On Preemptive Scheduling on Uniform Machines to Minimize Mean Flow Time

Svetlana A. Kravchenko 1
United Institute of Informatics Problems,
Surganova St. 6, 220012 Minsk, Belarus
kravch@newman.bas-net.by

Frank Werner
Otto-von-Guericke-Universität, Fakultät für Mathematik,
39106 Magdeburg, Germany
Frank.Werner@Mathematik.Uni-Magdeburg.De

July 28, 2008

Abstract

In this paper we give a polynomial algorithm for problem $Q \mid r_j, p_j = p, \text{pmtt} \mid \sum C_j$ whose complexity status was open yet. The algorithm is based on a reduction of the scheduling problem to a linear program. The crucial condition for implementing the proposed reduction is the known order of job completion times.

Keywords: parallel uniform machines, linear programming, maximum flow, polynomial algorithm

1 Introduction

The problem considered can be stated as follows. There are $n$ independent jobs and $m$ parallel uniform machines. For each job $J_j, j = 1, \ldots, n$, we know its processing time

1Supported by the Alexander von Humboldt Foundation
Suppose that in some optimal schedule, there are two jobs $J_i$ and $J_j$ such that $0 < f_j(r_j) < f_i(r_i)$ and $p = f_j(J_j) > f_i(J_i)$, i.e. the order of the jobs in the optimal schedule is not preserved. Since $0 = f_j(r_j) < f_i(r_i)$ and $p = f_j(J_j) > f_i(J_i)$, there exists some time point $\xi \in [r_j, C_j]$ such that $f_j(\xi) = f_i(\xi)$, i.e. the job $J_j$ processed in $[\xi, C_j]$ is equal to the job $J_i$ processed in $[\xi, C_i]$. Now we can swap jobs $J_i$ and $J_j$ within the interval $[\xi, C_i]$.

In fact, for the proposed algorithm this simple statement plays a crucial role. In the following, we only deal with schedules from the described class. Let $s^*$ be an optimal substitution schedule. Then, $\sum_{j=1}^{n} C_j$. The described problem can be denoted as $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$.

Note that problem $P \mid r_j, \text{pmtn} \mid \sum C_j$ and, therefore, also problem $Q \mid r_j, \text{pmtn} \mid \sum C_j$ are unary NP-hard [1] whereas problem $Q \mid r_j, \text{pmtn} \mid C_{\text{max}}$ can be solved in $O(n \log n + mn)$ time [3]. Problem $R \mid r_j, \text{pmtn} \mid L_{\text{max}}$ can be effectively solved by reducing it to linear programming [5]. Problem $R \mid \text{pmtn} \mid \sum C_j$ is NP-hard in the strong sense, and problem $R \mid p_{ji} \in \{p_j, \infty\} \mid \sum C_j$ can be solved in $O(n^3)$ time [6].

The main result in this paper is a polynomial algorithm for problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$. This result is a generalization of the polynomial algorithm for problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$ from paper [2]. Throughout the paper, we suppose that the jobs are enumerated in such a way that $r_1 \leq \ldots \leq r_n$ holds. Furthermore, we assume that $r_1 = 0$.

## 2 A polynomial algorithm for problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$

In this section, we derive a polynomial algorithm for problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$.

**Statement 1** For problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$ an optimal schedule can be found in the class of schedules, for which

$$C_1 \leq \ldots \leq C_n$$

holds.

**Proof:** Let $T_{kq}(t)$ be the total amount of time that machine $M_q$ spends on job $J_k$ in the time period $[0, t]$. Then, $f_k(t) = \sum_{q=1}^{m} T_{kq}(t) \cdot s_q$ is the part of job $J_k$ processed in $[0, t]$. Suppose that in some optimal schedule, there are two jobs $J_i$ and $J_j$ such that $r_i < r_j$ and $C_i > C_j$ holds. Since $0 = f_j(r_j) < f_i(r_i)$ and $p = f_j(J_j) > f_i(J_i)$, there exists some time point $\xi \in [r_j, C_j]$ such that $f_j(\xi) = f_i(\xi)$, i.e. the job $J_j$ processed in $[\xi, C_j]$ is equal to the job $J_i$ processed in $[\xi, C_i]$. Now we can swap jobs $J_i$ and $J_j$ within the interval $[\xi, C_i]$. $\square$

In fact, for the proposed algorithm this simple statement plays a crucial role. In the following, we only deal with schedules from the described class. Let $s^*$ be an optimal
schedule for problem $Q | r_j, p_j = p, \text{pmtn} | \sum C_j$ with $C_1(s^*) \leq \ldots \leq C_n(s^*)$. Each $C_j(s^*)$ belongs to some interval $[r_i, r_{i+1}]$. However, if we know for each $C_j(s^*)$ the corresponding interval $[r_i, r_{i+1}]$ such that $C_j(s^*) \in [r_i, r_{i+1}]$, then an optimal schedule can be easily found using a reduction to a network flow problem. Thus, the main question is to know the interval $[r_i, r_{i+1}]$ for each $C_j(s^*)$ such that $C_j(s^*) \in [r_i, r_{i+1}]$. However, this difficulty can be avoided due to criterion $\sum C_j$. For any job $J_j$, let the time interval $[r_i, r_{i+1}]$ be such that $C_j \in [r_i, r_{i+1}]$. Taking into account that $r_1 = 0$, we obtain

$$C_j = (r_2 - r_1) + (r_3 - r_2) + \ldots + (r_i - r_{i-1}) + (C_j - r_i).$$

Due to this property, we introduce the completion time of job $J_j$ for each interval $[r_i, r_{i+1}]$. For each job $J_j$ with $j = 1, \ldots, n$ and for each interval $[r_i, r_{i+1}]$ with $i = 1, \ldots, n$, we define the value $C(J_j, r_i)$ such that $C(J_j, r_i) = C_j(s^*)$ if $C_j(s^*) \in [r_i, r_{i+1}]$, but if $C_j(s^*) \leq r_i$, then we set $C(J_j, r_i) = r_i$, and if $C_j(s^*) \geq r_{i+1}$, then we set $C(J_j, r_i) = r_{i+1}$. Thus, for job $J_j$ with $j = 1, \ldots, n$, we have

$$C(J_j, r_i) = \begin{cases} C_j(s^*) & \text{if } r_i < C_j(s^*) < r_{i+1} \\ r_i & \text{if } C_j(s^*) \leq r_i \\ r_{i+1} & \text{if } C_j(s^*) \geq r_{i+1} \end{cases} \quad (2.1)$$

So, for each $i = 1, \ldots, n$, the values

$$r_i \leq C(J_1, r_i) \leq \ldots \leq C(J_i, r_i) \leq C(J_{i+1}, r_i) = \ldots = C(J_n, r_i) = r_{i+1}$$

define a partition of the interval $[r_i, r_{i+1}]$. Here, we set $r_{n+1} = r_n + n \cdot \max_q \{p/s_q\}$, i.e. $r_{n+1}$ is a time point after which no job will be processed.

In turn, each interval $[C(J_j, r_i), C(J_{j+1}, r_i)]$ is completely defined by the jobs processed in it. Thus, we denote by $v(J_k, M_q, J_j, r_i)$ the part of job $J_k$ processed in the interval $[C(J_{j-1}, r_i), C(J_j, r_i)]$ by machine $M_q$, i.e. the total processing time of job $J_k$ in the interval $[C(J_{j-1}, r_i), C(J_j, r_i)]$ equals $\frac{v(J_k, M_q, J_j, r_i)}{s_k}$ and for any job $J_k$, equality

$$\sum_{q=1}^m \sum_{i=1}^n \sum_{j=1}^n v(J_k, M_q, J_j, r_i) = p$$

holds.

The values $C(J_j, r_i)$, where $j = 1, \ldots, n, \quad i = 1, \ldots, n$, and the values $v(J_k, M_q, J_j, r_i)$, where $i, j = 1, \ldots, n, \quad k = j, \ldots, i, \quad q = 1, \ldots, m$, define a feasible solution of the following linear program. For convenience, we introduce $C(J_0, r_i) = r_i$. 

3
Figure 1: Here $v(J_k, M_q, J_j, r_i)$ is the part of job $J_k$ processed within $[C(J_{j-1}, r_i), C(J_j, r_i)]$.

Minimize

$$
\sum_{i=1}^{n} \left( (C(J_1, r_i) - r_i) + \ldots + (C(J_n, r_i) - r_i) \right)
$$

subject to

$$
r_i = C(J_0, r_i) \leq C(J_1, r_i) \leq \ldots \leq C(J_{i+1}, r_i) = \ldots = C(J_n, r_i) = r_{i+1}, \quad i = 1, \ldots, n
$$

$$
\sum_{q=1}^{m} \frac{v(J_k, M_q, J_j, r_i)}{s_q} \leq C(J_j, r_i) - C(J_{j-1}, r_i),
$$

$$
i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n
$$

$$
\sum_{k=1}^{n} v(J_k, M_q, J_j, r_i) \leq C(J_j, r_i) - C(J_{j-1}, r_i),
$$

$$
i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad q = 1, \ldots, m
$$

$$
\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{q=1}^{m} v(J_k, M_q, J_j, r_i) = p, \quad k = 1, \ldots, n
$$

$$
v(J_k, M_q, J_j, r_i) = 0 \quad i = 1, \ldots, n-1, \quad j = i+1, \ldots, n,
$$

$$
q = 1, \ldots, m, \quad k = 1, \ldots, n
$$

$$
v(J_k, M_q, J_j, r_i) = 0 \quad i = 1, \ldots, n, \quad j = 1, \ldots, i,
$$

$$
q = 1, \ldots, m, \quad k = 1, \ldots, j-1
$$

$$
v(J_k, M_q, J_j, r_i) = 0 \quad i = 1, \ldots, n-1, \quad j = 1, \ldots, i,
$$

$$
q = 1, \ldots, m, \quad k = i + 1, \ldots, n
$$
\[ C(J_j, r_i) \geq 0 \quad i = 1, \ldots, n, \quad j = 1, \ldots, n \]  
\[ v(J_k, M_q, J_j, r_i) \geq 0 \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n, \quad q = 1, \ldots, m. \]  

The above formulation includes \( O(mn^3) \) variables and constraints, i.e. this problem can be polynomially solved.

Note that:

- Constraints (2.3) hold since in \([r_i, r_{i+1}]\) only jobs \( J_1, \ldots, J_i \) can be processed. Jobs \( J_{i+1}, \ldots, J_n \) are available only after \( r_{i+1} \).
- Constraints (2.4) hold since \( \sum_{q=1}^{m} \frac{v(J_k, M_q, J_j, r_i)}{s_q} \) is the total time of processing job \( J_k \) in the time interval \([C(J_{j-1}, r_i), C(J_j, r_i)]\), see Figure 1.
- Inequalities (2.5) are true since \( \sum_{k=1}^{n} \sum_{j=1}^{n} v(J_k, M_q, J_j, r_i) \) is the total time when machine \( M_q \) is busy in \([C(J_{j-1}, r_i), C(J_j, r_i)]\). Note that constraints (2.4) and (2.5) provide the possibility to schedule all parts \( v(J_k, M_q, J_j, r_i) \) within the time interval \([C(J_{j-1}, r_i), C(J_j, r_i)]\) in a feasible way.
- Constraints (2.6) hold since \( \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{q=1}^{m} v(J_k, M_q, J_j, r_i) \) is the total part of job \( J_k \) being processed.
- Constraints (2.7) hold since

\[ C(J_{i+1}, r_i) = C(J_{i+2}, r_i) = \ldots = C(J_n, r_i) = r_{i+1}, \]

i.e. there are no jobs processed in \([C(J_{i+1}, r_i), r_{i+1}]\).
- Constraints (2.8) hold since jobs \( J_1, \ldots, J_{j-1} \) cannot be processed after time point \( C(J_{j-1}, r_i) \), see Figure 1.
- Constraints (2.9) hold since jobs \( J_{i+1}, \ldots, J_n \) cannot be processed before time point \( r_{i+1} \), see Figure 1.
- Furthermore, for schedule \( s^* \) function (2.2) corresponds to the optimality criterion \( \sum_{j=1}^{n} C_j \), since

\[ C(J_j, r_i) - r_i = \begin{cases} C_j(s^*) - r_i & \text{if } r_i < C_j(s^*) < r_{i+1} \\ 0 & \text{if } C_j(s^*) \leq r_i \\ r_{i+1} - r_i & \text{if } C_j(s^*) \geq r_{i+1} \end{cases} \]

and therefore, \( \sum_{i=1}^{n} (C(J_j, r_i) - r_i) = C_j(s^*) \).

Thus, \( \sum_{i=1}^{n} \left( (C(J_1, r_i) - r_i) + \ldots + (C(J_n, r_i) - r_i) \right) = \sum_{j=1}^{n} C_j(s^*). \)
Summarizing, we have proven

**Theorem 1** For any optimal schedule \(s^*\) of problem \(Q\) \(| r_j, p_j = p, pmtn | \sum C_j\), there is a corresponding feasible solution of (2.2)-(2.11) such that \(C_1(s^*) \leq \ldots \leq C_n(s^*)\) and

\[
\sum_{j=1}^{n} C_j(s^*) = \sum_{i=1}^{n} \left( (C(J_1, r_i) - r_i) + \ldots + (C(J_n, r_i) - r_i) \right)
\]

hold.

The next theorem shows that conversely any feasible solution of (2.2)-(2.11) also provides a feasible schedule.

**Theorem 2** Any feasible solution of problem (2.2)-(2.11) provides a feasible schedule \(s^*\) for the scheduling problem \(Q\) \(| r_j, p_j = p, pmtn | \sum C_j\) such that

\[
\sum_{j=1}^{n} C_j(s^*) = \sum_{i=1}^{n} \left( (C(J_1, r_i) - r_i) + \ldots + (C(J_n, r_i) - r_i) \right)
\]

holds.

**Proof:** Any feasible solution of (2.2)-(2.11) provides some values \(C(J_j, r_i)\) and \(v(J_k, M_q, J_j, r_i)\), and hence it provides some feasible schedule, which can be reconstructed by processing all parts \(v(J_k, M_q, J_j, r_i)\) in the intervals \([C(J_{j-1}, r_i), C(J_j, r_i)]\). If the values \(C(J_j, r_i)\) obtained are such that for any job \(J_j\), there is an index \(e\) such that \(C(J_j, r_i) = r_{i+1}\) holds for any \(r_i < r_e\) and \(C(J_j, r_i) = r_i\) holds for any \(r_i > r_e\), then for the schedule \(s^*\) defined by \(C(J_j, r_i)\) and \(v(J_k, M_q, J_j, r_i)\) equality

\[
\sum_{j=1}^{n} C_j(s^*) = \sum_{i=1}^{n} \left( (C(J_1, r_i) - r_i) + \ldots + (C(J_n, r_i) - r_i) \right)
\]

holds. Thus, in this case Theorem 2 is true.

Now suppose that for some job \(J_j\), there does not exist such an index \(e\). In this case, one can find two intervals \([r_k, r_{k+1}]\) and \([r_h, r_{h+1}]\) such that \(r_k \leq C(J_j, r_k) < r_{k+1} \leq r_h < C(J_j, r_h)\) holds. Transform the schedule \(s^*\) in the following way. Take the largest value of \(\delta\) such that in \([C(J_j, r_k), C(J_j, r_k) + \delta]\) and in \([C(J_j, r_h), C(J_j, r_h) - \delta]\), each machine is either idle or processes exactly one job. Now we swap \(J_j\) from the interval \([C(J_j, r_h), C(J_j, r_h) - \delta]\) and \(J_l\) (if any) from the interval \([C(J_j, r_k), C(J_j, r_k) + \delta]\) on the same machine, say \(M_z\) (see Figure 2). Since in \([r_k, r_{k+1}]\) inequality \(C(J_l, r_k) > C(J_j, r_k)\) holds, it follows that inequality \(r_j \leq r_l\) holds. This implies \(C(J_l, r_h) \geq C(J_j, r_h)\) and \(\sum C_j\) does not change.

Now, if it happens that \(J_l\) is processed in \([C(J_j, r_h), C(J_j, r_h) - \delta]\) by some other machine, say \(M_g \neq M_z\), then we swap job \(J_l\) from \([C(J_j, r_h), C(J_j, r_h) - \delta]\) and \(J_f\) (if any) from
Figure 2: Swap of \( J_j \) from \([C(J_j, r_k), C(J_j, r_h) - \delta]\) and \( J_l \) from \([C(J_l, r_k), C(J_l, r_k) + \delta]\).

\([C(J_j, r_k), C(J_j, r_k) + \delta]\) on machine \( M_p \). We will continue with this swapping as long as the schedule remains infeasible. Note that the value of (2.2) does not increase. \( \square \)

**Example.** Let us consider the instance from [2]. The data are the following: \( m = 3 \), \( p = 10 \), \( r_1 = \ldots = r_5 = 0 \), \( r_6 = r_7 = 11 \), \( r_8 = r_9 = 18 \), \( r_{10} = 22 \), \( r_{11} = 27 \). Let the speeds be \( s_1 = 1 \), \( s_2 = 1 \), and \( s_3 = 2 \). Using CPLEX \(^2\), we solve the corresponding LP problem. We obtain an optimal solution with the objective function value 196.54296875 and with the following non-zero variables:

\[
\begin{align*}
C(J_0, r_6) &= 11, & C(J_0, r_7) &= 11, & C(J_0, r_8) &= 18, & C(J_0, r_9) &= 18, \\
C(J_0, r_{10}) &= 22, & C(J_0, r_{11}) &= 27, & C(J_0, r_{12}) &= 11, & C(J_0, r_{13}) &= 11, & C(J_0, r_{14}) &= 11, & C(J_0, r_{15}) &= 11, \\
C(J_0, r_{16}) &= 11, & C(J_0, r_{17}) &= 11, & C(J_0, r_{18}) &= 11, & C(J_0, r_{19}) &= 11, & C(J_0, r_{20}) &= 11, \\
C(J_0, r_{21}) &= 11, & C(J_0, r_{22}) &= 11, & C(J_0, r_{23}) &= 11, & C(J_0, r_{24}) &= 11, & C(J_0, r_{25}) &= 11, \\
C(J_0, r_{26}) &= 11, & C(J_0, r_{27}) &= 11, & C(J_0, r_{28}) &= 11, & C(J_0, r_{29}) &= 11, & C(J_0, r_{30}) &= 11. \\
\end{align*}
\]

\(^2\)CPLEX is a trademark of ILOG, Inc. http://www.ilog.com/products/cplex/
The solution is defined by the five non-zero intervals $[v_1, M_3, J_1, r_5] = 10$, $v(J_2, M_1, J_1, r_5) = 3.5$, $v(J_3, M_3, J_1, r_5) = 1.5$, $v(J_4, M_1, J_2, r_5) = 2.5$, $v(J_5, M_1, J_3, r_5) = 3$, $v(J_6, M_3, J_4, r_5) = 0.5$, $v(J_7, M_1, J_5, r_7) = 3.25$, $v(J_8, M_3, J_3, r_7) = 0.75$, $v(J_9, M_3, J_7, r_9) = 1.3125$, $v(J_{10}, M_3, J_8, r_{10}) = 3.3125$, $v(J_{11}, M_3, J_9, r_{10}) = 0.515625$, $v(J_{12}, M_3, J_9, r_{11}) = 3.828125$, $v(J_{13}, M_3, J_9, r_{11}) = 8.0859375$.

Figure 3: The optimal schedule corresponding to the CPLEX solution. Here (b1) corresponds to $v(J_1, M_3, J_1, r_5) = 10$, (b2) corresponds to $v(J_2, M_1, J_1, r_5) = 3.5$, (b3) corresponds to $v(J_3, M_1, J_2, r_5) = 2.5$, (b4) corresponds to $v(J_4, M_1, J_3, r_5) = 3$, (b5) corresponds to $v(J_4, M_3, J_4, r_5) = 1$, (b6) corresponds to $v(J_5, M_1, J_5, r_7) = 3.25$, (b7) corresponds to $v(J_7, M_1, J_6, r_7) = 3.375$, (b8) corresponds to $v(J_8, M_3, J_7, r_7) = 0.75$, (b9) corresponds to $v(J_8, M_1, J_7, r_9) = 1.3125$, (b10) corresponds to $v(J_8, M_3, J_7, r_9) = 5.375$, (b11) corresponds to $v(J_9, M_3, J_8, r_{10}) = 1.65625$, (b12) corresponds to $v(J_9, M_3, J_8, r_{10}) = 0.515625$, (b13) corresponds to $v(J_{10}, M_3, J_9, r_{10}) = 6.171875$, (b14) corresponds to $v(J_{10}, M_3, J_9, r_{11}) = 3.828125$, (b15) corresponds to $v(J_{11}, M_3, J_{11}, r_{11}) = 8.0859375$.

The solution is defined by the five non-zero intervals $[r_5, r_6] = [0, 11]$, $[r_7, r_8] = [11, 18]$, $[r_9, r_{10}] = [18, 22]$, $[r_{10}, r_{11}] = [22, 27]$, and $[r_{11}, r_{12}] = [27, 137]$. For $[r_5, r_6]$, we get $C(J_1, r_5) = 5$, $C(J_2, r_5) = 7.5$, $C(J_3, r_5) = 10.5$, $C(J_4, r_5) = 11$. For $[r_7, r_8] = [11, 18]$, we get $C(J_4, r_7) = 11$, $C(J_5, r_7) = 14.25$, $C(J_6, r_7) = 17.625$, $C(J_7, r_7) = 18$. For $[r_9, r_{10}] = [18, 22]$, we have $C(J_6, r_9) = 18$, $C(J_7, r_9) = 19.3125$, $C(J_8, r_9) = 22$, $C(J_9, r_9) = 22$. For $[r_{10}, r_{11}] = [22, 27]$, we have $C(J_7, r_{10}) = 22$, $C(J_8, r_{10}) = 23.65625$, $C(J_9, r_{10}) = 23.9140625$, $C(J_{10}, r_{10}) = 27$. For $[r_{11}, r_{12}] = [27, 137]$, we have $C(J_8, r_{11}) = 27$, $C(J_{11}, r_{11}) = 28.9140625$, $C(J_{10}, r_{11}) = 28.9140625$, $C(J_{11}, r_{11}) = 32.95703125$.

In Figure 4, we leave only blocks corresponding to such values $v(J_k, M_q, J_j, r_i)$ that define the $C_j$ values for each interval $[r_i, r_{i+1}]$.

From Figure 4, one can see that (2.1) holds for each job except job $J_5$, since $r_{10} < C(J_9, r_{10}) < r_{11}$ and $r_{11} < C(J_9, r_{11}) < r_{12}$. Therefore, for all jobs except job $J_5$, we have $C_1 = C(J_1, r_5) = 5$, $C_2 = C(J_2, r_5) = 7.5$, $C_3 = C(J_3, r_5) = 10.5$, $C_4 = C(J_4, r_5) = 11$, $C_5 = C(J_5, r_7) = 14.25$, $C_6 = C(J_6, r_7) = 17.625$, $C_7 = C(J_7, r_9) = 19.3125$, $C_8 = 8$. 
For job calculation the value $C(J)$, machine $M$ obtains an infeasible schedule, see Figure 5, since job $J$ in the proof of Theorem 2. In the first step, we swap a part of job $J$ with length $\delta$ and a part of job $J_9$ with length $\delta$ on machine $M_3$, see Figure 5. After the first step, we obtain an infeasible schedule, see Figure 5, since job $J_{10}$ is processed on machine $M_3$ and machine $M_1$ at the same time. Nevertheless, the value of (2.2) is not changed. In the

![Figure 4](image)

Figure 4: Here each block corresponds to such a value $v(J_k, M, J_j, r_i)$ that defines $C(J, r_i)$.  

$$C(J_8, r_{10}) = 23.65625, C_{10} = C(J_{10}, r_{11}) = 28.9140625, C_{11} = C(J_{11}, r_{11}) = 32.95703125.$$

For job $J_9$, see Figure 3, we have $C_9 = C(J_9, r_{11}) = 28.9140625$, but from (2.2) we calculate the value

$$C_9 = \sum_{i=1}^{11} (C(J_9, r_i) - r_i) = (C(J_9, r_1) - r_1) + (C(J_9, r_2) - r_2) + (C(J_9, r_3) - r_3) + (C(J_9, r_4) - r_4) + (C(J_9, r_5) - r_5) + (C(J_9, r_6) - r_6) + (C(J_9, r_7) - r_7) + (C(J_9, r_8) - r_8) + (C(J_9, r_9) - r_9) + (C(J_9, r_{10}) - r_{10}) + (C(J_9, r_{11}) - r_{11})$$

$$= (0 - 0) + (0 - 0) + (0 - 0) + (0 - 0) + (11 - 0) + (11 - 11) + (18 - 11) + (18 - 18) + (22 - 18) + (23.9140625 - 22) + (28.9140625 - 27)$$

$$= 25.828125.$$

To get the schedule with $C_9 = 25.828125$, one can apply the transformation described in the proof of Theorem 2. In the first step, we swap a part of job $J_9$ with length $\delta$ and a part of job $J_{10}$ with length $\delta$ on machine $M_3$, see Figure 5. After the first step, we obtain an infeasible schedule, see Figure 5, since job $J_{10}$ is processed on machine $M_3$ and machine $M_1$ at the same time. Nevertheless, the value of (2.2) is not changed. In the
next step of the transformation, we swap a part of job \( J_{10} \) with length \( \delta \) with the idle interval of length \( \delta \) on machine \( M_1 \), see Figure 6. The obtained schedule is feasible and the value of (2.2) is not changed, but now we have

\[
C_9 = C(J_9, r_{10}) + \frac{1}{2} \cdot v(J_9, M_3, J_9, r_{11}) = 23.9140625 + \frac{1}{2} \cdot 3.828125 = 25.828125
\]

Thus, one can see that we do not need to apply the transformation. Since as a result of solving (2.2)-(2.11) we obtain the values \( C_1, \ldots, C_n \), we can reconstruct the optimal schedule by solving the following network flow problem, see [4].

Suppose that all points \( r_1, \ldots, r_n, C_1, \ldots, C_n \) are enumerated in non-decreasing order of their values, say \( t_1 \leq \ldots \leq t_k \). We construct a network which has the following vertices:

- a source \( u \) and a sink \( w \),
- job-vertices \( \{J_j \mid j = 1, \ldots, n\} \),
- machine-time-interval-vertices \( \{(t_i, t_{i+1}, M_e) \mid i = 1, \ldots, k-1, \ e = 1, \ldots, m\} \)

The arcs in the network defined are the following:

- for each job-vertex \( J_j \), there is an arc \((u, J_j)\) with the capacity \( p \), and there is an arc \((J_j, (t_i, t_{i+1}, M_e))\) if \( r_j \leq t_i \) and \( t_{i+1} \leq C_j \) hold,
- for each vertex \( (t_i, t_{i+1}, M_e) \), there is an arc \(((t_i, t_{i+1}, M_e), w)\) with the capacity \((t_{i+1} - t_i) \cdot s_e\).

Any maximal flow will reconstruct an optimal schedule.

Thus, to solve problem \( Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j \), one has to do the following:
1. Solve the corresponding linear program (2.2)-(2.11), and calculate the values $C_j = \sum_{i=1}^{a}(C(J, r_i) - r_i)$ for each job $J_j$.

2. Reconstruct an optimal schedule by solving the corresponding network flow problem.

### 3 Concluding remarks

In this paper, we considered problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$, whose complexity status was open yet. We presented a polynomial algorithm which is based on the solution of a linear program and a network flow problem. For further research, the most interesting question is whether it is possible to simplify the proposed linear programming formulation in the same way as it was made in [1] for problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum C_j$. In particular, is it possible to find such special properties of an optimal schedule which permit to reduce the size of the linear programming formulation?

### References


