**
180 × 12 × 5	**1**	0	**0.119025**	0.000000	**0**	1.154796	80.92	**77.32**
180 × 16 × 6	**1**	0	**0.172598**	0.009402	**0**	1.008510	89.50	**81.38**
200 × 16 × 5	**1**	0	**0.095296**	0.002898	**0**	1.022423	97.81	**82.59**
250 × 20 × 8	**1**	0	**0.181377**	0.001075	**0**	1.184056	119.84	**103.82**
300 × 18 × 10	**1**	0	**0.196716**	0.000912	**0**	1.215303	157.07	**118.40**
350 × 18 × 8	**1**	0	**0.186998**	0.009013	**0**	1.071018	179.14	**148.29**
350 × 22 × 10	**1**	0	**0.194492**	0.007340	**0**	1.108243	190.08	**139.31**
400 × 22 × 8	**1**	0	**0.153989**	0.002843	**0**	1.123926	223.86	**155.99**

Note: for each instance, significantly better results are marked in bold.

half of the six subgraphs get the minimum value at level 2 of N_p, which is higher than any other levels of N_p. Thus, we select level 2 of N_p (i.e., $N_p = 100$) for CEA. For p_{m1}, it is obvious to select level 4 of p_{m1} (i.e., $p_{m1} = 0.4$) for CEA. From Fig. 5, we see that the optimization target value decreases as p_{m1} increases in general, and for the value of the optimization objectives, two-thirds of the six subgraphs get the minimum value at level 4 of p_{m1}. For p_{m2}, we observe from Fig. 5 that the optimization target value is most likely to get the minimum value at level 2 of p_{m2}. Thereby, $p_{m2} = 0.2$ is considered for our CEA.

4.3.2. Effect of LS, SST and ECT

According to the description in Section 3, mainly three constructions contribute to CEA, which are the *LS*, *SST* and *ECT*. To prove the effectiveness of *LS*, *SST* and *ECT*, we construct other three algorithms, i.e.,

Table 5
The comparison results of CEA and EA3.

Benchmark	*C*		*HV*		*IGD*		time/s	
	C(CEA, EA3)	*C*(EA3, CEA)	CEA	EA3	CEA	EA3	CEA	EA3
40 × 8 × 3	**1**	0	**0.223396**	0.087697	**0**	0.368333	28.07	**26.94**
60 × 6 × 4	**1**	0	**0.152893**	0.015498	**0**	0.835445	36.11	**34.55**
80 × 6 × 3	**1**	0	**0.113997**	0.001875	**0**	1.116081	47.75	**42.58**
80 × 10 × 4	**1**	0	**0.278756**	0.020173	**0**	0.956209	46.95	**42.67**
100 × 10 × 3	**1**	0	**0.282095**	0.012955	**0**	1.066538	52.85	**50.43**
140 × 14 × 5	**1**	0	**0.385360**	0.011565	**0**	1.107172	69.51	**60.34**
160 × 12 × 6	**1**	0	**0.376515**	0.003442	**0**	1.214702	81.94	**67.64**
180 × 12 × 5	**1**	0	**0.471445**	0.008561	**0**	1.203914	80.92	**79.11**
180 × 16 × 6	**1**	0	**0.383156**	0.005554	**0**	1.166581	89.50	**72.42**
200 × 16 × 5	**1**	0	**0.464630**	0.010964	**0**	1.062974	97.81	**89.16**
250 × 20 × 8	**1**	0	**0.520434**	0.023001	**0**	1.077978	119.84	**118.36**
300 × 18 × 10	**1**	0	**0.646062**	0.026897	**0**	1.135166	157.07	**142.98**
350 × 18 × 8	**1**	0	**0.555958**	0.000985	**0**	1.302212	179.14	**167.10**
350 × 22 × 10	**1**	0	**0.640335**	0.009807	**0**	1.219025	190.08	**166.65**
400 × 22 × 8	**1**	0	**0.588502**	0.010950	**0**	1.206066	223.86	**178.44**

Note: for each instance, significantly better results are marked in bold.

Table 6
The *C*, *HV* and *IGD* comparison of CEA, NNIA and NSGA-II.

instance	*C*				*HV*			*IGD*			time/s
	C(CEA, NNIA)	*C*(NNIA, CEA)	*C*(CEA, NSGA-II)	*C* (NSGA-II,CEA)	CEA	NNIA	NSGA-II	CEA	NNIA	NSGA-II	CEA
40 × 6 × 3	**1**	0	**1**	0	**0.501403**	0.174997	0	**0**	1.001701	2.067899	27.82
40 × 6 × 4	**1**	0	**1**	0	**0.541345**	0.221338	0.005336	**0**	1.144825	2.793700	29.99
40 × 8 × 3	**1**	0	**1**	0	**0.614526**	0.240467	0.011311	**0**	1.114489	2.406813	27.46
40 × 8 × 4	**1**	0	**1**	0	**0.683721**	0.343803	0.045156	**0**	1.155560	3.009873	28.69
40 × 10 × 3	**1**	0	**1**	0	**0.612756**	0.267870	0.004384	**0**	0.969632	2.479921	29.27
40 × 10 × 4	**1**	0	**1**	0	**0.651598**	0.298697	0.007699	**0**	1.155926	3.048012	28.33
60 × 6 × 3	**1**	0	**1**	0	**0.608107**	0.196819	0.006148	**0**	1.206281	2.431669	38.47
60 × 6 × 4	**1**	0	**1**	0	**0.626381**	0.207368	0.012994	**0**	1.345371	2.805186	37.81
60 × 8 × 3	**1**	0	**1**	0	**0.679766**	0.254763	0.005548	**0**	1.219353	2.859087	38.35
60 × 8 × 4	**1**	0	**1**	0	**0.591274**	0.110591	0	**0**	1.313420	2.308497	36.35
60 × 10 × 3	**1**	0	**1**	0	**0.673676**	0.213853	0.003948	**0**	1.242544	2.555728	40.68
60 × 10 × 4	**1**	0	**1**	0	**0.577062**	0.084773	0	**0**	1.278479	2.069870	38.31
80 × 6 × 3	**1**	0	**1**	0	**0.650771**	0.147249	0.005391	**0**	1.355336	2.341968	46.54
80 × 6 × 4	**1**	0	**1**	0	**0.644363**	0.126127	0	**0**	1.303573	2.329219	43.54
80 × 8 × 3	**1**	0	**1**	0	**0.698966**	0.142678	0.000997	**0**	1.334800	2.174929	43.14
80 × 8 × 4	**1**	0	**1**	0	**0.682477**	0.176931	0.009163	**0**	1.361026	2.463044	40.10
80 × 10 × 3	**1**	0	**1**	0	**0.689156**	0.172356	0.007819	**0**	1.358525	2.466524	45.20
80 × 10 × 4	**1**	0	**1**	0	**0.737123**	0.212229	0.007963	**0**	1.359221	2.646405	41.55
100 × 6 × 3	**1**	0	**1**	0	**0.677948**	0.138246	0.000759	**0**	1.395922	2.445244	54.84
100 × 6 × 4	**1**	0	**1**	0	**0.700197**	0.134200	0.003562	**0**	1.387003	2.295261	57.11
100 × 8 × 3	**1**	0	**1**	0	**0.696317**	0.115563	0	**0**	1.383655	2.322555	55.50
100 × 8 × 4	**1**	0	**1**	0	**0.718463**	0.126039	0.005127	**0**	1.120998	1.768728	56.30
100 × 10 × 3	**1**	0	**1**	0	**0.715294**	0.148866	0.007116	**0**	1.352186	2.263156	52.85
100 × 10 × 4	**1**	0	**1**	0	**0.749453**	0.100896	0	**0**	1.330505	2.071065	59.56
140 × 12 × 5	**1**	0	**1**	0	**0.755403**	0.190168	0	**0**	1.371924	2.688091	64.69
140 × 12 × 6	**1**	0	**1**	0	**0.764630**	0.259560	0	**0**	1.383023	3.357198	65.18
140 × 14 × 5	**1**	0	**1**	0	**0.776723**	0.228994	0.000127	**0**	1.395913	4.562723	69.95
140 × 14 × 6	**1**	0	**1**	0	**0.784045**	0.234246	0	**0**	1.045416	1.967011	67.28
140 × 16 × 5	**1**	0	**1**	0	**0.758528**	0.126597	0	**0**	1.194072	1.983017	66.91
140 × 16 × 6	**1**	0	**1**	0	**0.787388**	0.709973	0.008020	**0**	0.605981	51.842462	66.11
160 × 12 × 5	0	0	**1**	0	**0.775817**	0.727880	0	0.598952	**0.354630**	20.039824	73.01
160 × 12 × 6	**1**	0	**1**	0	**0.764838**	0.064513	0	**0**	1.394157	1.969237	73.64
160 × 14 × 5	**1**	0	**1**	0	**0.774162**	0.648567	0	**0**	1.188882	31.123178	73.59
160 × 14 × 6	**1**	0	**1**	0	**0.766755**	0.483708	0	**0**	1.123633	8.186515	77.73
160 × 16 × 5	**1**	0	**1**	0	**0.770798**	0.643197	0	**0**	1.222941	29.092472	79.14
160 × 16 × 6	**1**	0	**1**	0	**0.784177**	0.462853	0	**0**	1.347473	7.088264	77.19
180 × 12 × 5	**1**	0	**1**	0	**0.789453**	0.445454	0	**0**	1.294324	7.168520	86.70
180 × 12 × 6	**1**	0	**1**	0	**0.796243**	0.523668	0.000252	**0**	1.324336	10.279342	87.02
180 × 14 × 5	**1**	0	**1**	0	**0.800677**	0.086288	0.004581	**0**	1.401421	1.947298	78.52
180 × 14 × 6	**1**	0	**1**	0	**0.801382**	0.099919	0	**0**	1.399751	2.140318	85.90
180 × 16 × 5	**1**	0	**1**	0	**0.822026**	0.143496	0.014281	**0**	1.400467	2.082115	79.08
180 × 16 × 6	**1**	0	**1**	0	**0.802425**	0.077427	0.001623	**0**	1.375545	1.876406	90.96
200 × 12 × 5	**1**	0	**1**	0	**0.780170**	0.043541	0	**0**	1.408341	1.846278	89.57
200 × 12 × 6	**1**	0	**1**	0	**0.825200**	0.073682	0.000149	**0**	1.362682	1.813930	90.53
200 × 14 × 5	**1**	0	**1**	0	**0.818229**	0.057735	0.000379	**0**	1.370994	1.800598	92.86
200 × 14 × 6	**1**	0	**1**	0	**0.820076**	0.076797	0.002771	**0**	1.402906	1.908537	93.75
	1	0	**1**	0	**0.804360**	0.089244	0.000261	**0**	1.390997	1.998202	82.44

(*continued on next page*)

Table 6 (*continued*)

instance	*C* C(CEA, NNIA)	C(NNIA, CEA)	C(CEA, NSGA-II)	*C* (NSGA-II,CEA)	*HV* CEA	NNIA	NSGA-II	*IGD* CEA	NNIA	NSGA-II	time/s CEA
40 × 6 × 3	**1**	0	**1**	0	**0.501403**	0.174997	0	**0**	1.001701	2.067899	27.82
200 × 16 × 5											
200 × 16 × 6	**1**	0	**1**	0	**0.808347**	0.062154	0	**0**	1.392830	1.909242	89.60
250 × 18 × 8	**1**	0	**1**	0	**0.804667**	0	0.006556	**0**	1.405565	1.495224	106.77
250 × 18 × 10	**1**	0	**1**	0	**0.849015**	0.050794	0.005530	**0**	1.407794	1.658819	98.92
250 × 20 × 8	**1**	0	**1**	0	**0.827688**	0.024462	0	**0**	1.386098	1.637175	104.13
250 × 20 × 10	**1**	0	**1**	0	**0.847866**	0.076287	0.012664	**0**	1.409814	1.720346	109.12
250 × 22 × 8	**1**	0	**1**	0	**0.840819**	0.092569	0.044210	**0**	1.401867	1.732086	106.42
250 × 22 × 10	**1**	0	**1**	0	**0.832179**	0.050170	0.004302	**0**	1.403646	1.731855	106.16
300 × 18 × 8	**1**	0	**1**	0	**0.825289**	0.000577	0.015217	**0**	1.399225	1.195015	120.46
300 × 18 × 10	**1**	0	**1**	0	**0.836783**	0	0.032972	**0**	1.408639	1.263023	128.12
300 × 20 × 8	**1**	0	**1**	0	**0.812751**	0	0.007819	**0**	1.403220	1.449272	143.43
300 × 20 × 10	**1**	0	**1**	0	**0.824948**	0.000220	0.001224	**0**	1.409564	1.466936	132.95
300 × 22 × 8	**1**	0	**1**	0	**0.836400**	0.036623	0.000553	**0**	1.377431	1.678376	132.01
300 × 22 × 10	**1**	0	**1**	0	**0.847342**	0.023446	0	**0**	1.408358	1.672163	131.85
350 × 18 × 8	**1**	0	**1**	0	**0.855785**	0.036068	0.000154	**0**	1.407460	1.737076	145.92
350 × 18 × 10	**1**	0	**1**	0	**0.850001**	0.014584	0	**0**	1.407758	1.619889	145.31
350 × 20 × 8	**1**	0	**1**	0	**0.847873**	0.021625	0	**0**	1.379062	1.642631	159.30
350 × 20 × 10	**1**	0	**1**	0	**0.857111**	0.014920	0	**0**	1.382366	1.590284	145.00
350 × 22 × 8	**1**	0	**1**	0	**0.851241**	0.016185	0	**0**	1.399955	1.569131	153.53
350 × 22 × 10	**1**	0	**1**	0	**0.860440**	0.025913	0.000542	**0**	1.406742	1.659968	152.62
400 × 18 × 8	**1**	0	**1**	0	**0.877507**	0.030258	0.000939	**0**	1.397896	1.653239	163.41
400 × 18 × 10	**1**	0	**1**	0	**0.863917**	0.012212	0	**0**	1.408170	1.600241	170.53
400 × 20 × 8	**1**	0	**1**	0	**0.863451**	0.016100	0	**0**	1.410968	1.635552	172.47
400 × 20 × 10	**1**	0	**1**	0	**0.868198**	0.006900	0	**0**	1.390142	1.528098	173.60
400 × 22 × 8	**1**	0	**1**	0	**0.849131**	0.003208	0	**0**	1.385028	1.474195	171.05
400 × 22 × 10	**1**	0	**1**	0	**0.869410**	0.021022	0.000408	**0**	1.410247	1.620921	179.91

Note: for each instance, significantly better results are marked in bold.

EA1, EA2 and EA3. EA1 denotes the CEA without *LS*, EA2 denotes the CEA without *SST* and EA3 denotes the CEA without *ECT*. 15 instances are selected from the instance set in Section 4.1 as the experiment benchmarks. For fairness, we set the same parameters obtained from Section 4.3.1 for EA1, EA2 and EA3, and the maximum iterations of these four algorithms are all set to 100, which is a moderate number.

The *LS* heuristic is carried out to select a machine for each job, so we randomly determine a machine for each job for EA1 while other operations are same to CEA. For EA2 without *SST*, it is easy to realize by deleting all the *SST* operations in CEA. In CEA, *ECT* exists in two places, i.e., initialization procedure (Section 3.2) and machine-oriented mutation (Section 3.3). Thus, in the algorithm of EA3 without *ECT*, a worker corresponding to each job is randomly chosen in the initialization procedure and the worker matching with a job will move along with the job in the machine-oriented mutation.

Tables 3, 4 and 5 show the performance comparison between CEA and EA1, CEA and EA2, and CEA and EA3, respectively. Even though the computational time of CEA bigger than those of other three algorithms holds for almost all the benchmarks, the gap is not widely and the results of the three metrics clearly illustrate that CEA performs much better than EA1, EA2 and EA3, which demonstrates the effectiveness of *LS*, *SST* and *ECT*. Another meaningful phenomenon can be observed from Tables 3, 4 and 5 is that the values of *HV*(CEA) and *IGD* (EA1\ EA2\ EA3) both show increasing trend as the scale of instance increases, while the values of *HV*(EA1\ EA2\ EA3) and *IGD* (CEA) are almost constant, indicating that the *LS*, *SST* and *ECT* are more effectiveness for larger instances.

4.3.3. Comparison of CEA with NNIA and NSGA-II

To further conform the validity of our proposed CEA, we compare it with NNIA and NSGA-II, which are the state-of-the-art multi-objective algorithms and have been widely used as the comparison algorithms. According to X. Wu and Che (2019), the parameters of NSGA-II are set as follows: the population size is 100, the crossover probability is 1 and the mutation probability is $1/2n$. The parameters of NNIA is same to NSGA-II. To ensure fairness, we record the run time of CEA for any instance and set it as the time limit for NNIA and NSGA-II when they run the same instance.

Table 6 shows the comparison results, which clearly certify the effectiveness of CEA. The three metrics of CEA are much better than NSGA-II for all test instances. From the C-metric, C(CEA, NNIA) = 1 and C(NNIA, CEA) = 0 hold for all instances except for instance $160 \times 12 \times 5$, demonstrating that all the solutions obtained by NNIA are dominated by those derived from CEA for any instance except instance $160 \times 12 \times 5$. But for instance $160 \times 12 \times 5$, the solutions obtained by CEA cannot be dominated by any solution obtained by NNIA. The reason for this phenomenon is probably the randomness of NNIA. Furthermore, the results of *HV* and *IGD* in Table 6 indicate that the quality of optimal Pareto fronts obtained by CEA is far better than those derived from NNIA and NSGA-II. In terms of *HV* and *IGD* for NNIA and NSGA-II, the results in Table 6 also indicate that NNIA outperforms NSGA-II for most instances.

5. Conclusion and future work

To conclude, we presented a new model of NUPMSP: unrelated parallel machine scheduling with sequence and machine-dependent setup times, limited worker resources, and learning effect. This model is the first to consider worker resources and learning effect in the context of unrelated parallel machine scheduling with sequence and machine-dependent setup times. It is also the first time to consider the processing time and *TEC* simultaneously, which helps operation managers to make balance between production efficiency and environmental protection. We propose an effective CEA to solve the NUPMSP, in which three methods including *LS* heuristic, *SST* rule and *ECT* rule are presented and combined to obtain a high-quality population. In our experiments, we first used the Taguchi method to obtain the optimum combination of the population size, machine-oriented mutation probability and worker-oriented mutation probability. Then, three experiments are carried out to prove the effectiveness of the proposed *LS* heuristic, *SST* rule and *ECT* rule based on 15 instances. Finally, extensively comparisons were conducted between our proposed CEA and other two well-known algorithms based on 72 instances. The experimental results demonstrated that the CEA can outperforms the other algorithms in almost all instances.

Our work is significant to bridge the theoretical gaps and guide practical production. Theoretically, we have pioneered a new research direction for the UPMSP-SMDST and have designed an effective combinatorial evolutionary algorithm to the proposed problem. Practically, our NUPMSP model can help manufacturing factories to schedule jobs and worker resources in a more reasonable manner, especially for the highly-manual mixed-model manufacturing factories, such as printing industries, in which workers are needed to reset each machine for the next job.

For future work, we will take the machine maintenance activities and human forgetting effect into account in our model. In addition, more efficient algorithms will be developed to solve the problem.

CRediT authorship contribution statement

Like Zhang: Conceptualization, Methodology, Formal analysis, Writing - original draft. **Qianwang Deng:** Supervision, Funding acquisition, Resources. **Ruihang Lin:** Software, Validation. **Guiliang Gong:** Project administration. **Wenwu Han:** Project administration.

Declaration of Competing Interest

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

Acknowledgements

This work was supported by the National Key R&D Program of China (grant nos. 2020YFB1712100, 2018YFB1701400); the National Natural Science Foundation of China (grant no. 61973108); the State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University (grant no. 71775004), the State Key Laboratory of Construction Machinery (grant no. SKLCM2019-03); and the National Nature Science Foundation of China (grant no. 72001217); the National Nature Science Foundation of Changsha (grant no. kq2007033).

References

Abreu Gomes, F. R., & Mateus, G. R. (2017). Improved combinatorial benders decomposition for a scheduling problem with unrelated parallel machines. *Journal of Applied Mathematics, 2017*, 1–10.

Anzanello, M. J., & Fogliatto, F. S. (2010). Scheduling learning dependent jobs in customised assembly lines. *International Journal of Production Research, 48*, 6683–6699.

Arnaout, J.-P., Rabadi, G., & Musa, R. (2010). A two-stage Ant Colony Optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing, 21*, 693–701.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *International Journal of Advanced Manufacturing Technology, 76*, 1705–1718.

Azzouz, A., Ennigrou, M., & Ben Said, L. (2018). Scheduling problems under learning effects: classification and cartography. *International Journal of Production Research, 56*, 1642–1661.

Bautista, J., Alfaro-Pozo, R., & Batalla-Garcia, C. (2015). Consideration of human resources in the mixed-model sequencing problem with work overload Minimization: Legal provisions and productivity improvement. *Expert Systems with Applications, 42*, 8896–8910.

Bektur, G., & Sarac, T. (2019). A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Computers & Operations Research, 103*, 46–63.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research, 115*, 173–178.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research, 188*, 315–329.

Bitar, A., Dauzere-Peres, S., Yugma, C., & Roussel, R. (2016). A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. *Journal of Scheduling, 19*, 367–376.

Błażewicz, P. D. J., Ecker, K. H., Pesch, E., Schmidt, P. D. G., & Węglarz, P. D. J. (1996). *Scheduling Computer and Manufacturing Processes*. Berlin: Springer, Berlin Heidelberg.

Bozorgirad, M. A., & Logendran, R. (2012). Sequence-dependent group scheduling problem on unrelated-parallel machines. *Expert Systems with Applications, 39*, 9021–9030.

Costa, A., Cappadonna, F. A., & Fichera, S. (2013). A hybrid genetic algorithm for job sequencing and worker allocation in parallel unrelated machines with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology, 69*, 2799–2817.

Davis, E., & Jaffe, J. M. (1981). Algorithms for scheduling tasks on unrelated processors. *Journal of the Association for Computing Machinery, 28*, 721–736.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*, 182–197.

Exposito-Izquierdo, C., Angel-Bello, F., Melian-Batista, B., Alvarez, A., & Baez, S. (2019). A metaheuristic algorithm and simulation to study the effect of learning or tiredness on sequence-dependent setup times in a parallel machine scheduling problem. *Expert Systems with Applications, 117*, 62–74.

Ezugwu, A. E. (2019). Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. *Knowledge-Based Systems, 172*, 15–32.

Ezugwu, A. E., & Akutsah, F. (2018). An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. *IEEE Access, 6*, 54459–54478.

Fleszar, K., Charalambous, C., & Hindi, K. S. (2012). A variable neighborhood descent heuristic for the problem of makespan minimisation on unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing, 23*, 1949–1958.

Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y., & Pan, Q. (2019). A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE-CAA Journal of Automatica Sinica, 6*, 904–916.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering, 121*, 139–149.

Gong, G., Chiong, R., Deng, Q., & Gong, X. (2019). A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility. *International Journal of Production Research*, 1–15.

Gong, G., Chiong, R., Deng, Q., Han, W., Zhang, L., Lin, W., & Li, K. (2020). Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert Systems with Applications, 141*.

Gong, M., Jiao, L., Du, H., & Bo, L. (2008). Multiobjective immune algorithm with nondominated neighbor-based selection. *Evolutionary Computation, 16*, 225–255.

Hamta, N., Ghomi, S. M. T. F., Jolai, F., & Shirazi, M. A. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics, 141*, 99–111.

Huang, S., Cai, L., & Zhang, X. (2010). Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering, 58*, 165–174.

Lei, D., & Guo, X. (2014). Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. *International Journal of Production Research, 52*, 2519–2529.

Lin, S.-W., & Ying, K.-C. (2014). ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. *Computers & Operations Research, 51*, 172–181.

Lin, Y.-K., & Hsieh, F.-Y. (2014). Unrelated parallel machine scheduling with setup times and ready times. *International Journal of Production Research, 52*, 1200–1214.

Marinho Diana, R. O., de Franca Filho, M. F., de Souza, S. R., & de Almeida Vitor, J. F. (2015). An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing, 163*, 94–105.

Ostermeier, F. F. (2019). The impact of human consideration, schedule types and product mix on scheduling objectives for unpaced mixed-model assembly lines. *International Journal of Production Research*, 1–20.

Othman, M., Gouw, G. J., & Bhuiyan, N. (2012). Workforce scheduling: A new model incorporating human factors. *Journal of Industrial Engineering and Management, 5*, 259–284.

Perez-Gonzalez, P., Fernandez-Viagas, V., Zamora Garcia, M., & Framinan, J. M. (2019). Constructive heuristics for the unrelated parallel machines scheduling problem with machine eligibility and setup times. *Computers & Industrial Engineering, 131*, 131–145.

Pindeo, M. J. P. H., USA. (2002). Scheduling: Theory, Algorithms, and Systems. USA: Prentice Hall.

Rabadi, G., Moraga, R. J., & Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing, 17*, 85–97.

Rauchecker, G., & Schryen, G. (2019). Using high performance computing for unrelated parallel machine scheduling with sequence-dependent setup times: Development and computational evaluation of a parallel branch-and-price algorithm. *Computers & Operations Research, 104*, 338–357.

Sheikhalishahi, M., Eskandari, N., Mashayekhi, A., & Azadeh, A. (2019). Multi-objective open shop scheduling by considering human error and preventive maintenance. *Applied Mathematical Modelling, 67*, 573–587.

Soares, L. C. R., & Carvalho, M. A. M. (2020). Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research, 285*, 955–964.

Torabi, S. A., Sahebjamnia, N., Mansouri, S. A., & Bajestani, M. A. (2013). A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing, 13*, 4750–4762.

Touat, M., Bouzidi-Hassini, S., Benbouzid-Sitayeb, F., & Benhamou, B. (2017). A hybridization of genetic algorithms and fuzzy logic for the single-machine scheduling with flexible maintenance problem under human resource constraints. *Applied Soft Computing, 59*, 556–573.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research, 211*, 612–622.

Wang, J.-B., Wang, D., Wang, L.-Y., Lin, L., Yin, N., & Wang, W.-W. (2009). Single machine scheduling with exponential time-dependent learning effect and past-sequence-dependent setup times. *Computers & Mathematics with Applications, 57*, 9–16.

Wang, M., & Pan, G. (2019). A novel imperialist competitive algorithm with multi-elite individuals guidance for multi-object unrelated parallel machine scheduling problem. *IEEE Access, 7*, 121223–121235.

Wu, C., Lee, W., & Liou, M. (2013). Single-machine scheduling with two competing agents and learning consideration. *Information Sciences, 251*, 136–149.

Wu, R., Li, Y., Guo, S., & Xu, W. (2018). Solving the dual-resource constrained flexible job shop scheduling problem with learning effect by a hybrid genetic algorithm. *Advances in Mechanical Engineering, 10*, 1–14.

Wu, X., & Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega-International Journal of Management Science, 82*, 155–165.

Xue, Y., Rui, Z., Yu, X., Sang, X., & Liu, W. (2019). Estimation of distribution evolution memetic algorithm for the unrelated parallel-machine green scheduling problem. *Memetic Computing, 11*, 423–437.

Yepes-Borrero, J. C., Villa, F., Perea, F., & Pablo Caballero-Villalobos, J. (2020). GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Systems with Applications, 141*.

Ying, K.-C., Lee, Z.-J., & Lin, S.-W. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing, 23*, 1795–1803.

Zhang, L., Deng, Q., Gong, G., & Han, W. (2019). A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption. *International Journal of Production Research*, 1–20.