International Journal of Production Research

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tprs20

A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption

Like Zhang, Qianwang Deng, Guiliang Gong & Wenwu Han

To cite this article: Like Zhang, Qianwang Deng, Guiliang Gong & Wenwu Han (2020) A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption, International Journal of Production Research, 58:22, 6826-6845, DOI: 10.1080/00207543.2019.1685708

To link to this article: https://doi.org/10.1080/00207543.2019.1685708

International Journal of Production Research, 2020
Vol. 58, No. 22, 6826–6845, https://doi.org/10.1080/00207543.2019.1685708

A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption

Like Zhang, Qianwang Deng*, Guiliang Gong and Wenwu Han

State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha, People's Republic of China

(*Received 20 March 2019; accepted 21 October 2019*)

The previous studies on scheduling problem with tool changes take processing time as the only reason for the tool wear, which is not accurate in the real manufacturing system. This paper takes processing speed and processing time into consideration simultaneously and proposes a new unrelated parallel machine scheduling problem (UPMSP) with tool changes caused by the tool wear, in which the energy consumption rate of the parallel machines is influenced by two factors: tool changes and corresponding processing speed. A new effective heuristic evolutionary algorithm (NHEA) is presented to solve the proposed UPMSP with objectives of optimising total energy consumption and makespan. For the NHEA, some effective operators such as target-searching operators are designed to accelerate the search efficiency and further exploit the solution space. A first fit decreasing algorithm is presented and incorporated into the NHEA to reduce the number of tool changes. The Taguchi method of Design of Experiments is used to obtain the best combination of key parameters of the NHEA. Extensive computational experiments are carried out to compare the NHEA with some well-known algorithms. The results validate that the proposed NHEA is able to obtain better Pareto solutions for UPMSP with tool changes.

Keywords: unrelated parallel machine; tool changes; tool wear; evolutionary algorithm; first fit decreasing algorithm

1. Introduction

In the process of machining, severe sliding friction will occur between cutting tool and workpiece (Yao and Hong 2019). Thus, tool wear is inevitable in the cutting process, which has been considered as one of the main reasons leading to the degradation of product quality (Hao et al. 2017) and pushing cutting tool to be changed. Gray, Seidmann, and Stecke (1993) pointed out that above 90% tool changes are caused by tool wear. Several papers indicated that frequent tool changes have some drawbacks, such as decreasing productivity (Xu and Cao 2014), consuming additional energy (Hu et al. 2018) and delaying completion time (Akturk and Avci 1996).

With regard to these drawbacks, some studies on scheduling problem with tool changes have been carried out. For example, Akturk, Ghosh, and Gunes (2004) and Chen (2008) investigated the single machine scheduling problem with tool changes, where the tool life was fixed and jobs were separated by tool changes into batches. Considering the influence of tool wear on job quality, Xu and Shi (2013) divided jobs into two sets, special jobs and normal jobs, where special jobs should be processed at the first prefixed time units of a tool life. However, as to the unrelated parallel machine scheduling problem with tool changes, which widely exits in the real-world production systems (Wu and Che 2019), no literature has been found.

Most of researchers on scheduling problem with tool changes assumed that the processing time was the only factor for tool wear (Akturk, Ghosh, and Gunes 2003, 2004; Chen 2008; Costa, Cappadonna, and Fichera 2016; Xu and Shi 2013; Xu and Xu 2018; Yazdani, Khalili, and Jolai 2016). In their papers, tool wear is linear with processing time and a cutting tool needs to be changed before the tool wear accumulates to a given value. However, Astakhov and Davim (2008) and Ji et al. (2018) pointed out that for a machine, under the same condition, the faster the speed is, the greater the tool wear rate is. Hence, it is more appropriate to consider both processing time and processing speed simultaneously to optimise the scheduling problem with tool changes.

Up to date, all the studies ignore the energy consumption caused by tool changes (Aghelinejad, Ouazene, and Yalaoui 2018; Arik and Toksari 2018; Li et al. 2016; Rager, Gahm, and Denz 2015; Thevenin, Zufferey, and Potvin 2017). However, as Hu et al. (2018) pointed out that the non-processing time of tool changes consumed amounts of energy and Oda et al.

*Corresponding author. Email: deng_arbeit@hnu.edu.cn

(2012) also indicated that the energy consumption of most machines will increase as tool wear deteriorates. International Energy Agency (EIA 2016) predicted that by 2040, the total energy consumption (*TEC*) of the world would increase 48% over 2012. Thus, it is of urgent and important to take the energy consumption caused by tool changes into consideration and explore energy-efficient scheduling methods to decrease energy consumption.

Inspired by these problems analysed above, we will study the unrelated parallel machine scheduling problem with tool changes (UPMSPWTC) in more depth with the following distinguishes: (1) a new mathematical model of the UPMSPWTC is proposed, in which the processing time, the processing speed and the energy consumption caused by tool changes are simultaneously taken into consideration; (2) a new effective heuristic evolutionary algorithm (NHEA) is designed to solve the UPMSPWTC; (3) three types of experiments based on 30 constructed instances are conducted to validate the performance of NHEA.

The rest of this paper is organised as follows: scheduling problems which relate to the UPMSP with energy consumption and tool changes are reviewed respectively in Section 2. In Section 3, the proposed problem of UPMSPWTC is illustrated specifically and a mathematical programming model is presented. In Section 4, a multi-objective algorithm for the UPMSPWTC is well-designed. Compared with other algorithms, the performance of the proposed algorithm is discussed based on the numerical computational results in Section 5. Finally, the study ends with conclusions in Section 6.

2. Literature review

2.1. Unrelated parallel machine scheduling problem with energy consumption

A reasonable scheduling plan can effectively reduce the expenditure of energy (Wang, Jiang et al. 2018), which has successfully attracted more and more scholars' attention. In the past few years, different energy-efficient methods have been put up (Che et al. 2017; Che, Zeng, and Lyu 2016; Cheng et al. 2017; Ding, Song, and Wu 2016; Gahm et al. 2016; Li et al. 2011; Lin et al. 2015; Mansouri, Aktas, and Besikci 2016; Shrouf et al. 2014; Wu and Che 2019; Zhang and Chiong 2016), such as speed-scaling method (Zhang and Chiong 2016) and off-peak method (Shrouf et al. 2014), and quite a few papers about UPMSP with energy consumption have been studied. We denote the UPMSP with energy consumption as UPMSPWEC throughout the paper. By analysing the articles in this stream in recent years, their innovations are mainly focused on key features, decision variables and solution algorithms. As Liu, Yang, and Cheng (2017) pointed out, the key features mainly refer to resource constraints and decision variables mainly include processing speed, start time, wait time and job sequence.

Key features and solution algorithms: Ding et al. (2016) studied the UPMSPWEC with the key features of a time-of-use (TOU) pricing scheme and a deadline of makespan. To solve that problem, they put up a new mixed linear programming model and a column generation heuristic. Che, Zhang, and Wu (2017) discussed a similar problem and built a modified continuous-time mixed-integer linear programming model for their problem. They also exploited a two-stage heuristic for their model with large-size jobs. Abikarram, McConky, and Proano (2019) researched the UPMSPWEC with a key feature of pricing policies. Considering both demand charges and consumption charges, they presented a mathematical optimisation model for their problem and validated that the total electricity costs can be effectively reduced by considering demand charges when making dispatching decision.

Decision variables and solution algorithms: considering the decision variables of machine's three states, i.e. operation, wait and stop, Li et al. (2016) studied the UPMSPWEC to minimise both the total tardiness and energy consumption. Meanwhile ten heuristic algorithms were developed and tested in their paper. Wu and Che (2019) considered the decision variable of machine running speed into the UPMSPWEC and designed a memetic differential evolution algorithm to tackle their problem. In that model, a machine speed was a discrete value choosing from a set. Liang et al. (2015) addressed an UPMSPWEC model with different arrival times, due date of jobs and different machine state operations. They aimed at minimising *TEC* and total tardiness by introducing the power-down method. And they proposed an ant optimisation algorithm based on apparent tardiness cost (ATC) heuristic rule for their model. Considering the energy-saving strategy of turning off and on, Meng et al. (2019) optimised the *TEC* of hybrid flow shop scheduling problem with unrelated parallel machines.

Key features, decision variables and solution algorithms: Contrary to discrete value of machine processing speed, Jin et al. (2015) considered a decision variable of speed with arbitrary continuous value, and they designed a polynomial-time algorithm for the UPMSPWEC with restricted parallel processors. Zheng and Wang (2018) integrated a key feature of reproducible resource constraint and a decision variable of processing speed into the UPMSP to minimise the *TEC*. They proposed a collaborative multi-objective fruit fly optimisation algorithm for their model and designed an energy saving technique at the end of their algorithm which effectively improved the solutions.

From the above literature review, we can see that incorporating the tool changes caused by tool wear into the UPMSP has not been discussed, although the tool changes are inevitable. As mentioned in the previous section above, the tool changes have negative effects on the optimisation targets and should be paid attention to. To fill this gap, in this paper, we originally

develop a novel mathematical model of UPMSP with tool changes, which reflects the influence of tool changes caused by tool wear on the bi-objective: *TEC* and makespan.

2.2. Related studies on tool changes

As an execution component of machine tools, the tool changes are unavoidable. However, few researches of job scheduling problems with tool changes were published. Gray, Seidmann, and Stecke (1993) pointed out that there was a strong relevance between tool management and the performance of manufacturing system. And about 25–30% of the production costs was caused by tooling cost (Kouvelis 1991). Thus, taking the tool changes into the scheduling problems is very necessary. Based on reality, Akturk, Ghosh, and Gunes (2003) studied a single CNC scheduling problem with constraint of tool changes caused by tool wear. According to simple dispatch rules and generic search, they investigated seven heuristic algorithms' performance on the objective of total completion time. In that paper, they assumed that the tool wear was proportional to the processing time. Afterwards, they focused on the performance of the shortest processing time first (SPT) rule and gave its worst-case bounds on the same scheduling problem in 2004 (Akturk, Ghosh, and Gunes 2004). A new model considering tool changes and machining conditions control into a single CNC job scheduling problem was proposed by Akturk, Ghosh, and Kayan (2007). Various factors affecting the tool wear were taken into account in their model. Chen (2008) discussed the tool changes constraint on a single machine to minimise the total tardiness of jobs. In their problem, the tool changes were periodic and they designed two mix binary integer programming (BIP) models to solve their problem. Xu and Shi (2013) addressed a new scheduling problem with tool changes on a single machine to optimise the makespan. In their paper, jobs were divided into two parts, i.e. normal jobs and special jobs which should be processed in a fixed initial time period of a tool life. Baykasoglu and Ozsoydan (2018) proposed a multiple starting simulated annealing algorithm to optimise the non-processing time of tool changes.

Depending on the relevant literature mentioned above, most of the studies paid their attention to the optimisation of a single machine scheduling or a single objective. What's more, there are almost no researches taking other factors except for processing time on tool wear into account and none of the studies were related to the UPMSP. To bridge the gap, this paper originally aims to explore the deep mechanism of tool changes caused by the processing time and the processing speed in UPMSP.

3. Mathematical programming model of the proposed UPMSPWTC

3.1. Problem description

The model of UPMSPWTC can be described as follows.

(1) n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ which have the same material are waiting to be processed on m unrelated machines $M = \{M_1, M_2, \ldots, M_m\}$. Each machine executes the process with the same tools and all the machines share the same set of $r + 1$ optional speeds $v = \{v_0, v_1, v_2, \ldots, v_r\}$, where v_0 denotes that the machine is in the stand-by mode.

(2) For the factor of the job processing time, each job J_i processed on machine M_j has a basic processing time p_{ij}. If the speed of machine M_j is set to v_k ($k = 1, 2, \ldots, r$) for job J_i, the actual processing time of job J_i is p_{ij}/v_k denoted as p_{ijk} and the unit power of machine M_j is PP_{jk}. Obviously, if $v_k > v_{k'}$, then $p_{ijk} \times PP_{jk} > p_{ijk'} \times PP_{jk'}$. This indicates that a lower processing speed will lead to more processing time but less energy consumption.

(3) For the factor of cutting tool with upper limit T of tool wear, the per unit time of tool wear corresponding to speed v_k is w_k. From the literature (Astakhov and Davim 2008), we observe that the unit tool wear is a non-decreasing function of the speed, i.e. $w_k \geq w_{k'}$ if $v_k \geq v_{k'}$. When the tool wear of a tool reaches T, the corresponding machine has to stop to have a tool change. At this time, the machine holds a stand-by mode and the duration time of the tool change is T_C.

(4) Under a stand-by mode, a machine M_j consumes SP_j energy per unit time.

The problem proposed needs to solve the three subproblems simultaneously under tool changes constraint: (a) assign a machine for each job; (b) determine a processing speed for each job; (c) determine the job sequence on each machine.

In this paper, our objectives are:

- Minimising the maximum completion time of all jobs (C_{max}).
- Minimising the *TEC*.

Some assumptions are put forward:

- Each machine only can process one job with one speed at a time.

- Each job is processed only once and not resumable.
- All the machines have finished a tool change and all jobs are available at time zero.
- Turn-off for a machine is not allowed until all the jobs allocated to the machine are finished.
- $\text{Max}\{p_{ijk} \times w_k\} \leq T$, where $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$, $k \in \{1, 2, \ldots, r\}$. Otherwise there is trivially no feasible solution.

THEOREM 1 *An optimal schedule exists where each of machine has no idle time before it finishes all the jobs assigned to it. That means the machine is running all the time except for executing the tool changes.*

Proof If there exists idle time except for tool changes for a machine, the subsequent jobs or subsequent tool changes can be moved forward, resulting the energy consumption of the stand-by mode and makespan of the machine decrease. ■

3.2. Mixed integer linear programming model

The notations used for the proposed model are specified as follows:

- n: total number of jobs;
- m: total number of machines;
- r: total number of speed levels except for v_0;
- i, h: index of jobs, $i, h = 1, 2, \ldots, n$;
- j: index of machines, $j = 1, 2, \ldots, m$;
- k: index of speed levels, $k = 0, 1, \ldots, r$;
- l: position index of a schedule on each machine, $l = 0, 1, \ldots, n$, where 0 denotes a dummy position at time zero;
- T: upper limit of tool wear;
- T_C: duration time of a tool change;
- p_{ijk}: actual processing time of J_i on M_j at speed v_k;
- w_k: the unit tool wear of a tool corresponding to speed v_k;
- PP_{jk}: the unit power of M_j when its running speed is v_k, $k = 1, 2, \ldots, r$;
- SP_j: the unit power of M_j under the stand-by mode;
- C_j: the maximum completion time of jobs on M_j;
- x_{ijlk}: a binary decision variable. $x_{ijlk} = 1$ if J_i is assigned to position l on M_j and its processing speed is v_k; otherwise, $x_{ijlk} = 0$;
- y_{jl}: a binary decision variable. $y_{jl} = 1$ if tool change is required immediately once the job at position l on M_j has been completed; otherwise, $y_{jl} = 0$. $y_{j0} = 1$ denotes that a tool change for M_j has been finished at time 0;
- B: a large positive number;
- E_{jl}: the cumulative value of tool wear between the completion time of the previous tool change and the completion time of the job at position l on M_j, and $E_{j0} = 0$ denotes the cumulative value of tool wear of M_j at time 0;
- j_q: the qth tool used on M_j, $q = 1, 2, \ldots, e_j$ for each machine j, where $e_j = \sum_{l=0}^{n} y_{jl}$;

The objective functions:

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

$$\min \quad TEC = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} PP_{jk} + \sum_{j=1}^{m}\sum_{l=1}^{n} y_{jl} T_c SP_j \tag{2}$$

Subject to:

$$C_{\max} \geq C_j, \quad \forall j = 1, 2, \ldots, m. \tag{3}$$

$$C_j = \sum_{i=1}^{n}\sum_{l=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} + \sum_{l=1}^{n} y_{jl} T_c, \quad \forall j = 1, 2, \ldots, m. \tag{4}$$

$$\sum_{j=1}^{m}\sum_{l=1}^{n}\sum_{k=1}^{r} x_{ijlk} = 1, \quad \forall i = 1, 2, \ldots, n. \tag{5}$$

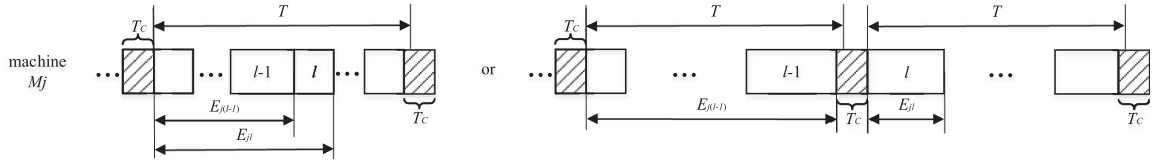

Figure 1. Illustration of E_{jl}.

$$\sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} \le 1, \quad \forall j = 1,2,\ldots,m, l = 1,2,\ldots,n. \tag{6}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} \ge \sum_{i=1}^{n}\sum_{k=1}^{r} x_{ij(l+1)k}, \quad \forall j = 1,2,\ldots,m, l = 1,2,\ldots,n-1. \tag{7}$$

$$E_{j(l-1)} + \sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} w_k \le E_{jl} + y_{j(l-1)} B, \quad \forall j = 1,2,\ldots,m, l = 1,2,\ldots,n. \tag{8}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} w_k \le E_{jl}, \quad \forall j = 1,2,\ldots,m, l = 1,2,\ldots,n. \tag{9}$$

$$E_{jl} \le T, \quad \forall j = 1,2,\ldots,m, l = 1,2,\ldots,n. \tag{10}$$

Equations (1)–(2) represent the objectives of minimising makespan and *TEC* respectively. Constraint (3) determines the makespan. Constraint (4) is the calculation of C_j. Constraint (5) guarantees that each job is only processed by one machine with one position and one speed. Constraint (6) ensures that no more than one job can be processed at any position on a machine. Constraint (7) guarantees that there is no vacant position before a filled position of the same machine. Constraints (8)-(9) guarantee that the values of E_{jl} are set properly, i.e. either $E_{j(l-1)} + \sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} w_k \le E_{jl}$ for $y_{j(l\text{-}1)} = 0$ or $\sum_{i=1}^{n}\sum_{k=1}^{r} x_{ijlk} p_{ijk} w_k \le E_{jl}$ for $y_{j(l\text{-}1)} = 1$, which is illustrated in Figure 1. Constraint (10) ensures that the cumulative value of tool wear of any tool cannot exceed its upper limit T.

4. New heuristic evolutionary algorithm

In this section, a new efficiency heuristic evolutionary algorithm is proposed and it consists of four main parts: (a) a list scheduling heuristic depending on time processing efficiency is used to generate a good initial population; (b) two targeted searching operates (i.e. a novel job-reassign heuristic and a novel speed-adjusting heuristic) which can quickly balance machine load and reduce energy consumption are designed; (c) update and perform hypermutation on elite population (*EP*) in each generation; (d) execute the first fit decreasing (*FFD*) algorithm on *EP* obtained by the proposed algorithm at the end. The frame of the NHEA is shown in Figure 2.

4.1. Individual representation

As described in Section 3, a dispatcher needs to select a machine and a processing speed for each job, then decides the processing sequence of all jobs on each machine under tool changes constraint caused by tool wear. Thereby, a solution for each machine can be consisted of five parts: a job processing sequence vector (*JS*), a corresponding speed sequence vector (*SS*), a corresponding tool wear vector (*TwJ*) caused by processing *JS*, a final tool wear vector of each tool (*TwT*) before it is changed and a tool use vector (*TJ*) for *JS*, where the size of *TwT* is the number of tools used by a machine. For an individual a in the tth generation, the *JS*, *SS*, *TwJ*, *TwT* and *TJ* on M_j are denoted by Equations (11)–(15), respectively, where TwT_{j_q} is the final tool wear of the qth tool before it is changed on M_j. Equation (16) is the calculation of TwJ_i and k is the corresponding speed level of processing J_i. $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$ and $TJ^{at}_{M_j}$ are derived from $J^{at}_{M_j}$ and $V^{at}_{M_j}$.

$$JS = J^{at}_{M_j} = [J_i], \quad J_i \in \{J_1,\ldots,J_n\} \tag{11}$$

$$SS = V^{at}_{M_j} = [v_k], \quad v_k \in \{v_1,\ldots,v_r\} \tag{12}$$

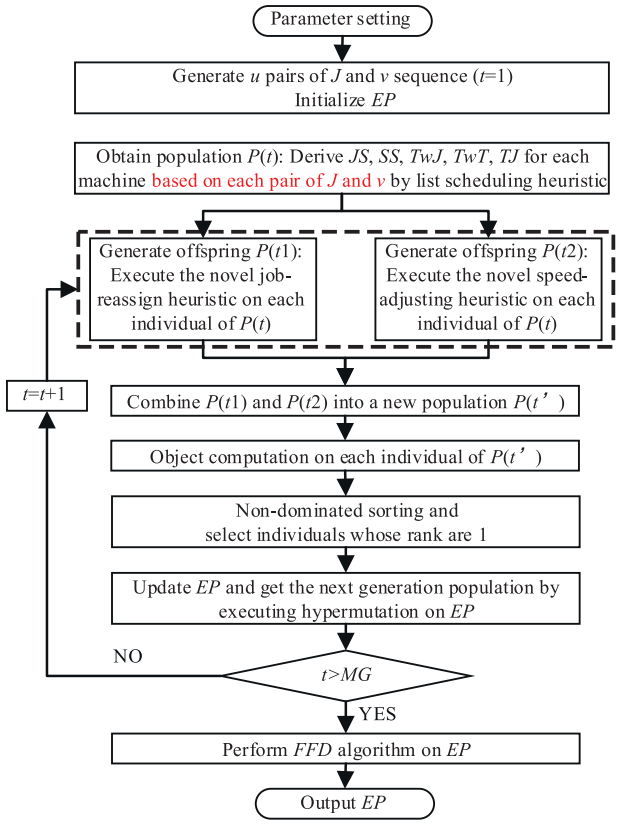

Figure 2. The frame of NHEA.

Table 1. Date of individual representation example.

Input						Output			
M	M_1			M_2		M_1		M_2	
J	J_3	J_1	J_2	J_4		$J^{at}_{M_1}$	$[J_3, J_1]$	$J^{at}_{M_2}$	$[J_2, J_4]$
p_{ij}	10	24	48	12		$V^{at}_{M_1}$	[1.25, 1.5]	$V^{at}_{M_2}$	[1.75, 1]
k	2	3	4	1		$TwJ^{at}_{M_1}$	[0.16, 0.48]	$TwJ^{at}_{M_2}$	[0.96, 0.18]
w_k	{0, 0.015, 0.02, 0.03, 0.035}					$TwT^{at}_{M_1}$	[0.64]	$TwT^{at}_{M_2}$	[0.96, 0.18]
v	{0, 1, 1.25, 1.5, 1.75}					$TJ^{at}_{M_1}$	$[1_1, 1_1]$	$TJ^{at}_{M_2}$	$[2_1, 2_2]$

$$TwJ = TwJ^{at}_{M_j} = [TwJ_i], \quad i \in \{1, 2, \ldots, |J^{at}_{M_j}|\} \tag{13}$$

$$TwT = TwT^{at}_{M_j} = [TwT_{j_q}], \quad q \in \{1, 2, \ldots, e_j\} \tag{14}$$

$$TJ = TJ^{at}_{M_j} = [j_q], \quad q \in \{1, 2, \ldots, e_j\} \tag{15}$$

$$TwJ_i = p_{ij}/v_k w_k, \quad k \in \{1, 2, \ldots, r\} \tag{16}$$

Table 1 shows an example of individual a in tth generation, where the $J^{at}_{M_j}$ and $V^{at}_{M_j}$ have been obtained in advance. There are two machines, four jobs and five speed levels, and the upper limit of tool wear is 1. From the information on the left, we get the complete representation of individual a on the right.

4.2. Initialisation

Due to the enormous search space, a good initial population can improve the search efficiency. In this paper, we use a list scheduling heuristic designed by Davis and Jaffe (1981) to generate *JS*, *SS*, *TwJ*, *TwT* and *TJ* for each machine of each

individual. The list scheduling heuristic depends on some principles to select a processor for each job. Considering that one of our objectives is to optimise the makespan, we definite a principle of time processing efficiency as follows.

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

The initialisation procedure based on list scheduling heuristic depending on principle 1 is illustrated in Algorithm 1.

Algorithm 1: Initialisation

1. Set $J := [J_i]$, $i \in \{1, 2, \ldots, n\}$ and $|J| = n$; % a set of unassigned jobs, in which the sequence is randomly generated.
2. Set $v := [v_k]$, $k \in \{1, 2, \ldots, r\}$ and $|v| = n$; % a set of processing speed corresponding to J, in which the sequence is also randomly generated.
3. Set $C_j := 0$, $q := 1$ and $TwT_{j_q} = 0$ for $1 \le j \le m$;
4. Set $DM := \varnothing$; % a set of deactivated machines.
5. Set $J^{a1}_{M_j} = \emptyset$, $V^{a1}_{M_j} = \emptyset$, $TwJ^{a1}_{M_j} = \emptyset$, $TwT^{a1}_{M_j} = \emptyset$ and $TJ^{a1}_{M_j} = \emptyset$ for $1 \le j \le m$;
6. Set $p^{min}_{ij} := \min_{1 \le j \le m} p_{ij}$ and calculate $Te_{ij} = p^{min}_{ij} / p_{ij}$ for $1 \le i \le n$, $1 \le j \le m$;
7. Create l_j by listing all jobs on M_j in non-increasing order of their values of Te_{ij} for $1 \le j \le m$;
8. **while** $J \neq \varnothing$ **do**
9. Find M_j whose $C_j := \min_{1 \le j \le m} C_j, j \notin DM$; find the next unassigned job J_i on l_j and its corresponding processing speed $v_k \in v$;
10. **if** $Te_{ij} > 1/\sqrt{m}$ **then**
11. $C_j := C_j + p_{ij}/v_k$; $l_j := l_j \backslash \{J_i\}$ for $1 \le j \le m$; $J^{a1}_{M_j} := [J^{a1}_{M_j}, J_i]$; $V^{a1}_{M_j} := [V^{a1}_{M_j}, v_k]$; $TwJ^{a1}_i = p_{ij}/v_k * w_k$;
12. $TwJ^{a1}_{M_j} := [TwJ^{a1}_{M_j}, TwJ^{a1}_i]$; $TwT_{j_q} := TwT_{j_q} + TwJ^{a1}_i$; $J := J \backslash \{J_i\}$; $v := v \backslash \{v_k\}$;
13. **if** $TwT_{j_q} > T$ **then**
14. $C_j := C_j + T_c$; $TwT_{j_q} := TwT_{j_q} - TwJ^{a1}_i$; $TwT^{a1}_{M_j} := [TwT^{a1}_{M_j}, TwT_{j_q} - TwJ^{a1}_i]$; $q = q + 1$; $TwT_{j_q} := TwJ^{a1}_i$; $TJ^{a1}_{M_j} := [TJ^{a1}_{M_j}, j_q]$;
15. **else**
16. $TJ^{a1}_{M_j} := [TJ^{a1}_{M_j}, j_q]$;
17. **endif**
18. **else**
19. $DM := DM \cup M_j$;
20. **endif**
21. **endwhile**
22. **for** $j := 1$ to m **do**
23. $TwT^{a1}_{M_j} := [TwT^{a1}_{M_j}, TwT_{j_q}]$;
24. **endfor**
25. Output $J^{a1}_{M_j}$, $V^{a1}_{M_j}$, $TwJ^{a1}_{M_j}$, $TwT^{a1}_{M_j}$, $TJ^{a1}_{M_j}$, C_j for $1 \le j \le m$;

To better understand the Algorithm 1, we consider an example with four jobs, two machines and three speed levels. The relevant data are listed in Table 2. Suppose that an individual a is consist of $J = [J_2, J_1, J_3, J_4]$ and $v = [v_2, v_1, v_2, v_2]$. First, according to lines 3–5, set $C_j = 0$, $q = 1$, $TwT_{j_q} = 0$, $DM = \varnothing$, $J^{a1}_{M_j} = \emptyset$, $V^{a1}_{M_j} = \emptyset$, $TwJ^{a1}_{M_j} = \emptyset$, $TwT^{a1}_{M_j} = \emptyset$ and $TJ^{a1}_{M_j} = \emptyset$ for each machine. Performing line 6, we get $Te_{21} = 1$, $Te_{11} = 0.8$, $Te_{31} = 1$, $Te_{41} = 0.75$, $Te_{22} = 0.7$, $Te_{12} = 1$, $Te_{32} = 0.67$ and $Te_{42} = 1$. Then we obtain $l_1 = [J_2, J_3, J_1, J_4]$ and $l_2 = [J_1, J_4, J_2, J_3]$ (line 7). Next, execute the **while loop** (lines 8–21). In the first cycle, the machine with C_{min} is M_1 and the next unassigned job on l_1 is J_2, whose corresponding processing speed is v_2 (line 9). Because $Te_{21} = 1 > 1/\sqrt{2}$ (line 10), execute lines 11–12. we obtain that $C_1 = 22.4$, $l_1 = [J_3, J_1, J_4]$, $l_2 = [J_1, J_4, J_3]$, $J^{a1}_{M_1} = [J_2]$, $V^{a1}_{M_1} = [v_2]$, $TwJ^{a1}_2 = 0.448$, $TwJ^{a1}_{M_1} = [0.448]$, $TwT_{1_1} = 0.448$, $J = [J_1, J_3, J_4]$ and $v = [v_1, v_2, v_2]$. Since $TwT_{1_1} = 0.448 < T$ (line 13), execute line 16 and we obtain $TJ^{a1}_{M_1} = [1_1]$. Repeat the **while loop** until $J = \varnothing$. After executing the **while loop**, we obtain that $J^{a1}_{M_1} = [J_2, J_3, J_4]$, $V^{a1}_{M_1} = [v_2, v_2, v_2]$, $TwJ^{a1}_{M_1} = [0.448, 0.16, 0.64]$,

Table 2. Relevant data of an example.

	p_{ij}				sp_j	pp_{jk}		w_k				
M	J_1	J_2	J_3	J_4	$v_0 = 0$	$v_1 = 1$	$v_2 = 1.25$	$v_0 = 0$	$v_1 = 1$	$v_2 = 1.25$	T	T_C
M_1	40	28	10	40	1	15	20	0	0.015	0.02	1	5
M_2	32	40	15	30	1	6	16					

$TwT^{a1}_{M_1} = [0.608]$, $TJ^{a1}_{M_1} = [1_1, 1_1, 1_2]$, $C_1 = 67.4$, $TwT_{1_2} = 0.64$ for M_1 and $J^{a1}_{M_2} = [J_1]$, $V^{a1}_{M_2} = [v_1]$, $TwJ^{a1}_{M_2} = [0.48]$, $TwT^{a1}_{M_2} = \emptyset$, $TJ^{a1}_{M_2} = [2_1]$, $C_2 = 32$, $TwT_{2_1} = 0.48$ for M_2. Then execute the **for loop** (lines 22–24), which is used to incorporate the final tool wear of the last tool used on each machine into TwT. After performing the **for loop**, we obtain that $TwT^{a1}_{M_1} = [0.608, 0.64]$ and $TwT^{a1}_{M_2} = [0.48]$. Finally, we output $J^{a1}_{M_1} = [J_2, J_3, J_4]$, $V^{a1}_{M_1} = [v_2, v_2, v_2]$, $TwJ^{a1}_{M_1} = [0.448, 0.16, 0.64]$, $TwT^{a1}_{M_1} = [0.608, 0.64]$, $TJ^{a1}_{M_1} = [1_1, 1_1, 1_2]$, $C_1 = 67.4$ for M_1 and $J^{a1}_{M_2} = [J_1]$, $V^{a1}_{M_2} = [v_1]$, $TwJ^{a1}_{M_2} = [0.48]$, $TwT^{a1}_{M_2} = [0.48]$, $TJ^{a1}_{M_2} = [2_1]$, $C_2 = 32$ for M_2.

4.3. A novel job-reassign heuristic

Job-reassign heuristic is an efficient method to optimise the UPMSP (Wu and Che 2019; Zheng and Wang 2018). Its main strategy is moving jobs from the machine with the C_{max} to another machine so that the C_{max} may be reduced. However, which job to move and how many jobs to move from the machine with C_{max} are difficult to decide. Some authors sacrifice the computing time to improve the result of job-reassign heuristic. For example, Wu and Che calculated the effect of moving every job and then decided which one to move (Wu and Che 2019). In this section, a novel job-reassign heuristic (NJRH) is designed. Algorithm 2 depicts the detail procedure for individual a in tth generation and the steps are as follows.

Step 1. Find $M_{\tilde{j}} := \arg max_{1\le j\le m} C_j$ and $M_{\tilde{j}'} := \arg min_{1\le j\le m} C_j$ for individual a; set $C_{max} := C_{\tilde{j}}$, $C_{min} := C_{\tilde{j}'}$.

Step 2. Calculate $\sigma = C_{max} - C_{min}$. Go to step 3 if $\sigma > CT$, where $CT = min(p_{\tilde{i}\tilde{j}}/v_k)$, $\tilde{i}$ is the index of jobs processed on $M_{\tilde{j}}$ and k is the corresponding speed level of processing $J_{\tilde{i}}$ on $M_{\tilde{j}}$; otherwise, end job-reassign heuristic for individual a.

Step 3. Let l denotes the position index of jobs in $J^{at}_{M_{\tilde{j}}}$, then set $l = 1$.

Step 4. Calculate the $C_{\widetilde{max}}$ under the assumption that the job in the l position on $M_{\tilde{j}}$ has been removed to $M_{\tilde{j}'}$. If $C_{\widetilde{max}} < C_{max}$, remove the job and its corresponding processing speed from $M_{\tilde{j}}$ to $M_{\tilde{j}'}$ and update JS, SS, TwJ, TwT, TJ for $M_{\tilde{j}}$ and $M_{\tilde{j}'}$, then go to step 1. Otherwise, go to step 5.

Step 5. Set $l = l + 1$. if $l \le |J^{at}_{M_{\tilde{j}}}|$, go to step 4. Otherwise, end job-reassign heuristic for individual a.

The detail procedure of the novel job-reassign heuristic is illustrated in Algorithm 2.

We use the same example listed in Section 4.2 to explain the process of Algorithm 2. Based on the output of M_1 and M_2 of individual a and executing lines 2–3, we obtain that $M_{\tilde{j}} = M_1$, $M_{\tilde{j}'} = M_2$, $C_{\widetilde{max}} = C_{max} = C_1 = 67.4$, $C_{\widetilde{min}} = C_{min} = C_2 = 32$ and get $RC_{max} = 0$. By performing line 4, $\sigma = 35.4$ and $CT = \min[p_{22}/v_2, p_{32}/v_2, p_{42}/v_2] = 12$. Then execute the **while loop** (lines 5–39) because of $\sigma > CT$ (line 5). For $l = 1$ (line 6), J_2 is in the lth position of $J^{a1}_{M_1}$, its processing speed is v_2 and $TwJ'_2 = p_{22}/v_2 \times w_2 = 0.64$ (lines 7–8). The **for loop** (lines 9–13) tries to find a tool used on M_2 to process J_2 without adding a new tool. Since no tool satisfies $TwT_{2_q} + TwJ'_2 \le T$ (line 10), it's true that $C_{\widetilde{min}} = = C_{min}$ (line 14), indicating that a tool change activity is required if remove J_2 from $J^{a1}_{M_1}$ to $J^{a1}_{M_2}$. Then execute line 15 and we obtain $C_{\widetilde{min}} = C_{min} + p_{22}/v_2 + T_C = 69$, which is obviously not feasible. For $l = 2$ (line 6), J_3 is in the lth position of $J^{a1}_{M_1}$, its processing speed is v_2 and $TwJ'_3 = p_{32}/v_2 \times w_2 = 0.24$ (lines 7–8). Performing the **for loop** (lines 9–13), we find that the first tool of M_2 satisfies $TwT_{2_1} + TwJ'_3 \le T$ (line 10). Then performing line 11, we obtain that $C_{\widetilde{max}} = C_{max}$ - $p_{31}/v_2 = 59.4$ and $C_{\widetilde{min}} = C_{min} + p_{32}/v_2 = 44$. Performing line 17, $C_{\widetilde{max}} := \max([C_{\widetilde{max}}, C_{\widetilde{min}} = 59.4$. **Break** the **for loop** (lines 6–21) because of $C_{\widetilde{max}} < C_{max}$ (line 18). lines 22–38 are used to actually perform the removal operations and prepare for the next **while loop**. Thus, remove J_3 from $J^{a1}_{M_1}$ to $J^{a1}_{M_2}$ and update JS, SS, TwJ, TwT, TJ for M_1 and M_2 (lines 23–33). After performing lines 22–33, we obtain that $J^{a1}_{M_1} = [J_2, J_4]$, $V^{a1}_{M_1} = [v_2, v_2]$, $TwJ^{a1}_{M_1} = [0.448, 0.64]$, $TwT^{a1}_{M_1} = [0.448, 0.64]$, $TJ^{a1}_{M_1} = [1_1, 1_2]$, $C_1 = 59.4$ for M_1 and $J^{a1}_{M_2} = [J_1, J_3]$, $V^{a1}_{M_2} = [v_1, v_2]$, $TwJ^{a1}_{M_2} = [0.48, 0.24]$, $TwT^{a1}_{M_2} = [0.72]$, $TJ^{a1}_{M_2} = [2_1, 2_1]$, $C_2 = 44$ for M_2. Executing lines 34–35, we obtain that $M_{\tilde{j}} = M_1$, $M_{\tilde{j}'} = M_2$, $C_{\widetilde{max}} = C_{max} = C_1 = 59.4$, $C_{\widetilde{min}} = C_{min} = C_2 = 44$, $\sigma = 15.4$, $CT = \min[p_{22}/v_2, p_{42}/v_2] = 24$ and $RC_{max} = 0$. End the Algorithm 2 because of $\sigma < CT$ (line 5). Finally, we output $J^{a1}_{M_1} = [J_2, J_4]$, $V^{a1}_{M_1} = [v_2, v_2]$, $TwJ^{a1}_{M_1} = [0.448, 0.64]$, $TwT^{a1}_{M_1} = [0.448, 0.64]$, $TJ^{a1}_{M_1} = [1_1, 1_2]$, $C_1 = 59.4$ for M_1 and $J^{a1}_{M_2} = [J_1, J_3]$, $V^{a1}_{M_2} = [v_1, v_2]$, $TwJ^{a1}_{M_2} = [0.48, 0.24]$, $TwT^{a1}_{M_2} = [0.72]$, $TJ^{a1}_{M_2} = [2_1, 2_1]$, $C_2 = 44$ for M_2.

4.4. A novel speed-adjusting heuristic

Ding, Song, and Wu (2016) indicates that the speed-adjusting heuristic can quickly and effectively reduce the energy consumption based on the assumption that the lower processing speed will result in less energy consumption for a job processed on a certain machine. According to the property of the problem proposed in this paper, a novel speed-adjusting heuristic (NSAH) is devised. Before describing the NSAH, a property is presented.

Algorithm 2: The novel job-reassign heuristic

1. Get $J^{at}_{M_j}$, $V^{at}_{M_j}$, $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$, $TJ^{at}_{M_j}$ and C_j for $1 \le j \le m$ of individual a in tth generation;
2. Find $M_{\tilde{j}} := \arg max_{1\le j\le m} C_j$, $M_{\tilde{j'}} := \arg min_{1\le j\le m} C_j$, then set $C_{\widetilde{max}} = C_{max} := C_{\tilde{j}}$, $C_{\widetilde{min}} = C_{min} := C_{\tilde{j'}}$;
3. Set $RC_{max} := \max C_j, j \in \{1, 2, \ldots, m\} \backslash \{\tilde{j}, \tilde{j'}\}$;
4. Calculate σ and CT;
5. **while** $\sigma > CT$ **do**
6. **for** $l := 1$ to $|J^{at}_{M_{\tilde{j}}}|$ **do**
7. Find the job J_i in the lth position of $J^{at}_{M_{\tilde{j}}}$ and its corresponding processing speed v_k;
8. Calculate the tool wear TwJ_i' caused by processing J_i on $M_{\tilde{j'}}$ at the same speed v_k;
9. **for** $q := 1$to $|TwT^{at}_{M_{\tilde{j'}}}|$ **do**
10. **if** $TwT_{j_q} + TwJ'_i \le T$ **then**
11. set $C_{\widetilde{max}} := C_{max} - p_{i\tilde{j}}/v_k$, $C_{\widetilde{min}} := C_{min} + p_{i\tilde{j'}}/v_k$; **break**;
12. **endif**
13. **endfor**
14. **if** $C_{\widetilde{min}} == C_{min}$ **then** % a tool change activity is required.
15. Set $C_{\widetilde{max}} := C_{max} - p_{i\tilde{j}}/v_k$; $C_{\widetilde{min}} := C_{min} + p_{i\tilde{j'}}/v_k + T_C$;
16. **endif**
17. Set $C_{\widetilde{max}} := \max([C_{\widetilde{max}}, C_{\widetilde{min}}, RC_{max}])$;
18. **if** $C_{\widetilde{max}} < C_{max}$ **then**
19. **break**;
20. **end**
21. **endfor**
22. **if** $C_{\widetilde{max}} < C_{max}$ **then** % execute the movement operations
23. **for** $q := 1$ to $|TwT^{at}_{M_{\tilde{j'}}}|$ **do**
24. **if** $TwT_{j_q} + TwJ_i{}' \le T$ **then**
25. Remove $J_i$from $J^{at}_{M_{\tilde{j}}}$into $J^{at}_{M_{\tilde{j'}}}$and remove v_k from $V^{at}_{M_{\tilde{j}}}$ into $V^{at}_{M_{\tilde{j'}}}$ respectively, and the positions are at the end of the
26. jobs and speed processed by qth tool, respectively. Update TwJ, TwT, TJ and makespan for $M_{\tilde{j}}$ and $M_{\tilde{j'}}$;
27. **break**;
28. **endif**
29. **endfor**
30. **if** $TwT_{j_q} + TwJ'_i > T$ for each $TwT_{j_q} \in TwT^{at}_{M_{\tilde{j'}}}$ **then**
31. Remove J_i into the end position of $J^{at}_{M_{\tilde{j'}}}$ and remove v_k into the end position of $V^{at}_{M_{\tilde{j'}}}$ respectively. Update TwJ, TwT, TJ and
32. makespan for $M_{\tilde{j}}$ and $M_{\tilde{j'}}$;
33. **endif**
34. Find $M_{\tilde{j}} := \arg max_{1\le j\le m} C_j$, $M_{\tilde{j'}} := \arg min_{1\le j\le m} C_j$; set $C_{\widetilde{max}} = C_{max} := C_{\tilde{j}}$, $C_{\widetilde{min}} = C_{min} := C_{\tilde{j'}}$; Calculate σ and CT;
35. Set $RC_{max} := \max C_j, j \in \{1, \ldots, m\} \backslash \{\tilde{j}, \tilde{j'}\}$;
36. **else**
37. break;
38. **endif**
39. **endwhile**
40. Output $J^{at}_{M_j}$, $V^{at}_{M_j}$, $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$, $TJ^{at}_{M_j}$ and C_j for $1 \le j \le m$;

Property 1. For individual a, whose C_{max} is occurred on $M_{\tilde{j}}$, a new individual a' which dominates a can be produced by slowing down the processing speed of one or a few jobs on other machines if the C_{max} isn't deteriorated.

Without changing the C_{max} of individual a, we expect a' consumes as little energy as possible by speed-adjusting heuristic. Thus, we need to solve a problem that which job on other machines should has speed-adjusting. The problem is equivalent to the knapsack problem. For a machine M_j ($M_j \in M \backslash \{M_{\tilde{j}}\}$), the $T_{gap,j} = C_{max}$-C_j can be defined as the capacity value and the job set $J^{at}_{M_j}$processed on M_j can be defined as item set, which consists of $|J^{at}_{M_j}|$ items h with profit of energy reduction ΔE_h and weight of time increase Δt_h when the job's processing speed decreases one level. For job J_i processed on M_j by qth tool with speed v_k ($k \ge 2$) in the tth generation, if k lowers one level, ΔE_i and Δt_i can be obtained by formulas (17) and (18) respectively, where ΔTwJ_i derived by formula (19) is the corresponding increase of tool wear. Based on the above analysis, we get that the speed-adjusting operate for J_i is feasible only it satisfies the conditions: $\Delta t_i \le T_{gap,j}$ and

Algorithm 3: The novel speed-adjusting heuristic

1. Get $J_{M_j}^{at}$, $V_{M_j}^{at}$, $TwJ_{M_j}^{at}$, $TwT_{M_j}^{at}$, $TJ_{M_j}^{at}$ and C_j for $1 \le j \le m$ of individual a in tth generation;
2. Find $M_{\tilde{j}} := \arg max_{1 \le j \le m} C_j$ and set $C_{max} = C_{\tilde{j}}$;
3. Calculate $T_{gap,j}$ for each $M_j \in M \setminus \{M_{\tilde{j}}\}$;
4. **for** each $M_j \in M \setminus \{M_{\tilde{j}}\}$ **do**
5. **for** $l := 1$ to $|J_{M_j}^{at}|$ **do**
6. Find the job J_i in the lth position of $J_{M_j}^{at}$, its corresponding processing speed v_k and tool index q it used;
7. **if** $(k \ge 2)$ && $(T_{gap,j} > 0)$ **then**
8. Calculate ΔE_i and Δt_i;
9. **if** $(\Delta t_i \le T_{gap,j})$ && $(\Delta E_i > 0)$ **then**
10. Set $v_k := v_{k-1}$, $T_{gap,j} := T_{gap,j} - \Delta t_i$; update TwJ, TwT, TJ and C_j for M_j;
11. **endif**
12. **endif**
13. **endfor**
14. **endfor**
15. Output $J_{M_j}^{at}$, $V_{M_j}^{at}$, $TwJ_{M_j}^{at}$, $TwT_{M_j}^{at}$, $TJ_{M_j}^{at}$ and C_j for $1 \le j \le m$;

$\Delta E_i > 0$. The procedure of the NSAH is shown as Algorithm 3.

$$\Delta E_i = \begin{cases} p_{ij}/v_k \cdot PP_{jk} - p_{ij}/v_{k-1} \cdot PP_{j(k-1)} & \text{if } TwT_{j_q} + \Delta TwJ_i \le T \\ p_{ij}/v_k \cdot PP_{jk} - p_{ij}/v_{k-1} \cdot PP_{j(k-1)} - SP_j \cdot T_C & \text{otherwise} \end{cases} \tag{17}$$

$$\Delta t_i = \begin{cases} p_{ij}/v_{k-1} - p_{ij}/v_k & \text{if } TwT_{j_q} + \Delta TwJ_i \le T \\ p_{ij}/v_{k-1} - p_{ij}/v_k + T_C & \text{otherwise} \end{cases} \tag{18}$$

$$\Delta TwJ_i = p_{ij}/v_{k-1} \cdot w_{k-1} - p_{ij}/v_k \cdot w_k \tag{19}$$

The Algorithm 3 is explained with an example whose relative data are shown in Table 2. Consider an individual a with $J_{M_1}^{at} = [J_2, J_4]$, $V_{M_1}^{at} = [v_2, v_2]$, $TwJ_{M_1}^{at} = [0.448, 0.64]$, $TwT_{M_1}^{at} = [0.448, 0.64]$, $TJ_{M_1}^{at} = [1_1, 1_2]$, $C_1 = 59.4$ for M_1 and $J_{M_2}^{at} = [J_1, J_3]$, $V_{M_2}^{at} = [v_1, v_2]$, $TwJ_{M_2}^{at} = [0.48, 0.24]$, $TwT_{M_2}^{at} = [0.72]$, $TJ_{M_2}^{at} = [2_1, 2_1]$, $C_2 = 44$ for M_2. Performing line 2, we find $M_{\tilde{j}} = M_1$ and $C_{max} = C_1 = 59.4$. Executing line 3, we obtain $T_{gap,2} = 15.4$ for M_2. Then perform the **for loop** (lines 4–14). We know that the **for loop** (lines 4–14) is only executed for M_2 because $M \setminus \{M_{\tilde{j}}\}$ is $\{M_2\}$. For $l = 1$(line 5), J_1 is in the lth position of $J_{M_2}^{a1}$, its processing speed is v_1 and the tool index q it used is 1 (line 6). Skip the **if** programme (lines 7–12) to line 5 for the reason $k = 1$. For $l = 2$ (line 5), J_3 is in the lth position of $J_{M_2}^{a1}$, its processing speed is v_2 and the tool index q it used is 1. Because of $k = 2 \ge 2$ and $T_{gap,2} = 15.4 > 0$ (line 7), perform line 8. We obtain that $\Delta E_3 = 102$ and $\Delta t_3 = 3$. Obviously, it is feasible to lower the speed level of J_3. Then performing line 10, we obtain that $J_{M_2}^{at} = [J_1, J_3]$, $V_{M_2}^{at} = [v_1, v_1]$, $TwJ_{M_2}^{at} = [0.48, 0.225]$, $TwT_{M_2}^{at} = [0.705]$, $TJ_{M_2}^{at} = [2_1, 2_1]$, $C_2 = 47$ for M_2 and $T_{gap,2} = 15.4\text{-}\Delta t_3 = 12.4$. End the Algorithm 3 for the reason that $l = 2$ (line 5) is equal to $|J_{M_2}^{at}|$ (the number of jobs in $J_{M_2}^{at}$). The C_{max} of individual a is unchanged, but its TEC has reduced by 102.

4.5. EP updating and hypermutation

Inspired by Gong et al. (2008), we introduce the concept of EP, which stores the optimal solutions searched so far. Then get the next generation population by performing hypermutation with mutation probability p_m on each solution in EP. Let M_{EP} denote the maximum number of EP and $N_{EP,t}$ represent the number of optimal solutions in EP at tth generation. The detailed operation steps are as follows:

Step 1. Use non-dominated sorting algorithm to rank the last individuals in tth generation. Select the individuals whose rank are 1, denoted them as *P(t)_rank1*.
Step 2. Put *P(t)_rank1* into EP and update EP. If $N_{EP,t} > M_{EP}$, calculate crowding distances for all the solutions in EP. Then sort them in non-increasing order of their crowding distances and select the first M_{EP} solutions into EP.
Step 3. Perform hypermutation on each individual in EP to generate offspring population.

Step 3.1. Generate a random number between 0 and 1, denoted by R_{n1}. If $R_{n1} < p_m$, randomly select two machines which are denoted as M_{s1} and M_{s2}. Then exchange part of jobs and their corresponding processing speeds of the two machines. Finally update *TwJ*, *TwT* and *TJ* for the two machines.

Step 3.2. For each machine except M_{s1} and M_{s2}, generate a random number between 0 and 1, denoted by R_{n2}. If $R_{n2} < p_m$, randomly select a few number of jobs and switch their processing speeds. Finally update *TwJ*, *TwT* and *TJ* for the machine.

4.6. Reducing tool changes based on FFD algorithm

At the end of the proposed algorithm, we use the *FFD* which is a classical method for solving bin packing problem to optimise the number of tool changes for each individual in *EP*. To better illustrate the significance of using the *FFD* algorithm, we introduce the Theorem 2.

THEOREM 2 *For a given job set and a corresponding speed level set of a machine, the optimal processing order must have the minimum number of tool changes.*

Proof For a machine, the $TEC = TEC_{pro} + TEC_{tc}$ and the makespan $C = C_{pro} + C_{tc}$. In the two equations, the subindexes *pro* and *tc* represent the processing mode and the tool changes mode respectively. Clearly, when the job set and the corresponding speed level set are given for a machine, TEC_{pro} and C_{pro} are fixed values. Thereby, the fewer number of tool changes, the smaller values of *TEC* and *C* will be. ■

Note that the worst-case ratio of the *FFD* is 3/2 (Simchi-Levi 1994). Thereby, if the number of tool changes after the *FFD* algorithm is not less than the number of tool changes before, do not execute the *FFD* algorithm. The detail procedure is shown in Algorithm 4.

Algorithm 4: Reducing tool changes based on *FFD* algorithm

1. Get $J^{at}_{M_j}$, $V^{at}_{M_j}$, $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$ and $TJ^{at}_{M_j}$ for $1 \le j \le m$ of individual a in *EP*;
2. **for** each $M_j \in M$ **do**
3. Use the *FFD* method to calculate the number of tool changes of each machine. If the number of tool changes isn't reduced for M_j, don't
4. execute the method for M_j; otherwise, use the *FFD* method to derive $J^{at}_{M_j}$, $V^{at}_{M_j}$, $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$ and $TJ^{at}_{M_j}$ for M_j.
5. **endfor**
6. Output $J^{at}_{M_j}$, $V^{at}_{M_j}$, $TwJ^{at}_{M_j}$, $TwT^{at}_{M_j}$ and $TJ^{at}_{M_j}$ for $1 \le j \le m$;

To illustrate explicitly, an example is present to explain how the Algorithm 4 works. Assume individual a with $J^{at}_{M_1} = [J_2, J_3, J_4]$, $V^{at}_{M_1} = [v_2, v_2, v_2]$, $TwJ^{at}_{M_1} = [0.48, 0.64, 0.5]$, $TwT^{at}_{M_1} = [0.48, 0.64, 0.5]$ and $TJ^{a1}_{M_1} = [1_1, 1_2, 1_3]$. According to the values of $TwJ^{at}_{M_1}$, the *FFD* algorithm can reorder it to [0.64, 0.5, 0.48]. Thus, J_4 and J_2 can be processed by a same tool for $0.5 + 0.48 \le T$ and the number of tool changes of M_1 is reduced from twice to one. Then update *JS*, *SS*, *TwT*, *TJ* for M_1. After performing the Algorithm 4, we obtain that $J^{at}_{M_1} = [J_3, J_4, J_2]$, $V^{at}_{M_1} = [v_2, v_2, v_2]$, $TwJ^{at}_{M_1} = [0.64, 0.5, 0.48]$, $TwT^{at}_{M_1} = [0.64, 0.98]$ and $TJ^{a1}_{M_1} = [1_1, 1_2, 1_2]$ for individual a.

5. Computational results

In this section, we evaluate the performance of the proposed NHEA for solving the UPMSPWTC. The algorithm is encoded in MATLAB R2016a and run on an Asus Notebook with Inter Core i7-8750 CPU at 2.20 GHz and 8GB RAM. We design three types of experiments to test the performance of NHEA. The first type of experiment is implemented to obtain the suitable combination of critical parameters; the second type of experiment is executed to prove the validity of the initial population produced by the list scheduling heuristic and the *FFD* algorithm; the third type of experiment is carried out to demonstrate the effectiveness of the NHEA by comparing with other two well-known algorithms, i.e. NSGA-II (Deb et al. 2002) and NNIA (Gong et al. 2008).

5.1. Instances constructing

The UPMSPWTC is first proposed, and accordingly, we construct 30 instances named $n \times m$ for testing, where $n \in \{300, 600, 800, 1000, 1500, 2000\}$, $m \in \{10, 20, 30, 40, 50\}$, $v \in \{0, 1, 1.25, 1.5, 1.75\}$, $w_k \in \{0, 0.015, 0.02, 0.03, 0.035\}$, $T = 1$, $T_C = 5$, $SP_j = 1$ for $1 \leq j \leq m$, the basic processing time of each job on each machine follows a uniform distribution between [5, 50] and the energy consumption rate of each machine at speed v_k ($k = 1, 2, \ldots, r$) belongs to [4, 45]. All the data of the instances can be downloaded via the link: https://github.com/likezhang1990/UPMSPWTC-instances.git. The method for designing the instances is described in Algorithm 5.

Algorithm 5: Generate Instances

1. Input n, m, v, $bound1 = [5, 50]$ and $bound2 = [4, 45]$;
2. set $v := v\backslash\{v_0\}$;
3. **for** $k = 1$:size(v,2) **do**
4. $Db(1,k) := bound2(1) \times v(k)/v(1)$;
5. Ub(1,k): = bound2(2) × v(k)/v(end);
6. **endfor**
7. **for** $j = 1$:m **do**
8. $W(j,:) :=$ 'randomly generate n integers, where each integer is between $bound1(1)$ and $bound1(2)$';
9. **for** $k = 1$:size(v, 2) **do**
10. Randomly generate a integer denoted by $E(j, k)$ between $Db(1, k)$ and $Ub(1, k)$; % $PP_{jk} = E(j, k)$.
11. **if** $k <$ size(v, 2) **then**
12. $Db(1, k + 1) = E(j, k) \times v(k + 1)/v(k)$;
13. **endif**
14. **endfor**
15. **endfor**
16. Output W and E;

An example is given to illustrate the Algorithm 5 explicitly. Suppose that the input values of n, m, v are 3, 2, and {0, 1, 1.25} respectively. After performing $v\backslash\{v_0\}$ (line 2), we obtain that $v = \{1, 1.25\}$ and size(v, 2) = 2. Execute the **for loop** (lines 3–6), by which we get $Db = [4, 5]$ and $Ub = [36, 45]$, where $Db(1, 1) = 4$ and $Ub(1, 1) = 36$ respectively denote the initial minimal and maximal power of each machine at speed $v_1 = 1$; $Db(1, 2) = 5$ and $Ub(1, 2) = 45$ respectively denote the initial minimal and maximal power of each machine at speed $v_2 = 1.25$. Db and Ub are generated base on the assumption that for a job to be processed on a certain machine, higher processing speed will lead to more energy consumption. Execute the **for loop** (lines 7–15). For $j = 1$, assuming $n = 3$ integers randomly generated between $bound1(1)$ and $bound1(2)$ are 25, 8 and 36 (line 8). This means $W(1,:) = [25, 8, 36]$. For $k = 1$ (line 9), suppose $E(1, 1) = 12 \in [Db(1, 1), Ub(1, 1)]$ (line 10). Lines 11–13 are used to adjust the minimal power of each machine at speed v_{k+1}. After performing line 12, we get $Db(1, 2) = 15$. Thus, for $k = 2$, the randomly generated integer $E(1, 2)$ should between $Db(1, 2) = 15$ and $Ub(1, 2) = 45$ (line 10). Suppose $E(1, 2) = 35$ (line 10). Then conduct the second time of **for loop** (lines 7–15). Let's suppose that $W(2,:) = [15, 18, 24]$, $E(2, 1) = 24$ and $E(2, 2) = 38$ after conducting this **for loop**. Finally, we obtain that $W = [25, 8, 36; 15, 18, 24]$ and $E = [12, 35; 24, 38]$. This indicates that $p_{11} = 25$, $p_{21} = 8$, $p_{31} = 36$, $p_{12} = 15$, $p_{22} = 18$, $p_{32} = 24$, $PP_{11} = 12$, $PP_{12} = 35$, $PP_{21} = 24$ and $PP_{22} = 38$, where p_{ij} is the basic processing time of job J_i processed on machine M_j, and PP_{jk} is the unit power of machine M_j at speed v_k.

5.2. Effectiveness metrics

In this paper, five well-known metrics are used to test the performance of NHEA. They are the set coverage (*SC*), the hypervolume indicator *(HV)*, the number of non-dominated solutions (N_{nd}), the rate of non-dominated solutions (R_{nd}) and the inverted generational distance (*IGD*), which can be depicted as [1]–[5] (Wu and Che 2019; Yuan and Xu 2015; Zitzler and Thiele 1999). Due to the complexity of the UPMSPWTC, the exact Pareto front is hardly to obtain especially for the large-scale instances. Thus, we use P^* which is the set of non-dominated solutions among all the solutions obtained by all runs of all algorithms to replace the the exact Pareto front.

[1] Let A and B be two approximate solution sets to the Pareto front. The set coverage C(A, B) represents the percentage of solutions in B that are dominated by at least one solution in A, i.e.

$$C(A, B) = \frac{|\{x \in B | \exists y \in A : y \text{ dominates } x\}|}{|B|} \tag{20}$$

If C(A, B) is large and C(B, A) is small, A is better than B in a sense.

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

$$HV(ref, A) = \lambda(\bigcup_{f \in A} [f_1, ref_1] \times \cdots \times [f_m, ref_m]) \tag{21}$$

Where $HV(ref, A)$ resolves the size of the space covered by an approximation set A, $r = (r_1, r_2, \ldots r_m)^{\mathrm{T}}$ refers to a chosen reference point from the target space, m here refers to the number of targets and λ is the Lebesgue measure. The higher the value of *HV*, the better *A* is.

[3] Let A be the Pareto front derived from an algorithm. The N_{nd} of the algorithm is the total number of solutions in A that remains in P^*, i.e.

$$N_{nd} = |\{x \in A | \forall y \in P^*, y \text{ nondominates } x\}| \tag{22}$$

[4] The R_{nd} of an algorithm is derived by Equation (23)

Table 3. Combinations of key parameters.

	Factor level	
Parameters	Small- and medium-scale instances	Large-scale instances
M_{EP}	{50, 75, 100, 125}	{50, 75, 100, 125}
p_m	{0.5/m, 0.75/m, 1/m, 1.25/m}	{0.75/m, 1/m, 1.25/m, 1.5/m}
MG	{50, 100, 150, 200}	{100, 150, 200, 300}

Table 4. The result of parameter analysis.

	Parameters			300 × 10	800 × 30	1500 × 50
Experiment	M_{EP}	p_m	MG	R_{nd}(%)	R_{nd}(%)	R_{nd}(%)
1	1	1	1	0.00%	0.00%	0.68%
2	1	2	2	1.12%	1.93%	2.71%
3	1	3	3	0.00%	1.45%	3.05%
4	1	4	4	6.70%	7.25%	4.41%
5	2	1	2	0.00%	0.00%	0.68%
6	2	2	1	0.00%	0.48%	0.00%
7	2	3	4	6.15%	18.84%	29.15%
8	2	4	3	13.41%	15.46%	3.73%
9	3	1	3	15.64%	0.00%	7.46%
10	3	2	4	31.84%	25.12%	9.83%
11	3	3	1	0.00%	0.00%	2.71%
12	3	4	2	4.47%	0.48%	7.80%
13	4	1	4	12.29%	13.04%	8.14%
14	4	2	3	5.59%	12.08%	16.61%
15	4	3	2	2.79%	3.86%	1.69%
16	4	4	1	0.00%	0.00%	1.36%

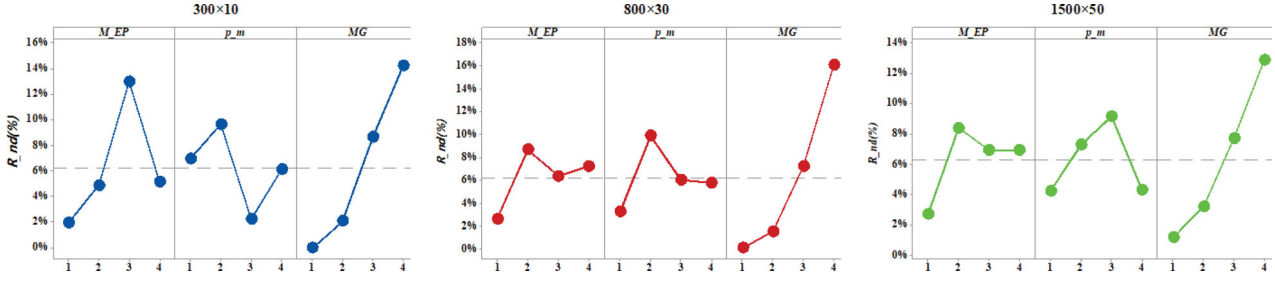

Figure 3. Factor level trend of key parameter.

$$R_{nd} = N_{nd}/|P^*| \tag{23}$$

[5] 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 x and y. The smaller $IGD(A, P^*)$ is, the closer A is to the Pareto front. Besides, the objectives of all solutions are normalised 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 among all solutions.

Table 5. Tuning results of key parameters for NHEA.

		Key parameters		
		M_{EP}	p_m	MG
Small-scale instances	opt. level	3	2	4
	opt. value	100	0.75/m	200
Medium-scale instances	opt. level	2	2	4
	opt. value	75	0.75/m	200
Large-scale instances	opt. level	2	3	4
	opt. value	75	1.25/m	300

Table 6. The SC and HV comparison of NHEA, EA1 and EA2**.**

		NHEA vs EA1		NHEA vs EA2		*HV*		
n	m	C(NHEA,EA1)	C(EA1,NHEA)	C(NHEA,EA2)	C(EA2,NHEA)	NHEA	EA1	EA2
300	10	1	0	1	0	**0.526133**	0.287679	0.525807
300	20	1	0	0.905512	0	**0.512856**	0.159558	0.512632
300	30	1	0	0.808081	0	**0.454934**	0.034316	0.454765
300	40	1	0	0	0	**0.431593**	0.023317	0.431326
300	50	1	0	0	0	**0.510247**	0.079509	0.509931
600	10	1	0	1	0	**0.533413**	0.217589	0.532623
600	20	1	0	0.992647	0	**0.545049**	0.149194	0.544727
600	30	1	0	0.963504	0	**0.572498**	0.148367	0.572224
600	40	1	0	0.936170	0	**0.589332**	0.162431	0.589236
600	50	1	0	0.924812	0	**0.515791**	0.046325	0.515646
800	10	1	0	1	0	**0.536977**	0.213073	0.536582
800	20	1	0	0.992754	0	**0.552659**	0.142846	0.552447
800	30	1	0	1	0	0.559756	0.122946	**0.559844**
800	40	1	0	0.944954	0	0.587006	0.136665	**0.587256**
800	50	1	0	0.934959	0	0.594156	0.127219	**0.594301**
1000	10	1	0	1	0	**0.557855**	0.229553	0.556706
1000	20	1	0	1	0	**0.573998**	0.166759	0.573818
1000	30	1	0	1	0	0.602688	0.171916	**0.602785**
1000	40	1	0	0.968000	0	**0.541527**	0.081400	0.541457
1000	50	1	0	0.834711	0	0.602917	0.144071	**0.602935**
1500	10	1	0	1	0	**0.551349**	0.226923	0.550005
1500	20	1	0	1	0	0.581265	0.178201	**0.581640**
1500	30	1	0	1	0	0.564029	0.118376	**0.564191**
1500	40	1	0	0.978571	0	**0.603963**	0.144826	0.603887
1500	50	1	0	0.970149	0	**0.588040**	0.113953	0.588039
2000	10	1	0	1	0	**0.571126**	0.273338	0.570335
2000	20	1	0	1	0	**0.638928**	0.258206	0.638553
2000	30	1	0	1	0	**0.611969**	0.181925	0.611729
2000	40	1	0	1	0	0.602199	0.150263	**0.602237**
2000	50	1	0	1	0	**0.611458**	0.143977	0.611071

5.3. Experimental results

5.3.1. Effectiveness of the key parameters

There are three key parameters in NHEA, which are the *EP* size M_{EP}, mutation probability p_m and the number of iterations *MG*. We use the design of experiment method proposed by Taguchi to obtain the suitable combination of the three key parameters. Inspired by the paper (Wang, Wang et al. 2018), the 30 instances are divided into three levels: (a) small-scale instances where $n \in \{300, 600\}$ and $m \in \{10, 20, 30, 40, 50\}$; (b) medium-scale instances where $n \in \{800, 1000\}$ and $m \in \{10, 20, 30, 40, 50\}$; (c) large-scale instances where $n \in \{1500, 2000\}$ and $m \in \{10, 20, 30, 40, 50\}$. Based on the experimental study on instances 300×10, 800×30 and 1500×50, the combinations of different values of the three parameters are list in Table 3. For each combination of the parameters, the NHEA runs ten times independently and the metric R_{nd} is chosen as the evaluation parameter of each combination. The parameter combinations and their results are shown in Table 4. Clearly, the bigger the R_{nd} is, the better the combination is.

Based on the results in Table 4, the trend of each key parameter level for R_{nd} is shown in Figure 3. The level with the maximum value of R_{nd} is the best one. Obviously, the most appropriate parameter combinations of NHEA for the proposed problem are suggested in Table 5.

5.3.2. Effect of the initial population and FFD algorithm

To prove the validity of the initial population produced by the list scheduling heuristic and the *FFD* algorithm, the NHEA is compared with EAs without the initial population produced by the list scheduling heuristic or the *FFD* algorithm using the metrics in Section 5.2. The EAs are denoted as EA1 and EA2, in which the initial population produced by the list scheduling heuristic and the *FFD* algorithm is not employed, respectively. To carry out a fair comparison, first, we set the same parameters got from Table 5 for the NHEA, EA1 and EA2, and the initial population size u is 100, which is a moderate

Table 7. The N_{nd}, R_{nd} and *IGD* comparison of NHEA, EA1 and EA2.

		NHEA			EA1				EA2		
n	*m*	N_{nd}	R_{nd}	*IGD*	N_{nd}	R_{nd}	*IGD*	*time/s*	N_{nd}	R_{nd}	*IGD*
300	10	119	1	0	0	0	0.264134	74.19	0	0	0.000420
300	20	124	0.911765	0	0	0	0.526848	71.49	12	0.0882353	0.000128
300	30	94	0.831858	0	0	0	0.983608	52.59	19	0.1681416	0.000082
300	40	90	0.5	0	0	0	1.026718	73.67	90	0.5	0.000000
300	50	56	0.5	0	0	0	0.842435	43.74	56	0.5	0.000000
600	10	97	1	0	0	0	0.454819	133.67	0	0	0.001155
600	20	137	0.992754	0	0	0	0.609183	97.32	1	0.0072464	0.000179
600	30	135	0.964286	0	0	0	0.654150	111.86	5	0.0357143	0.000091
600	40	92	0.938776	0	0	0	0.634784	112.03	6	0.0612245	0.000107
600	50	132	0.929577	0	0	0	0.982701	149.89	10	0.0704225	0.000049
800	10	107	1	0	0	0	0.498930	175.51	0	0	0.000885
800	20	136	0.992701	0	0	0	0.642033	118.69	1	0.0072993	0.000226
800	30	105	1	0	0	0	0.715782	119.28	0	0	0.000093
800	40	108	0.947368	0	0	0	0.713759	133.72	6	0.0526316	0.000066
800	50	122	0.938462	0	0	0	0.719130	119.81	8	0.0615385	0.000068
1000	10	99	1	0	0	0	0.483664	153.96	0	0	0.001047
1000	20	112	1	0	0	0	0.594465	136.05	0	0	0.000170
1000	30	135	1	0	0	0	0.597421	143.33	0	0	0.000136
1000	40	123	0.968504	0	0	0	0.856194	157.84	4	0.0314961	0.000079
1000	50	119	0.856115	0	0	0	0.691794	189.64	20	0.1438849	0.000255
1500	10	101	1	0	0	0	0.505077	746.29	0	0	0.001645
1500	20	136	1	0	0	0	0.611021	605.10	0	0	0.000359
1500	30	132	1	0	0	0	0.757590	454.79	0	0	0.000111
1500	40	140	0.979021	0	0	0	0.684444	342.05	3	0.020979	0.000051
1500	50	135	0.971223	0	0	0	0.769503	383.42	4	0.028777	0.000133
2000	10	113	1	0	0	0	0.460682	936.03	0	0	0.001279
2000	20	107	1	0	0	0	0.482593	393.00	0	0	0.000481
2000	30	151	1	0	0	0	0.610223	426.06	0	0	0.000182
2000	40	142	1	0	0	0	0.696493	445.28	0	0	0.000146
2000	50	140	1	0	0	0	0.702861	434.20	0	0	0.000048

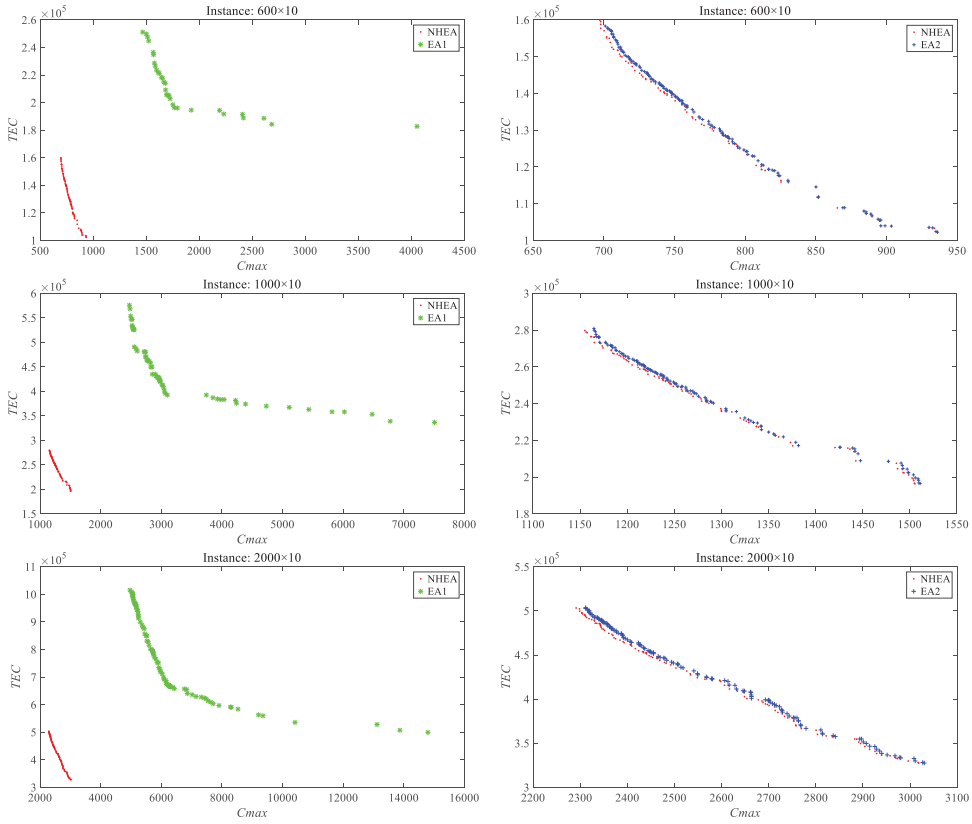

Figure 4. The Pareto front of some instances.

choice; second, we record each run time of EA1 and set it as the time limit for each run of EA2 when they deal with the same instance; third, execute the *FFD* algorithm on the end results of EA2 to get the results of NHEA (Because the run time of *FFD* algorithm is very short, it can be considered that the run time of NHEA is the same as that of EA1); fourth, each instance is run ten times independently by each algorithm. The comparison results of the ten independent runs are shown in Tables 6 and 7. The average time of ten independent runs for each instance computed by EA1 can be found in Table 7.

Tables 6 and 7 show that the initial population produced by the list scheduling heuristic can significantly improve the searching results. C(EA1, NHEA) = 0, N_{nd}(EA1) = 0 and R_{nd}(EA1) = 0 hold for all the test instances. This means that all the optimal results obtained by NHEA dominate those obtained by EA1. From the values of *IGD* and *HV*, we can obtain that *IGD*(EA1) > *IGD*(NHEA) = 0 and *HV*(NHEA) > *HV*(EA1) by a large margin hold for all the test instances, demonstrating that the initial population produced by the list scheduling heuristic has effectively increased the quality of Pareto fronts searched by EA1.

From the perspective of comparison of the NHEA and EA2, Tables 6 and 7 show that the *FFD* algorithm can greatly reduce the number of tool changes. It can be observed that the four metrics (*SC*, N_{nd}, R_{nd} and *IGD*) of NHEA are better than EA2 nearly for all test instances except instance 300 × 40 and instance 300 × 50, and the values of N_{nd}(EA2) and R_{nd}(EA2) show a downward trend with the increase of average number of jobs per machine. That is, the number of tool changes shows a rising trend as the number of jobs allocated to each machine increases and the probability of optimising the Pareto fronts by the *FFD* algorithm is raising correspondingly. From the perspective of *HV*, the values of *HV*(NHEA) and *HV*(EA2) are very close for all the test instances, indicating that the quality of the pareto fronts obtained by both algorithms is very good. But *HV*(NHEA) of 22 out 30 instances are slightly larger than *HV*(EA2), which indicates that NHEA outperforms EA2. Compared the *IGD*(EA2) with *IGD*(NHEA), we find that the Pareto fronts obtained by EA2 are considerably close to the Pareto fronts searched by NHEA.

To better prove the effectiveness of the initial population produced by the list scheduling heuristic and the *FFD* algorithm, choose one example from each of the three scale instances and depict their Pareto fronts obtained by the NHEA, EA1 and EA2, which are shown in Figure 4.

Table 8. The SC and HV comparison of NHEA, NSGA-II and NNIA.

		NHEA vs NSGA-II		NHEA vs NNIA		*HV*		
n	*m*	C(NHEA,NSGA-II)	C(NSGA-II,NHEA)	C(NHEA,NNIA)	C(NNIA,NHEA)	NHEA	NSGA-II	NNIA
300	10	1	0	1	0	**0.522537**	0.074118	0.344322
300	20	1	0	1	0	**0.496157**	0.017278	0.294024
300	30	1	0	1	0	**0.562454**	0.031906	0.321202
300	40	1	0	1	0	**0.571999**	0.026342	0.347830
300	50	1	0	1	0	**0.587470**	0.029237	0.299640
600	10	1	0	1	0	**0.377817**	0.011079	0.179880
600	20	1	0	1	0	**0.480579**	0.015134	0.193056
600	30	1	0	1	0	**0.535047**	0.013819	0.182183
600	40	1	0	1	0	**0.587814**	0.029909	0.243705
600	50	1	0	1	0	**0.577153**	0.026735	0.246655
800	10	1	0	1	0	**0.362697**	0.007522	0.177793
800	20	1	0	1	0	**0.471224**	0.008219	0.178064
800	30	1	0	1	0	**0.518901**	0.007999	0.140115
800	40	1	0	1	0	**0.564856**	0.009871	0.181597
800	50	1	0	1	0	**0.536917**	0.004104	0.157567
1000	10	1	0	1	0	**0.482898**	0.016812	0.224487
1000	20	1	0	1	0	**0.463467**	0.005799	0.102119
1000	30	1	0	1	0	**0.502039**	0.006397	0.092894
1000	40	1	0	1	0	**0.542746**	0.011474	0.148277
1000	50	1	0	1	0	**0.579143**	0.009255	0.152826
1500	10	1	0	1	0	**0.482580**	0.015709	0.204553
1500	20	1	0	1	0	**0.527832**	0.011901	0.211009
1500	30	1	0	1	0	**0.500822**	0.004376	0.115405
1500	40	1	0	1	0	**0.523636**	0.008607	0.161070
1500	50	1	0	1	0	**0.507769**	0.002652	0.146067
2000	10	1	0	1	0	**0.401707**	0.011187	0.101635
2000	20	1	0	1	0	**0.461992**	0.002005	0.097345
2000	30	1	0	1	0	**0.542886**	0.011690	0.129469
2000	40	1	0	1	0	**0.526901**	0.008283	0.135023
2000	50	1	0	1	0	**0.528044**	0.002686	0.084943

5.3.3. Comparison of NHEA, NSGA-II and NNIA

To further investigate the performance of the proposed NHEA, we compare it with two widely used multi-objective optimisation algorithms, i.e. NSGA-II and NNIA. To ensure fairness, we also use the running time of EA1 as the stopping criteria of NSGA-II and NNIA when they deal with the same instance. The population size, crossover probability and mutation probability of NSGA-II and NNIA are set 100, 1 and 0.15 respectively according to Gong et al. (2008). The comparison results of the three algorithms are shown in Tables 8 and 9.

In Tables 8 and 9, the five metrics of NHEA are much better than those of NSGA-II and NNIA for all the test instances, which means that the solutions obtained by NSGA-II or NNIA are all dominated by the Pareto fronts derived by NHEA. For the metric of *HV* in Table 8, *HV*(NNIA) bigger than *HV*(NSGA-II) by a large margin holds for all the test instances, demonstrating that NNIA outperforms NSGA-II in terms of Pareto fronts. In terms of the *IGD*, the values of *IGD*(NNIA) and *IGD*(NSGA-II) from Table 9 also indicate that NNIA performs better than NSGA-II.

6. Conclusions and future work

In this paper, a new UPMSP based on actual CNC workshop with tool changes caused by tool wear is studied. The targets are minimising both makespan and *TEC*. As for the UPMSPWTC, machines are speed-scaling and have different energy consumption rates. We not only analyse the influence of processing speed and processing time simultaneously on tool wear, but also establish the relationship between tool changes, *TEC* and makespan. To deal with the UPMSPWTC, the NHEA is designed, in which each individual is presented by five parts. To improve the search efficiency, a good initial population is produced by the list scheduling heuristic and two targeted searching operates are designed. To increase the diversity of optimal solutions, an elite population is introduced and mutated to generate offspring population. In addition, the *FFD* algorithm is incorporated at the end of the proposed algorithm, which can effectively reduce the number of tool changes and then optimise the bi-objective. Computational results of 30 instances constructed have demonstrated the high efficiency of

Table 9. The N_{nd}, R_{nd} and IGD comparison of NHEA, NSGA-II and NNIA.

		NHEA			NSGA-II			NNIA		
n	m	N_{nd}	R_{nd}	IGD	N_{nd}	R_{nd}	IGD	N_{nd}	R_{nd}	IGD
300	10	119	1	0	0	0	0.713574	0	0	0.214617
300	20	124	1	0	0	0	0.998423	0	0	0.272183
300	30	94	1	0	0	0	1.005875	0	0	0.282852
300	40	90	1	0	0	0	0.999162	0	0	0.298617
300	50	56	1	0	0	0	1.057441	0	0	0.369475
600	10	97	1	0	0	0	1.040583	0	0	0.328043
600	20	137	1	0	0	0	1.064503	0	0	0.425351
600	30	135	1	0	0	0	1.091578	0	0	0.514207
600	40	92	1	0	0	0	1.010332	0	0	0.431142
600	50	132	1	0	0	0	1.031861	0	0	0.441122
800	10	107	1	0	0	0	1.082718	0	0	0.310041
800	20	136	1	0	0	0	1.124653	0	0	0.469009
800	30	105	1	0	0	0	1.129729	0	0	0.573220
800	40	108	1	0	0	0	1.097983	0	0	0.534187
800	50	122	1	0	0	0	1.208170	0	0	0.577198
1000	10	99	1	0	0	0	0.882233	0	0	0.347634
1000	20	112	1	0	0	0	1.154367	0	0	0.643094
1000	30	135	1	0	0	0	1.157526	0	0	0.698513
1000	40	123	1	0	0	0	1.097338	0	0	0.595149
1000	50	119	1	0	0	0	1.088662	0	0	0.596060
1500	10	101	1	0	0	0	0.845978	0	0	0.370232
1500	20	136	1	0	0	0	0.968962	0	0	0.422119
1500	30	132	1	0	0	0	1.184417	0	0	0.638508
1500	40	140	1	0	0	0	1.145142	0	0	0.544096
1500	50	135	1	0	0	0	1.231695	0	0	0.575624
2000	10	113	1	0	0	0	0.994264	0	0	0.580100
2000	20	107	1	0	0	0	1.187736	0	0	0.643971
2000	30	151	1	0	0	0	1.047799	0	0	0.617862
2000	40	142	1	0	0	0	1.141686	0	0	0.607093
2000	50	140	1	0	0	0	1.210399	0	0	0.755534

the initial population produced by the list scheduling heuristic and the *FFD* algorithm. And the computational results also validate that the proposed NHEA performs much better than NSGA-II and NNIA.

For the further work, some other more efficient algorithms for the proposed problem could be developed to accelerate the convergence speed. Furthermore, another extension of the proposed model with different material of jobs could be investigated.

Acknowledgements

The authors are grateful to the editor and anonymous referees for their valuable comments and suggestions.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Key R&D Program of China [grant number 2018YFB1701400]; National Natural Science Foundation of China [grant number 71473077]; National Key Technology R&D Program of China [grant number 2015BAF01B00]; Project of State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University [grant number 71775004].

References

Abikarram, J. B., K. McConky, and R. Proano. 2019. "Energy Cost Minimization for Unrelated Parallel Machine Scheduling Under Real Time and Demand Charge Pricing." *Journal of Cleaner Production* 208: 232–242.

Aghelinejad, M., Y. Ouazene, and A. Yalaoui. 2018. "Production Scheduling Optimisation with Machine State and Time-Dependent Energy Costs." *International Journal of Production Research* 56 (16): 5558–5575.

Akturk, M. S., and S. Avci. 1996. "Tool Allocation and Machining Conditions Optimization for CNC Machines." *European Journal of Operational Research* 94 (2): 335–348.

Akturk, M. S., J. B. Ghosh, and E. D. Gunes. 2003. "Scheduling with Tool Changes to Minimize Total Completion Time: A Study of Heuristics and Their Performance." *Naval Research Logistics* 50 (1): 15–30.

Akturk, M. S., J. B. Ghosh, and E. D. Gunes. 2004. "Scheduling with Tool Changes to Minimize Total Completion Time: Basic Results and SPT Performance." *European Journal of Operational Research* 157 (3): 784–790.

Akturk, M. S., J. B. Ghosh, and R. K. Kayan. 2007. "Scheduling with Tool Changes to Minimize Total Completion Time Under Controllable Machining Conditions." *Computers & Operations Research* 34 (7): 2130–2146.

Arik, O. A., and M. D. Toksari. 2018. "Multi-objective Fuzzy Parallel Machine Scheduling Problems Under Fuzzy Job Deterioration and Learning Effects." *International Journal of Production Research* 56 (7): 2488–2505.

Astakhov, V. P., and J. P. Davim. 2008. *Tools (Geometry and Material) and Tool Wear*. Vol. 6795: Machining.

Baykasoglu, A., and F. B. Ozsoydan. 2018. "Minimisation of Non-Machining Times in Operating Automatic Tool Changers of Machine Tools Under Dynamic Operating Conditions." *International Journal of Production Research* 56 (4): 1548–1564.

Che, A., X. Wu, J. Peng, and P. Yan. 2017. "Energy-efficient Bi-Objective Single-Machine Scheduling with Power-Down Mechanism." *Computers & Operations Research* 85: 172–183.

Che, A., Y. Zeng, and K. Lyu. 2016. "An Efficient Greedy Insertion Heuristic for Energy-Conscious Single Machine Scheduling Problem Under Time-of-use Electricity Tariffs." *Journal of Cleaner Production* 129: 565–577.

Che, A., S. Zhang, and X. Wu. 2017. "Energy-conscious Unrelated Parallel Machine Scheduling Under Time-of-use Electricity Tariffs." *Journal of Cleaner Production* 156: 688–697.

Chen, J. S. 2008. "Optimization Models for the Tool Change Scheduling Problem." *Omega-International Journal of Management Science* 36 (5): 888–894.

Cheng, J., F. Chu, M. Liu, P. Wu, and W. Xia. 2017. "Bi-criteria Single-Machine Batch Scheduling with Machine on/off Switching Under Time-of-use Tariffs." *Computers & Industrial Engineering* 112: 721–734.

Costa, A., F. A. Cappadonna, and S. Fichera. 2016. "Minimizing the Total Completion Time on a Parallel Machine System with Tool Changes." *Computers & Industrial Engineering* 91: 290–301.

Davis, E., and J. M. Jaffe. 1981. "Algorithms for Scheduling Tasks on Unrelated Processors." *Journal of the Association for Computing Machinery* 28 (4): 721–736.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Ding, J. Y., S. Song, and C. Wu. 2016. "Carbon-efficient Scheduling of Flow Shops by Multi-Objective Optimization." *European Journal of Operational Research* 248 (3): 758–771.

Ding, J. Y., S. Song, R. Zhang, R. Chiong, and C. Wu. 2016. "Parallel Machine Scheduling Under Time-of-use Electricity Prices: New Models and Optimization Approaches." *IEEE Transactions on Automation Science and Engineering* 13 (2): 1138–1154.

EIA (U.S. Energy Information Administration). 2016. "International Energy Outlook 2016." Report No. DOE/EIA-0484(2016). Release date May 11, 2016. https://www.eia.gov/outlooks/archive/ieo16/.

Gahm, C.,F. Denz, M. Dirr, and A. Tuma. 2016. "Energy-efficient Scheduling in Manufacturing Companies: A Review and Research Framework." *European Journal of Operational Research* 248 (3): 744–757.

Gong, M., L. Jiao, H. Du, and L. Bo. 2008. "Multiobjective Immune Algorithm with Nondominated Neighbor-Based Selection." *Evolutionary Computation* 16 (2): 225–255.

Gray, A. E., A. Seidmann, and K. E. Stecke. 1993. "A Synthesis of Decision Models for Tool Management in Automated Manufacturing." *Management Science* 39 (5): 549–567.

Hao, L., L. Bian, N. Gebraeel, and J. Shi. 2017. "Residual Life Prediction of Multistage Manufacturing Processes With Interaction Between Tool Wear and Product Quality Degradation." *IEEE Transactions on Automation Science and Engineering* 14 (2): 1211–1224.

Hu, L., Y. Liu, C. Peng, W. Tang, R. Tang, and A. Tiwari. 2018. "Minimising the Energy Consumption of Tool Change and Tool Path of Machining by Sequencing the Features." *Energy* 147: 390–402.

Ji, W., Y. Yuan, B. Zou, S. Dai, and H. Zhang. 2018. "Friction and Wear Behaviour of Cemented Carbide Tool Materials Sliding Against Al2O3 and Si3N4 Ceramics Under Dry Condition." *Ceramics International* 44 (14): 17486–17491.

Jin, X., F. Zhang, L. Fan, Y. Song, and Z. Liu. 2015. "Scheduling for Energy Minimization on Restricted Parallel Processors." *Journal of Parallel and Distributed Computing* 81–82: 36–46.

Kouvelis, P. 1991. "An Optimal Tool Selection Procedure for the Initial Design Phase of a Flexible Manufacturing System." *European Journal of Operational Research* 55 (2): 201–210.

Li, K., Y. Shi, S. L. Yang, and B. Y. Cheng. 2011. "Parallel Machine Scheduling Problem to Minimize the Makespan with Resource Dependent Processing Times." *Applied Soft Computing* 11 (8): 5551–5557.

Li, Z., H. Yang, S. Zhang, and G. Liu. 2016. "Unrelated Parallel Machine Scheduling Problem with Energy and Tardiness Cost." *The International Journal of Advanced Manufacturing Technology* 84 (1–4): 213–226.

Liang, P., H. D. Yang, G. H. Liu, and J. H. Guo. 2015. "An Ant Optimization Model for Unrelated Parallel Machine Scheduling with Energy Consumption and Total Tardiness." *Mathematical Problems in Engineering* 2015: 1–8. Article ID 907034.

Lin, W., D. Y. Yu, C. Zhang, X. Liu, S. Zhang, Y. Tian, S. Liu, and Z. Xie. 2015. "A Multi-Objective Teaching – Learning-Based Optimization Algorithm to Scheduling in Turning Processes for Minimizing Makespan and Carbon Footprint." *Journal of Cleaner Production* 101: 337–347.

Liu, G. S., H. D. Yang, and M. B. Cheng. 2017. "A Three-Stage Decomposition Approach for Energy-Aware Scheduling with Processing-Time-Dependent Product Quality." *International Journal of Production Research* 55 (11): 3073–3091.

Mansouri, S. A., E. Aktas, and U. Besikci. 2016. "Green Scheduling of a Two-Machine Flowshop: Trade-off Between Makespan and Energy Consumption." *European Journal of Operational Research* 248 (3): 772–788.

Meng, L., C. Zhang, X. Shao, Y. Ren, and C. Ren. 2019. "Mathematical Modelling and Optimisation of Energy-Conscious Hybrid Flow Shop Scheduling Problem with Unrelated Parallel Machines." *International Journal of Production Research* 57 (4): 1119–1145.

Oda, Y., M. Mori, K. Ogawa, S. Nishida, M. Fujishima, and T. Kawamura. 2012. "Study of Optimal Cutting Condition for Energy Efficiency Improvement in Ball End Milling with Tool-Workpiece Inclination." *Cirp Annals-Manufacturing Technology* 61 (1): 119–122.

Rager, M., C. Gahm, and F. Denz. 2015. "Energy-oriented Scheduling Based on Evolutionary Algorithms." *Computers & Operations Research* 54: 218–231.

Shrouf, F., J. Ordieres-Mere, A. Garcia-Sanchez, and M. Ortega-Mier. 2014. "Optimizing the Production Scheduling of a Single Machine to Minimize Total Energy Consumption Costs." *Journal of Cleaner Production* 67: 197–207.

Simchi-Levi, D. 1994. "New Worst-Case Results for the Bin-Packing Problem." *Naval Research Logistics* 41 (4): 579–585.

Thevenin, S., N. Zufferey, and J.-Y. Potvin. 2017. "Makespan Minimisation for a Parallel Machine Scheduling Problem with Preemption and Job Incompatibility." *International Journal of Production Research* 55 (6): 1588–1606.

Wang, H., Z. Jiang, Y. Wang, H. Zhang, and Y. Wang. 2018. "A Two-Stage Optimization Method for Energy-Saving Flexible Job-Shop Scheduling Based on Energy Dynamic Characterization." *Journal of Cleaner Production* 188: 575–588.

Wang, S., X. Wang, J. Yu, S. Ma, and M. Liu. 2018. "Bi-objective Identical Parallel Machine Scheduling to Minimize Total Energy Consumption and Makespan." *Journal of Cleaner Production* 193: 424–440.

Wu, X., and A. Che. 2019. "A Memetic Differential Evolution Algorithm for Energy-Efficient Parallel Machine Scheduling." *Omega* 82: 155–165.

Xu, W., and L. Cao. 2014. "Energy Efficiency Analysis of Machine Tools with Periodic Maintenance." *International Journal of Production Research* 52 (18): 5273–5285.

Xu, D., and X. Shi. 2013. "Scheduling Tool Changes and Special Jobs on a Single Machine to Minimize Makespan." *Omega-International Journal of Management Science* 41 (2): 299–304.

Xu, Z., and D. Xu. 2018. "Single-machine Scheduling with Workload-Dependent Tool Change Durations and Equal Processing Time Jobs to Minimize Total Completion Time." *Journal of Scheduling* 21 (4): 461–482.

Yao, C. W., and T. W. Hong. 2019. "Evaluating Tool Wear by Measuring the Real-Time Contact Resistance." *The International Journal of Advanced Manufacturing Technology* 100 (9–12): 2349–2355.

Yazdani, M., S. M. Khalili, and F. Jolai. 2016. "A Parallel Machine Scheduling Problem with Two-Agent and Tool Change Activities: An Efficient Hybrid Metaheuristic Algorithm." *International Journal of Computer Integrated Manufacturing* 29 (10): 1075–1088.

Yuan, Y., and H. Xu. 2015. "Multiobjective Flexible Job Shop Scheduling Using Memetic Algorithms." *IEEE Transactions on Automation Science and Engineering* 12 (1): 336–353.

Zhang, R., and R. Chiong. 2016. "Solving the Energy-Efficient Job Shop Scheduling Problem: A Multi-Objective Genetic Algorithm with Enhanced Local Search for Minimizing the Total Weighted Tardiness and Total Energy Consumption." *Journal of Cleaner Production* 112: 3361–3375.

Zheng, X. L., and L. Wang. 2018. "A Collaborative Multiobjective Fruit Fly Optimization Algorithm for the Resource Constrained Unrelated Parallel Machine Green Scheduling Problem." *IEEE Transactions on Systems Man Cybernetics-Systems* 48 (5): 790–800.

Zitzler, E., and L. Thiele. 1999. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach." *IEEE Transactions on Evolutionary Computation* 3 (4): 257–271.