UNIFORM PARALLEL MACHINE SCHEDULING CONSIDERING THE MACHINE COST PER UNIT TIME

Leilei Tai\textsuperscript{1} Chao Xie\textsuperscript{2} and Kai Li\textsuperscript{2*}

\textsuperscript{1}Anhui University of Chinese Medicine, Hefei 230012, Anhui, China
\textsuperscript{2}Hefei Univ Technol, Sch Management, Hefei 230009, Anhui, China
48366867@qq.com

ABSTRACT: This paper tackles the pre-emptive and non-pre-emptive versions of a uniform parallel machine scheduling (UPMS) problem considering the machine cost. In the UPMS, each machine has a unique cost per unit time, and each job has a distinct makespan. Our goal is to develop a scheduling plan that minimizes the makespan under a specified limit on the total cost. A total of four heuristic algorithms were designed. Specifically, Algorithm A2 was developed based on level algorithm for machine selection; Algorithm A3 was presented for the pre-emptive problem, and proved to be optimal; Algorithm A4 was proposed for the non-pre-emptive problem, and verified experimentally that it was suitable to solve the problem. The research findings shed new light on the scheduling of a job-shop with uniform parallel machines.


1 INTRODUCTION

In recent years, much attention has been paid to the wide application of new information technology in manufacturing. For instance, (O’Rourke, 2014) held that new information technology promotes lean production, making it easier for companies to achieve consumer satisfaction. (Bonvillian, 2013) suggested that the embedment of information technology can benefit the entire manufacturing value chain, create intelligent production elements, enhance the manufacturing efficiency and extend the lifecycle of products.

The introduction new information technology to manufacturing has given birth to a novel manufacturing model known as cloud manufacturing. By this model, the use rights of machines are treated as a type of services, similar to the computing resources in cloud computing, and are open for purchase via the Internet. Cloud manufacturing mainly covers two aspects, namely, the online trading of the use rights of machines and the offline production process. The online-offline (O2O) pattern can integrate individual, discrete resource, produce a huge production capability and elevate the efficiency of machines. In classical scheduling theory, machines are assumed to be the property of manufacturers. In cloud manufacturing, however, machines are rented from their owner. Thus, the machine cost of the production process should not be neglected in cloud manufacturing.

In light of the above, this paper assumes that some machines with different processing speeds are acquired via the Internet, forming a uniform parallel machine production (UPMP) system, and that the cost of each machine per unit time is known before production. The purpose of our research is to minimize the makespan under a specified limit on the total cost.

The remainder of this paper is organized as follows: Section 2 reviews the related works on production scheduling considering machine cost; Section 3 describes the uniform parallel machine scheduling (UPMS) problem; Section 4 proposes a machine selection method under the constraint of the total cost; Sections 5 and 6 develop the algorithms for pre-emptive and non-pre-emptive problems, respectively; Section 7 wraps up this paper with some meaningful conclusions.

1. LITERATURE REVIEW

The research of production scheduling problems considering machine cost is still in its infancy. Imreh and Noga (1999) were the first to take account of machine cost in production scheduling. Assuming the cost of each machine as a unit, the competitive ratio of online scheduling for minimal makespan was determined as $\frac{1 + \sqrt{5}}{2}$ by the classical list scheduling (LS) algorithm, and that of online scheduling with different makespans for jobs as $\frac{6 + \sqrt{205}}{12}$ by the same algorithm. Focusing on semi-online scheduling problems, He & Cai (2002), Jiang and He (2005 and 2006) hypothesized...
that the total makespan or the maximum makespan of jobs are known before any job is scheduled. Based on Tabu search, Cao et al., (2005) constructed a heuristic algorithm for machine maintenance cost and delay cost. Dosa and He, (2006) included penalty cost into machine cost, and computed the corresponding competitive ratio as 2 for online scheduling.

Imreh and Nagy (2007) set the cost of each machine to 1, and introduced the penalty cost to online scheduling. Imreh (2009) divided the machine costs to a low sub-cost and a high sub-cost, maximized the use of cheap machines in the scheduling process. Jiang et al., (2014) explored an online scheduling problem which allows the purchase of new machines, and attempted to minimize the total makespan and the total cost of all purchased machines. Taking the cost of each machine as K, Rustogi and Strusevich, (2013) created a novel objective function “increasing the number of jobs from m to m" while the classical objective is to minimize the total makespan. Leung et al., (2012) investigated several types of dual-objective, unrelated machine scheduling problems, under the assumption that a cost $c_{ij}$ is incurred in the processing of any job $j$ on any machine $M_i$, and that the processing cost of a job differs with the machines. Huang et al., (2013) established an optimization model to minimize the makespan and machine maintenance cost of parallel machine scheduling problem.

In view of these studies on production scheduling problems considering machine cost, this paper attempts to minimize the makespan of UPMS problems under the constraint of machine cost.

2. PROBLEM DESCRIPTION

Let $M = \{M_i | i = 1, 2, ..., m\}$ be a set of m machines with different processing speeds, and $s_i$, $i = 1, 2, ..., m$ be the processing speed of machine $M_i$. Without loss of generality, it is assumed that $s_1 \geq s_2 \geq \cdots \geq s_i \geq \cdots \geq s_m$, and that each machine has a unique cost per unit time.

In reality, the processing speed is positively correlated with the level of technology. Any machine that runs slowly at a high cost per unit time will be scrapped. Hence, the machine cost is assumed to be a non-decreasing function of machine speed, that is, the faster the speed, the higher the cost. Here, the cost of machine $M_i$ is denoted as $u_i$ ($u_i > 0$), and the ratio of $s_i/u_i$ is considered as a non-decreasing of the level of technology.

In addition, the number of jobs to be processed is denoted as n, and the makespan of job $j$ as $p_j$, $j = 1, 2, ..., n$, when it is processed at the machine speed of 1. It is also assumed that $p_1 \geq p_2 \geq \cdots \geq p_j \geq \cdots \geq p_n$. Here, a machine can only allow to process one job at a time, and a job can only be processed on one machine at a time.

Let $\sigma$ be a feasible scheduling plan for our scheduling problem, and $C_j(\sigma)$ be the makespan of job $j$. Then, the maximum makespan on machine $M_i$ under plan $\sigma$ can be expressed as $C_{\max}(\sigma)$. In this case, the cost of machine $M_i$ is $U_i(\sigma) = u_i \cdot C_{\max}(\sigma)$, the total cost of $\sigma$ is $TC(\sigma) = \sum_{i=1}^{m} U_i(\sigma)$, and the maximum makespan of $\sigma$ is $C_{\max}(\sigma) = \max_{i=1}^{m} C_{\max}(\sigma)$.

For simplicity, the notation $\sigma$ is omitted. Our goal is to find a plan that minimizes the makespan, provided that the total cost does not exceed the limit $\overline{U}$.

In this paper, the pre-emptive problem as $Q_{tn}|pmtn, TC \leq \overline{U}|C_{\max}$ and the non-pre-emptive problem $Q_{tn}|TC \leq \overline{U}|C_{\max}$ are discussed using the three-field notation proposed by (Graham et al., 1979). The optimal algorithm for the pre-emptive problem is given in Section 5, while the heuristic algorithm for the non-pre-emptive problem is proposed in Section 6. The heuristic solution of the non-pre-emptive problem was verified through the evaluation using the optimal objective value of the pre-emptive problem.

3. MACHINE SELECTION METHOD

To solve the UPMS problem with machine cost, the machines should be selected under the limit on the total cost. This section puts forward a machine selection method for the pre-emptive problem, which is also applied to the non-pre-emptive problem.

Without considering the machine cost, the pre-emptive problem $Q_{tn}|pmtn|C_{\max}$ can be solved by the following level algorithm (Horvath et al., 1977).

4.1 Level algorithm:

Step 1: Determine the processing speed of each job, and identify the pre-emptive plan based on the processing speeds. Initialize the time for machine assignment $q$ as zero.

Step 2: Let $j$ be the number of unassigned machines at time $q$, and $k$ be the number of jobs to be processed at the highest level at time $q$. If $k \leq j$, assign the $k$ jobs to be processed at the same speed on the $k$ fastest machines; otherwise, assign the $k$ jobs to be processed at the same speed on the $j$ machines. If there are idle machines, repeat the above procedure for the jobs to be processed at the next highest level.

Step 3: Repeat the previous steps until one of the following conditions is satisfied:
(1) A job is completed at $t$.
(2) There exist two jobs $T$ and $T'$ such that $Level_q(T) > Level_q(T')$ but $Level_q(T) = Level_q(T')$. In other words, $T$ has caught up with $T'$ at time $t$.

In either case, set $q = t$ and reassign all machines to the jobs not yet processed.

Step 4: To construct a pre-emptive plan from the schedule in Step 2, assign the part of the schedule between every pair of events as follows:

If $k$ jobs have been sharing $j$ machines, $k \geq j$, then each of the $k$ jobs will face the same service requirements in the interval, starting from Step 2; divide the interval into $k$ equal subintervals, and assign the $k$ jobs in such a manner that each job has exactly $j$ subintervals, during each of which on a different machine.

Let $F^*_\text{ncp}$ be the optimal makespan of the problem $Q_m|p|\text{max}C_{\text{max}}$ obtained by level algorithm. Then, $F^*_\text{ncp} = \max \{ \max_{1 \leq k \leq m}(P_k/S_k, P/S) \}$, where $P_k = \sum_{j=1}^{k} p_j$, $S_k = \sum_{i=1}^{k} s_i$, $P = \sum_{j=1}^{m} p_j$ and $S = \sum_{i=1}^{m} s_i$. The $F^*_\text{ncp}$ is the lower bound of the problem $Q_m|p|\text{max}C_{\text{max}}$ for the corresponding problem without considering machine cost can be viewed as a relaxed version of our problem. Let $F^*_{\text{cp}}$ be the optimal makespan of the problem $Q_m|p|\text{min}TC \leq \bar{U}|\text{max}C_{\text{max}}$. Then, we have $F^*_{\text{cp}} \geq F^*_\text{ncp}$.

There are two forms of the optimal solution to the problem $Q_m|p|\text{max}C_{\text{max}}$ as obtained by the level algorithm. If the solution exists in Form 2 (Figure 1(b)), some jobs can be reassigned from machines with larger subscripts to those with smaller subscripts, such as to fully load the faster machines and reduce the total machine cost.

**Figure 1** Two forms of the optimal solution to the problem $Q_m|p|\text{max}C_{\text{max}}$

Assuming that $M_k$ ($k < m$) is first machine that satisfies $C_{\text{max}} < F^*_\text{ncp}$ in $\{M_i|i = 1, 2, ..., m\}$, the jobs on machine $M_i(k + 1 \leq i \leq m)$ can be reassigned to faster machines as much as possible. In this way, it is possible to achieve the minimal makespan $F^*_\text{ncp}$ without using up all $m$ machines. Then, a solution can be derived which uses $K(k \leq m)$ machines, in which machine $K$ is not fully loaded. In this case, Algorithm A1 was developed to select the machines, i.e. determine the value of $K$, under a relaxed limit on the total cost.

Algorithm A1:

Step 1: Call level algorithm to get $F^*_\text{ncp}$.

Step 2: Let $sum = 0$ and $K = 0$.

Step 3: While ($K < m$ and $F^*_\text{ncp} \cdot \sum_i s_i < P \cdot u_k + F^*_\text{ncp} \cdot \sum_i u_i$)

   $K = K + 1$; $sum = sum + sK.$

Step 4: Return $K$.

**Theorem 1**: If the first $K$ machines are selected and the makespan equals $F^*_\text{ncp}$, then the total cost $TC = (P \cdot F^*_\text{ncp} \cdot \sum_{i=1}^{K-1} s_i) \cdot u_k + F^*_\text{ncp} \cdot \sum_{i=1}^{K-1} u_i$ is minimum.

This theorem is clearly valid. Obviously, there exists a feasible solution such that all the first $K$ machines run under full load, machine $K$ processes the remaining jobs, and the total cost reaches the minimum. In this case, the total cost stands at $TC$.

Under the assumption that $s_i/u_i = \max_{1 \leq k \leq m}(s_i/u_i)$, if $TC = P \cdot u_k/s_k$, then $TC$ is the minimal feasible total cost. For a given $\bar{U}$, there exist the following three conditions:

1. $\bar{U} < TC$;
2. $\bar{U} \geq TC$;
3. $TC < \bar{U} < TC$.

The problem has no feasible solution in condition (1), and $K$ machines can be selected by Algorithm A1 in condition (2).

In conditions (3), there must be ($\sum_{i=1}^{K} u_i$) \cdot $P/\sum_{i=1}^{K} s_i$ \geq $\bar{U}$ if the first $K$ machines are selected. Since the ratio of $s_i/u_i$ is non-decreasing with $s_i$, the first $k$ that makes ($\sum_{i=1}^{k} u_i$) \cdot $P/\sum_{i=1}^{k} s_i$ \geq $\bar{U}$ should be selected to minimize the total cost. Here, ($\sum_{i=1}^{k-1} s_i$) \cdot $P/\sum_{i=1}^{k-1} s_i$ < $\bar{U}$. Since $TC \geq \bar{U} < TC$, it is obvious that $k \leq K$, and the makespan must exceed $F^*_\text{ncp}$.

Algorithm A2:

Step 1: If ($\bar{U} < TC$), then there is no feasible solution, end.

Step 2: If ($\bar{U} \geq TC$), then {call $A_1$; $k = K$; go to Step 5.}

Step 3: $s_{\text{sum}} = s_k$; $u_{\text{sum}} = u_k$; $k = 1$.

Step 4: While ($k < K$ and $u_{\text{sum}} \cdot P/s_{\text{sum}} < \bar{U}$)

   $k = k + 1$; $s_{\text{sum}} = s_{\text{sum}} + s_k$; $u_{\text{sum}} = u_{\text{sum}} + u_k$.

Step 5: Return $k$.

4. **PRE-EMPTIVE PROBLEM**

Based on the machine selection method, this section presents an optimal polynomial time algorithm for the problem $Q_m|p|\text{min}TC \leq \bar{U}|\text{max}C_{\text{max}}$. The makespan is deemed as the lower bound for the problem $Qm|TC \leq \bar{U}|C_{\text{max}}$ in Section 6.
According to Algorithm A2, the first $k (k < m)$ machines were selected for our problem. Since $C_{\text{max}}^i = C_{\text{max}}$ for all $i = 1, 2, \ldots, k-1$, the following two equations are valid at the same time

\begin{align*}
C_{\text{max}} \cdot \sum_{i=1}^{k-1} u_i + C_{\text{max}}^k \cdot u_k &= \bar{U} \\
C_{\text{max}} \cdot \sum_{i=1}^{k-1} s_i + C_{\text{max}}^k \cdot s_k &= P
\end{align*}

Then, the makespan can be derived as:

\begin{equation}
C_{\text{max}} = \frac{P \cdot s_k - P \cdot u_k}{s_k \cdot \sum_{i=1}^{k-1} u_i - u_k \cdot \sum_{i=1}^{k-1} s_i}
\end{equation}

and

\begin{equation}
C_{\text{max}}^k = \frac{P - C_{\text{max}} \cdot \sum_{i=1}^{k-1} s_i}{s_k}
\end{equation}

Next, the level algorithm was modified into Algorithm A3 to solve the problem.

**Algorithm A3:**

**Step 1:** Call Algorithm A2 to determine the number of the selected machines $k$.

**Step 2:** If ($k = K$), then {$c = 0$; $U = \bar{U}$}; otherwise, compute $C_{\text{max}}$ and $C_{\text{max}}^k$ by equations (3) and (4), respectively.

**Step 3:** Call level algorithm to sort all the jobs on the first $k$ machines until reaching $C_{\text{max}}^k$.

**Step 4:** Call level algorithm to sort the remaining jobs on the first $k-1$ machines.

**Theorem 2:** The solution obtained by Algorithm A3 is the optimal solution to the problem $Q_m|\text{pmtn}, TC \leq \bar{U}|C_{\text{max}}$.

The proof of this theorem is quite easy. If $C_{\text{max}}^k = C_{\text{max}}$, the solution is obviously the optimal one by the level algorithm; if $C_{\text{max}}^k < C_{\text{max}}$, all the jobs on the $k$ machines can be sorted by level algorithm until reaching $C_{\text{max}}^k$, and the remaining jobs on the remaining $k-1$ machines can also be sorted. Under the constant total makespan, it is obviously optimal to sort the remaining jobs on the first $k$ machines until reaching $C_{\text{max}}^k$ according to the nature of level algorithm. Q.E.D.

5. **NON-PRE-EMPTIVE PROBLEM**

6.1 Heuristic algorithm

This section mainly tackles the problem $Q_m|TC \leq \bar{U}|C_{\text{max}}$. The longest processing time (LPT) algorithm, a general heuristic rule to minimize the makespan of parallel machine scheduling problems, was modified into a heuristic algorithm A4, for $Q_m|C_{\text{max}}$ is a relaxation of the problem $Q_m|TC \leq \bar{U}|C_{\text{max}}$.

**Algorithm A4:**

**Step 1:** Call Algorithm A3 to solve the problem $Q_m|\text{pmtn}, TC \leq \bar{U}|C_{\text{max}}$, and determine the values of $k$, $C_{\text{max}}$ and $C_{\text{max}}^k$. Then, denote $C_{\text{max}}$ as $R$, and $C_{\text{max}}$ as $F_{\text{cp}}$.

**Step 2:** Set $U$ to 0. For all $i \leq k$, set $S_i$ to $\emptyset$ and $t_i$ to 0. Let $J = \{J_1, J_2, \ldots, J_n\}$. Set a counter $c = 2$.

**Step 3:** Sort the jobs in $J$ by the LPT. Denote the first job in $J$ as $J_f$ and its process time as $p_{J_f} = \arg \min_{i \leq k}(t_i + \frac{p_{J_i}}{s_i})$. If multiple machines correspond to the minimal value of $t_i + \frac{p_{J_i}}{s_i}$, then choose the machine with the smallest subscript. Set a counter $c = 1$.

**Step 4:** If ($U + \frac{p_{J_f}}{s_i} \cdot u_i \leq \bar{U}$), then go to Step 8.

**Step 5:** Set $l$ to 1. If ($U + \frac{p_{J_f}}{s_i} \cdot u_i \leq \bar{U}$), then go to Step 8.

**Step 6:** While ($p_{J_f} < p_{J_f+1}$ or $p_{J_f} \in S_1$) and $f - c + 1 > c$

\begin{itemize}
  \item If ($c \neq 1$), then
    \begin{itemize}
      \item Assuming that job $J_f+1$ is processed on machine $M_i$ at the current moment, $S_i = S_i \cup \{J_f+1\}$; $t_i = t_i + \frac{p_{J_f+1}}{s_i}; U = U - \frac{p_{J_f+1}}{s_i} \cdot u_i; J = J \cup \{J_f+1\}$.
    \end{itemize}
  \item If ($f - c = c$) $c' = c + 1$;
  \item $c = c + 1$.
\end{itemize}

**Step 7:** Assuming that job $J_f-c$ is processed on machine $M_i$ and $J_f-c+1$ on machine $M_i'$, If ($c \neq 1$), then

\begin{itemize}
  \item {Assuming that job $J_f-c+1$ is processed on machine $M_i$ at the current moment, $S_i = S_i \cup \{J_f-c+1\}$; $S_1 = S_1 \cup \{J_f-c+1\}$.}
  \item $t_i = t_i + \frac{p_{J_f-c+1}}{s_i}; t_1 = t_1 + \frac{p_{J_f-c}}{s_i}; U = U - \frac{p_{J_f-c}}{s_i} \cdot u_i; J = J \cup \{J_f-c+1\}$.}
  \item Otherwise,
    \begin{itemize}
      \item $S_i = S_i \cup \{J_f-c+1\}; S_1 = S_1 \cup \{J_f-c\}; t_1 = t_1 + \frac{p_{J_f-c}}{s_i}; U = U - \frac{p_{J_f-c}}{s_i} \cdot u_i + \frac{p_{J_f-c}}{s_i} \cdot u_1$.
    \end{itemize}
\end{itemize}

Go to Step 3.

**Step 8:** If ($t_1 + \frac{p_{J_f}}{s_i} > F_{\text{cp}}$ and $t_k + \frac{p_{J_f}}{s_k} \leq R$ and $U + \frac{p_{J_f}}{s_k} \cdot u_k \leq \bar{U}$);

\begin{itemize}
  \item Then, {Set $k = S_k \cup \{J_f\}; S_k = S_k + \{J_f\}; t_k = t_k + \frac{p_{J_f}}{s_k}; U = U + \frac{p_{J_f}}{s_k} \cdot u_k$; $J = J \backslash \{J_f\}$; } otherwise, {Set $J = J \backslash \{J_f\}$}. If ($J \neq \emptyset$), then go to Step 3.
Step 9: For all \( 1 \leq i \leq k \), assign \( S_i \) to machine \( M_i \).
\[ C_{\text{max}} = \max_{i=1}^{k} C_{i_{\text{max}}}^{\text{algorithm}} \]. Return \( C_{\text{max}} \).

Let cost be the current machine cost to process a job. If cost > \( \bar{U} \), the nearest job can be determined via Step 6 on the machine different from \( M_1 \) and whose makespan is longer than \( pf \). Then, the jobs determined for \( M_2 \) in Step 6 can be sorted in Step 7. If cost \( \leq \bar{U} \), the current job can be sorted directly by Step 8.

**Theorem 3:** The plan produced by Algorithm A4 is a feasible solution for the problem \( Q_m|TC \leq \bar{U}|C_{\text{max}} \).

**Proof:** Algorithm A4 combines the LPT algorithm and Algorithms A1, A2 and A3 to sort the jobs. The core idea of this algorithm is as follows: once cost > \( \bar{U} \), the most adjacent job with a longer makespan yet does not exist in \( M_1 \) should be found, and then sorted on \( M_1 \). This strategy can either reduce the machine cost or minimize the makespan. Next, three different cases of job makespan were discussed in details.

1. Case 1: \( p_1 > p_2 > \ldots > p_n \), i.e. the job makespans are strictly unequal. The cost decreases whenever Steps 6 and 7 are implemented. Even in the extreme case limit(\( U - TC \)) = 0, all the jobs will be sorted on \( M_1 \) after a few cycles. In this case, cost = \( TC \), satisfying cost \( \leq \bar{U} \). Hence, the plan produced by Algorithm A4 is a feasible solution.

2. Case 2: \( p_1 = p_2 = \ldots = p_n \), i.e. the job makespans are identical. Because of the value of counter \( c^\prime \), the \((c^\prime - 1)\)-th job will be sorted on \( M_1 \) in every cycle. Even in the extreme case limit(\( U - TC \)) = 0, all the jobs will be sorted on \( M_1 \) after a few cycles. In this case, cost = \( TC \), satisfying cost \( \leq \bar{U} \). Hence, the plan produced by Algorithm A4 is a feasible solution.

3. Case 3: \( p_1 \geq p_2 \geq \ldots \geq p_n \), i.e. the job makespans are not always equal. Since Case 3 is a combination of Cases 1 and 2, it is obvious that the plan produced by Algorithm A4 is a feasible solution.

Through Steps 6 and 7, it is guaranteed that the current total machine cost will not exceed the limit \( \bar{U} \) after sorting any job, which proves the feasibility of Algorithm A4.

### 6.2 Experimental verification

Using VC6.0 code, the author built a computer model based on random data to verify the performance of the proposed algorithm. The experiment was conducted on a computer with a memory of 4GB and running on Intel® Core™ i5 processor (2.40GHz) and Windows 7 SP1.

The number of machines \( m \) was selected for the four cases from \{4, 8\}. The machine speed \( s_i \) was randomly generated from the interval \[1, 8\]. The unit cost of each machine was determined according to the machine speed, and was controlled within the interval \[1, 15\]. During the experiment, the machine speed was generated randomly before determining the machine cost satisfying \( \frac{s_i}{u_i} > \frac{s_i}{u_{i+1}} \) by computer simulation. The parameter setting of the machines is listed in Table 1 below. The job-machine relationship is denoted as \( n = (5 * i) * m, i = 1, 2, 3, 4, 5 \). The job makespan \( p_j \) was set to different ranges depending on the specific case.

<table>
<thead>
<tr>
<th>m</th>
<th>S</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>4</td>
<td>{6,5,3,1}</td>
</tr>
<tr>
<td>Case 2</td>
<td>8</td>
<td>{8,5,4,3,3,3}</td>
</tr>
</tbody>
</table>

Tables 2 and 4 show the experimental results for \( p_{\text{max}} = 20 \) problem sets and \( p_{\text{max}} = 80 \) problem sets, respectively. The makespan of each job was initialized randomly. The cost interval was computed according to the method in Section 3. Then, the \( \bar{U} \) was selected within the range of \( \bar{U} = TC + \lambda \cdot (TC - TC) \). The value of \( \lambda \) was set to 0.2, 0.5, and 0.8 in the three cases, respectively.

Since F \(_{\text{cp}}^{\text{**}} \) is the lower bound of the non-preemptive problem \( Q_m|TC \leq \bar{U}|C_{\text{max}} \), the value of F \(_{\text{cp}}^{\text{**}} \) was adopted to evaluate the Fc obtained by Algorithm A4. Three groups of examples were provided for each \( n \) and \( m \). The Gap value of each example was computed by Gap = \( \frac{\text{F}_{\text{cp}}^{\text{*}} - \text{F}_{\text{cp}}^{(*)}}{\text{F}_{\text{cp}}^{(*)}} \times 100 \). Here, Gap is the relative error bound of F \(_c \) and F \(_{\text{cp}}^{(*)} \). As shown in Tables 2 and 3, the values of F \(_{\text{cp}}^{(*)} \) and F \(_{\text{cp}}^{**} \) were calculated, as well as the number of machines \( k \) that are actually in use. Finally, the mean value \( \bar{\text{Gap}} \) of Gap was determined.

According to the experimental results for \( p_{\text{max}} = 20 \) in Table 2, the mean value \( \bar{\text{Gap}} \) was quite large when \( m = 2 \). In all cases, the values of F \(_{\text{cp}}^{**} \) and Fc were rather small, with the maximum value being 1.71%. Thus, both Gap performed well. Compared with the cases of \( \lambda = 0.5 \) and \( \lambda = 0.8 \), the performance of \( \bar{\text{Gap}} \) was excellent at \( \lambda = 0.2 \).

The following things can be inferred from the experimental results for \( p_{\text{max}} = 80 \) in Table 3:
The number of machines in actual use was limited when \( \lambda \) was small (i.e., \( \lambda = 0.2 \)). When \( \lambda \) was large (i.e., \( \lambda = 0.8 \)), this number approached \( m \) because Algorithm \( A_2 \) is in line with the previous assumptions on cost and the number of machines.

(2) When the \( m \) remained constant, the mean \( Gap \) of Algorithm \( A_2 \) decreased with the increase of \( n \). Thus, the algorithm accuracy is positively correlated with the number of jobs.

(3) When the \( n \) remained constant, the \( Gap \) of Algorithm \( A_4 \) increased with \( \lambda \). This is because the value of \( \lambda \) is positively correlated with the number of machines and the change of sorting. In this case, the solution is close to the optimal plan.

(4) The mean \( Gap \) of each \( m \) decreased with the increase of \( p_{max} \).

Since the solution is close to the optimal one, Algorithm \( A_4 \) is proved feasible to solve the problem \( Q_m | T | C_{max} \leq D | C_{max} \).

6. CONCLUSIONS

This paper tackles the pre-emptive and non-pre-emptive versions of a UPMS problem considering the machine cost per unit time, under the constraint
that the total cost should not exceed the limit $\bar{U}$. Firstly, a machine selection method was developed based on level algorithm. Then, a heuristic algorithm $A_3$ was put forward for the pre-emptive problem, and proved to be optimal for problem $Q_m|\text{ptmn}, TC \leq \bar{U}|C_{\text{max}}$. Meanwhile, a heuristic algorithm $A_4$ was designed for the non-pre-emptive problem, and verified experimentally that is was suitable to solve the problem $Q_m|TC \leq \bar{U}|C_{\text{max}}$.

The future research will explore the same problem under more general assumptions (e.g. there is no strict correlation between the machine cost per unit time and the machine speed).

ACKNOWLEDGMENTS

Anhui soft science project (1607a0202031); key research project of Humanities and Social Sciences in Anhui Universities (SK2016A0526); Research project of innovation and development of Social Sciences in Anhui (2017CXF083); National Natural Science Foundation of China under grant nos 71471052.

REFERENCES