# Minimizing a Separable Convex Function on Parallel Machines with Preemptions

Svetlana A. Kravchenko
United Institute of Informatics Problems,
Surganova St. 6, 220012 Minsk, Belarus

Frank Werner *
Otto-von-Guericke-Universität, Fakultät für Mathematik,
39106 Magdeburg, Germany

August 24, 2009

**Abstract**

The basic scheduling problem we are dealing with in this paper is the following one. A set of jobs has to be scheduled on a set of parallel uniform machines. Each machine can handle at most one job at a time. All jobs have the same execution requirement. Each machine has a known speed. The processing of any job may be interrupted arbitrarily often and resumed later on any machine. The goal is to find a schedule that minimizes $\sum f_j$, where $f_j$ are convex non-decreasing functions such that $f_i - f_j$ are all monotonic functions. Thus, we consider problem $Q \mid p_j = p, \text{ pmtn} \mid \sum f_j$. We show that problem $Q \mid p_j = p, \text{ pmtn} \mid \sum f_j$ is equivalent to the problem of minimizing a convex-separable function under linear constraints. We use this representation to solve problems $Q \mid p_j = p, \text{ pmtn} \mid \sum T_j$ and $Q \mid p_j = p, \text{ pmtn} \mid \sum w_j C_j$ in polynomial time. Note that both problems $Q \mid p_j = p, \text{ pmtn} \mid \sum w_j C_j$ and $Q \mid p_j = p, \text{ pmtn} \mid \sum T_j$ had an open complexity status. Recently, Tian et al. [10] proposed a polynomial algorithm for problem $1 \mid r_j, p_j = p, \text{ pmtn} \mid \sum T_j$. We show that both the problem $P \mid \text{ pmtn} \mid \sum T_j$ of minimizing total tardiness on a set of parallel machines with allowed preemptions and the problem $P \mid r_j, p_j = p, \text{ pmtn} \mid \sum T_j$ of minimizing total tardiness on a set of parallel

*Corresponding author. Tel.:+0391 6712025; fax:+0391 6711171.
*E-mail address:* frank.werner@mathematik.uni-magdeburg.de (F.Werner).

machines with release dates, equal processing times and allowed preemptions are NP-hard.

**Keywords:** parallel uniform machines, linear programming, polynomial algorithm, NP-hardness

# 1 Introduction

The problem considered can be stated as follows. There are $n$ independent jobs and $m$ parallel uniform machines. For each job $J_j$, $j = 1, \ldots, n$, there is given its processing time $p_j = p$. Each machine $M_q$, $q = 1, \ldots, m$, has some speed $s_q$, i.e., the execution of job $J_j$ on machine $M_q$ requires $p/s_q$ time units. Any machine can process any job but only one job at a time. Furthermore, a job can be processed only on one machine at a time. Preemptions of processing are allowed, i.e., the processing of any job may be interrupted at any time and resumed later, possibly on a different machine. We assume that all numerical data are integers. For a schedule $s$, let $f_j(C_j(s))$, where $j = 1, \ldots, n$, $j = 1, \ldots, n$, denote convex non-decreasing functions such that $f_i - f_j$ are all monotonic functions, where $C_j(s)$ denotes the time at which the processing of job $J_j$ is completed. If no ambiguity arises, we drop the reference to schedule $s$ and write $C_j$. The problem is to schedule all jobs so as to minimize the optimality criterion $\sum_{j=1}^{n} f_j$. The described problem can be denoted as $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$. We show that this problem is equivalent to the problem of minimizing a convex-separable function subject to linear constraints. We use this representation to develop polynomial algorithms for problem $Q \mid p_j = p,\ \text{pmtn} \mid \sum T_j$ and for problem $Q \mid p_j = p,\ \text{pmtn} \mid \sum w_j C_j$, whose complexity status was open.

Note that problem $P \mid r_j,\ \text{pmtn} \mid \sum C_j$ and therefore, problem $P \mid r_j,\ \text{pmtn} \mid \sum T_j$ are unary NP-hard [1]. Recall that $P$ in the notation of the problem means that all machines have identical speeds, the integer $r_j \geq 0$ denotes the release date and $T_j(s) = \max\{0, C_j(s) - d_j\}$ denotes the tardiness of job $J_j$ with an integer due date $d_j \geq 0$.

Problem $1 \mid r_j, p_j = p,\ \text{pmtn} \mid \sum T_j$ can be solved in $O(n^2)$ time [10]. In [3], it has been shown that problem $1 \mid\ \text{pmtn} \mid T_j$ is NP-hard in the ordinary sense whereas in [7], a pseudopolynomial algorithm has been proposed for problem $1 \mid\ \text{pmtn} \mid T_j$. Both results are valid for the problem with allowed preemptions as far as for the non-preemptive case. As it was noted in [3], preemptions cannot reduce the total tardiness on one machine but for parallel machines, it is easy to see that preemptions can reduce the total tardiness. In this paper, it is shown that both problems $P \mid\ \text{pmtn} \mid \sum T_j$ and $P \mid r_j, p_j = p,\ \text{pmtn} \mid \sum T_j$ are NP-hard under a binary encoding. Note that the complexity status of both problems mentioned above was open.

The paper is organized as follows. In Section 2, we describe a polynomial reduction of problem $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$ to the problem of minimizing a convex-separable

function subject to linear constraints. In Section 3, we develop polynomial algorithms for problems $Q \mid p_j = p,\ \text{pmtn} \mid \sum T_j$ and $Q \mid p_j = p,\ \text{pmtn} \mid \sum w_j C_j$. In Section 4, we prove the NP-hardness of problem $P \mid \text{pmtn} \mid \sum T_j$. In Section 5, we prove the NP-hardness of problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum T_j$. Section 6 presents some concluding remarks.

# 2 Problem $Q \mid p_j = p, \textbf{pmtn} \mid \sum f_j$

In this section, we show that problem $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$, where $f_j$ are convex non-decreasing functions such that $f_i - f_j$ are all monotonic functions, can be reduced to the problem of minimizing a convex-separable function subject to linear constraints. The idea of the reduction is taken from [5], where a polynomial algorithm for problem $Q \mid r_j, p_j = p,\ \text{pmtn} \mid \sum C_j$ was derived.

Suppose that all the jobs are numbered in such a way that for any pair of jobs $J_i$ and $J_j$, $f_j - f_i$ is a non-decreasing function for $i < j$.

**Statement 1** *For problem $Q \mid p_j = p, pmtn \mid \sum f_j$, where for any pair of jobs $J_i$ and $J_j$ with $i < j$, it follows that $f_j - f_i$ is a non-decreasing function and all $f_j$ are convex non-decreasing functions, an optimal schedule can be found in the class of schedules for which*

$$C_1 \leq C_2 \leq \cdots \leq C_n$$

*holds.*

**Proof:** Suppose that in some optimal schedule, there are two jobs $J_i$ and $J_j$ such that $i < j$, i.e., $f_i - f_j$ is non-decreasing and $C_i > C_j$ holds. Swap jobs $J_i$ and $J_j$. The objective function value will increase by

$$-f_j(C_j) - f_i(C_i) + f_i(C_j) + f_j(C_i) = (f_i(C_j) - f_j(C_j)) - (f_i(C_i) - f_j(C_i)).$$

Since $f_i - f_j$ is non-decreasing and $C_i > C_j$, we obtain that

$$(f_i(C_j) - f_j(C_j)) - (f_i(C_i) - f_j(C_i)) \leq 0$$

holds, i.e., the objective function value will not increase. Therefore, all jobs can be scheduled in such a way that for any two jobs $J_i$ and $J_j$ with $i < j$, inequality $C_i \leq C_j$ holds. □

Thus, we will look for an optimal schedule among the class of schedules for which $C_1 \leq \cdots \leq C_n$ holds. Let $b_1 < \ldots < b_z$ be the set of break points of functions $f_1, \ldots, f_n$. Furthermore, we set $b_0 = 0$ and suppose that $b_z < b_{z+1} = n \cdot p$, i.e., $[b_0, b_{n+1}]$ is the time interval within which all jobs have to be processed. Note that the set of all points $\{b_0, \ldots, b_{z+1}\}$ together with the set of all completion times $\{C_1, \ldots, C_n\}$ define a partition of the time interval. If we know both sets, then an optimal schedule can be easily found using a reduction to a network flow problem, see [6].

Now, for each job $j \in \{1, \ldots, n\}$, we define the 'completion' time of job $J_j$ for each interval $[b_i, b_{i+1}]$ : For each job $J_j$ with $j = 1, \ldots, n$ and for each interval $[b_i, b_{i+1}]$ with $i = 0, \ldots, z$, we define the value $C_j^i$ such that, if some part of job $J_j$ is scheduled in $[b_i, b_{i+1}]$, then this part has to be scheduled in $[b_i, C_j^i]$, i.e., there is no part of job $J_j$ processed within the interval $[C_j^i, b_{i+1}]$. Moreover, we set $C_1^i \leq C_2^i \leq \ldots \leq C_n^i$.

Thus, if we know a feasible schedule $s$, we can set

$$C_j^i = \begin{cases} C_j(s) & \text{if} \quad b_i < C_j(s) < b_{i+1} \\ b_i & \text{if} \quad C_j(s) \leq b_i \\ b_{i+1} & \text{if} \quad C_j(s) \geq b_{i+1} \end{cases} \tag{2.1}$$

for each $j = 1, \ldots, n$, and $i = 0, \ldots, z$. Using equalities (2.1), we can calculate the value $\sum f_j$ by the formula

$$f_j(C_j) = f_j(b_0) + (f_j(C_j^0) - f_j(b_0)) + \ldots + (f_j(C_j^z) - f_j(b_z)).$$

Indeed, let $C_j(s) \in [b_i, b_{i+1}]$, then

$$f_j(C_j) = f_j(b_0) + (f_j(C_j^0) - f_j(b_0)) + \ldots + (f_j(C_j^{i-1}) - f_j(b_{i-1})) +$$

$$(f_j(C_j^i) - f_j(b_i)) + (f_j(C_j^{i+1}) - f_j(b_{i+1})) + \ldots + (f_j(C_j^z) - f_j(b_z)).$$

Since $C_j^0 = b_1, \ldots, C_j^{i-1} = b_i$ and $C_j^{i+1} = b_i, \ldots, C_j^z = b_{z-1}$, we obtain

$$f_j(C_j) = f_j(b_0) + (f_j(b_1) - f_j(b_0)) + \ldots + (f_j(b_i) - f_j(b_{i-1})) +$$

$$(f_j(C_j(s)) - f_j(b_i)) + (f_j(b_{i+1}) - f_j(b_{i+1})) + \ldots + (f_j(b_z) - f_j(b_z)) = f_j(C_j(s)).$$

So, for each $i = 0, \ldots, n$, the values

$$b_i = C_0^i \leq C_1^i \leq \cdots \leq C_n^i \leq C_{n+1}^i = b_{i+1}$$

define a partition of the interval $[b_i, b_{i+1}]$.

In turn, each interval $[C_k^i, C_{k+1}^i]$ is completely defined by the jobs processed in it. Thus, we denote by $v_j^q([C_k^i, C_{k+1}^i])$ the part (amount) of job $J_j$ processed in the interval $[C_k^i, C_{k+1}^i]$ on machine $M_q$, see Figure 1, i.e., the total processing time of job $J_j$ on machine $M_q$ in the interval $[C_k^i, C_{k+1}^i]$ equals $\frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q}$ and for any job $J_j$, equality

$$\sum_{q=1}^{m} \sum_{k=0}^{n} \sum_{i=0}^{z} v_j^q([C_k^i, C_{k+1}^i]) = p$$

holds.

The values $C_k^i$, where $k = 0, \ldots, n+1, \quad i = 0, \ldots, z$, and the values $v_j^q([C_k^i, C_{k+1}^i])$, where $j = 1, \ldots, n, \quad i = 0, \ldots, z, \; k = 0, \ldots, n+1, \quad q = 1, \ldots, m$, define a feasible solution of the following minimization problem.

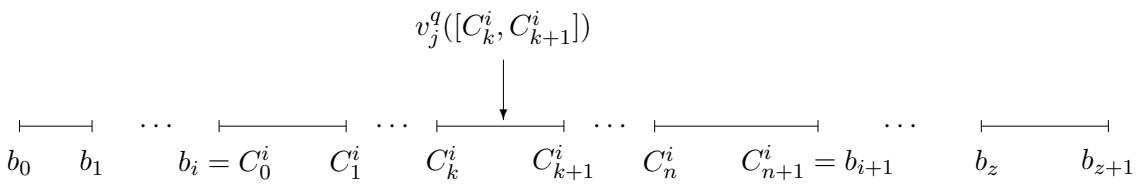


Figure 1: A partition of the interval $[b_i, b_{i+1}]$.

Minimize

$$\sum_{k=1}^{n}\Big(f_k(b_0) + \sum_{i=0}^{z}(f_k(C_k^i) - f_k(b_i))\Big) \tag{2.2}$$

subject to

$$b_i = C_0^i \le C_1^i \le \cdots \le C_n^i \le C_{n+1}^i = b_{i+1}, \qquad i = 0, \ldots, z \tag{2.3}$$

$$\sum_{q=1}^{m} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \le C_{k+1}^i - C_k^i, \qquad i = 0, \ldots, z, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n \tag{2.4}$$

$$\sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \le C_{k+1}^i - C_k^i, \qquad i = 0, \ldots, z, \quad q = 1, \ldots, m, \quad k = 1, \ldots, n \tag{2.5}$$

$$\sum_{i=0}^{z}\sum_{k=0}^{n}\sum_{q=1}^{m} v_j^q([C_k^i, C_{k+1}^i]) = p, \qquad j = 1, \ldots, n \tag{2.6}$$

$$C_k^i \ge 0, \qquad i = 0, \ldots, z, \quad k = 0, \ldots, n \tag{2.7}$$

$$v_j^q([C_k^i, C_{k+1}^i]) \ge 0, \qquad i = 0, \ldots, z, \quad j = 1, \ldots, n, \quad k = 0, \ldots, n, \quad q = 1, \ldots, m. \tag{2.8}$$

The above formulation includes $O(mn^2z)$ variables and constraints.

**Theorem 1** *For any feasible schedule $s$ of problem $Q \mid p_j = p, pmtn \mid \sum f_j$, where the completion times $C_1(s) \le \cdots \le C_n(s)$ are in the same order as the corresponding function values $f_1(C_1(s)) \le \cdots \le f_n(C_n(s))$, there exists a corresponding feasible solution of problem (2.3)-(2.8) such that*

$$\sum_{j=1}^{n} f_j(C_j(s)) = \sum_{k=1}^{n}\Big(f_k(b_0) + \sum_{i=0}^{z}(f_k(C_k^i) - f_k(b_i))\Big)$$

*holds.*

**Proof:** Let $s$ be a feasible schedule of problem $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$ with $f_1 \leq \cdots \leq f_n$ and $C_1 \leq \cdots \leq C_n$. Using schedule $s$ and equalities (2.1), we obtain the values of all variables $C_k^i$ and $v_j^q([C_k^i, C_{k+1}^i])$.

First, condition (2.3) holds since for any two jobs $x$ and $y$ such that $f_x \leq f_y$ with $C_x \in [b_a, b_{a+1}]$ and $C_y \in [b_e, b_{e+1}]$, we obtain

$$C_x^0 = b_1, \cdots, \ C_x^{a-1} = b_a, \ C_x^a \in [b_a, b_{a+1}], \ C_x^{a+1} = b_{a+1}, \cdots, \ C_x^z = b_z,$$

and

$$C_y^0 = b_1, \cdots, \ C_y^{e-1} = b_e, \ C_y^e \in [b_e, b_{e+1}], \ C_y^{e+1} = b_{e+1}, \cdots, \ C_y^z = b_z.$$

Since $a \leq e$, we have $C_x^i \leq C_y^i$ for any $i = 0, \ldots, z$.

Inequalities (2.4) hold since all parts $v_j^1([C_k^i, C_{k+1}^i]), \cdots, v_j^m([C_k^i, C_{k+1}^i])$ of job $j$ have to be scheduled in the interval $[C_k^i, C_{k+1}^i]$ on different machines without overlapping.

Inequalities (2.5) hold since the parts $v_1^q([C_k^i, C_{k+1}^i]), \cdots, v_n^q([C_k^i, C_{k+1}^i])$ of all jobs have to be scheduled in the interval $[C_k^i, C_{k+1}^i]$ on machine $M_q$ without overlapping.

For each job $j$, the sum of all values $v_j^q([C_k^i, C_{k+1}^i])$ has to be equal to $p$ since job $j$ has to be processed completely. Therefore, equalities (2.6) must hold.

Furthermore, function (2.2) corresponds to the optimality criterion $\sum_{j=1}^n f_j(s)$ because

$$f_j(C_j^i) - f_j(b_i) = \begin{cases} f_j(C_j(s)) - f_j(b_i) & \text{if} \quad b_i < C_j(s) < b_{i+1} \\ 0 & \text{if} \quad C_j(s) \leq b_i \\ f_j(b_{i+1}) - f_j(b_i) & \text{if} \quad C_j(s) \geq b_{i+1} \end{cases}$$

and therefore, $f_j(C_j(s)) = f_j(b_0) + \sum_{i=0}^z (f_j(C_j^i) - f_j(b_i))$. □

Now we prove

**Theorem 2** *Any feasible solution of problem (2.2)-(2.8) provides a feasible schedule $s$ for the scheduling problem $Q \mid p_j = p, pmtn \mid \sum f_j$ such that*

$$\sum_{j=1}^n f_j(C_j(s)) = \sum_{k=1}^n \Big( f_k(b_0) + \sum_{i=0}^z (f_k(C_k^i) - f_k(b_i)) \Big)$$

*holds.*

**Proof:** Let $v_j^q([C_k^i, C_{k+1}^i])$ and $C_k^i$, where $k = 0, \ldots, n$, $i = 0, \ldots, z$, $j = 1, \ldots, n$, and $q = 1, \ldots, m$, be a feasible solution of problem (2.2)-(2.8). For any interval $[C_k^i, C_{k+1}^i]$, it is possible to construct a feasible schedule with the length

$$\max \left\{ \max_{1 \leq j \leq n} \sum_{q=1}^m \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q}, \max_{1 \leq q \leq m} \sum_{j=1}^n \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \right\},$$

with a finite number of preemptions, see [9]. Taking into account inequalities (2.4) and (2.5), we obtain

$$\max\left\{\max_{1\le j\le n}\sum_{q=1}^{m}\frac{v_j^q([C_k^i,C_{k+1}^i])}{s_q},\max_{1\le q\le m}\sum_{j=1}^{n}\frac{v_j^q([C_k^i,C_{k+1}^i])}{s_q}\right\}\le C_{k+1}^i-C_k^i.$$

Thus, for any interval $[C_{k+1}^i,C_k^i]$, one can construct a feasible schedule. Therefore, for any feasible solution of problem (2.2)–(2.8), one can construct a feasible schedule $\tilde{s}$.

Now, if for any $k=0,\dots,n$, the values $C_k^i$ satisfy equalities (2.1), then $f_k(C_k(\tilde{s}))=f_k(b_0)+\sum_{i=1}^{n}(f_i(C_k^i)-f_i(b_i))$ and therefore, $\sum_{k=1}^{n}\Big(f_k(b_0)+\sum_{i=0}^{z}(f_k(C_k^i)-f_k(b_i))\Big)=\sum_{k=1}^{n}f_k(C_k(\tilde{s}))$ holds. Thus, $\tilde{s}$ is an optimal schedule.

Now, suppose that for some job $J_k$, the values $C_k^i$ do not satisfy equalities (2.1). Then there exist two intervals $[b_g,b_{g+1}]$ and $[b_h,b_{h+1}]$ such that $b_g\le C_k^g<b_{g+1}\le b_h<C_k^h\le b_{h+1}$ holds. Transform the schedule $\tilde{s}$ in the following way. Take the largest value of $\delta$ such that in the intervals $[C_k^g,C_k^g+\delta]$ and $[C_k^h-\delta,C_k^h]$, each machine is either idle or processes exactly one job. Now, we swap $J_k$ from the interval $[C_k^h-\delta,C_k^h]$ and $J_l$ (if any) from the interval $[C_k^g,C_k^g+\delta]$ on the same machine, say $M_z$ (see Figure 2). Set $C_k^g=C_k^g+\delta$ and $C_k^h=C_k^h-\delta$. Since in the interval $[b_g,b_{g+1}]$ inequality $C_g^l>C_k^j$ holds, it follows that inequality $b_g\le b_l$ holds and therefore, $C_l^h\ge C_l^h$ holds. This implies that after the described swapping the completion time of job $J_l$ is not changed.

Consider the $f_k$ value before and after the swapping in $[b_g,b_{h+1}]$. Before the swapping, it was $f_k(C_k^g)+f_k(C_k^h)$ and after the swapping, the value is $f_k(C_k^g+\delta)+f_k(C_k^h-\delta)$. We need to compare these two values. Take $\alpha_1=\beta_2=1-\frac{\delta}{C_k^h-C_k^g}$ and $\alpha_2=\beta_1=\frac{\delta}{C_k^h-C_k^g}$. Since $f_k$ is convex, inequalities

$$\alpha_1 f_k(C_k^g)+\alpha_2 f_k(C_k^h)\ge f(\alpha_1 C_k^g+\alpha_2 C_k^h)$$

and

$$\beta_1 f_k(C_k^g)+\beta_2 f_k(C_k^h)\ge f(\beta_1 C_k^g+\beta_2 C_k^h)$$

hold. Therefore, inequalities

$$f_k(C_k^g)+f_k(C_k^h)\ge f(\alpha_1 C_k^g+\alpha_2 C_k^h)+f(\beta_1 C_k^g+\beta_2 C_k^h)$$

and

$$f_k(C_k^g)+f_k(C_k^h)\ge f(C_k^g+\delta)+f(C_k^h-\delta)$$

hold. Thus, the $f_k$ value does not increase after the swapping.

Now, if it happens that after such a swapping the schedule becomes infeasible, i.e., $J_l$ is processed in $[C_k^h-\delta,C_k^h]$ on some other machine, say $M_q\ne M_z$, then we swap job $J_l$ from $[C_k^h-\delta,C_k^h]$ and $J_f$ (if any) from $[C_k^g,C_k^g+\delta]$ on machine $M_q$. Since inequalities $C_l^g\ge C_k^g+\delta$ and $C_f^g\ge C_k^g+\delta$ hold, also inequalities $C_l^h\ge C_k^h$ and $C_f^h\ge C_k^h$ hold.

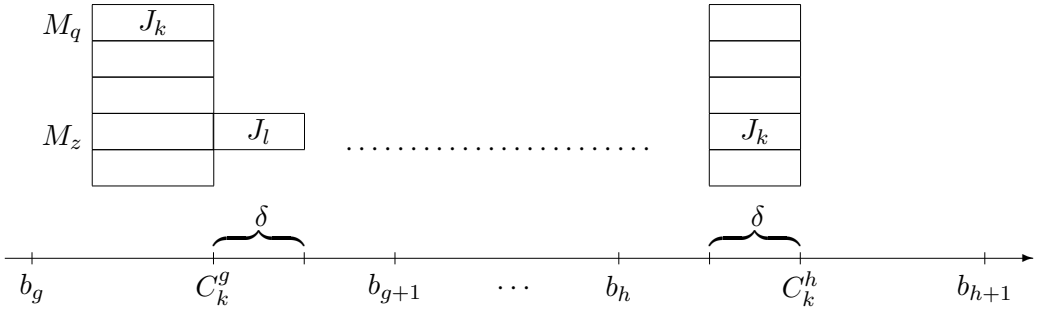


Figure 2: Swap of $J_k$ from $[C_k^h - \delta, C_k^h]$ and $J_l$ from $[C_k^g, C_k^g + \delta]$.

Since this swapping does not influence the jobs within the intervals $[C_k^g + \delta, b_{g+1}]$ and $[C_k^h, b_{h+1}]$, the completion times of the jobs $J_l$ and $J_f$ are not changed.

We will continue with this swapping as long as the schedule remains infeasible. The maximal number of required swaps is determined by the number of different due dates, by the number of different $C_k^i$ values for $k = 1, \ldots, n$ and $i = 0, \ldots, z$, and by the number of preemptions.

Note that any swapping does not change the value of function (2.2). □

Since the described transformation does not change the value $(C_j^0 - b_0) + \cdots + (C_j^z - b_z)$ for each job $J_j$, we do not need to apply the transformation. As a result of solving problem (2.2)-(2.8), we obtain the values

$$C_1 = (C_1^0 - b_0) + \cdots + (C_1^z - d_z),$$

$$\ldots,$$

$$C_n = (C_n^0 - b_0) + \cdots + (C_n^z - d_z),$$

and we can reconstruct an optimal schedule using the known values $C_1, \ldots, C_n$ by solving the corresponding network flow problem, see pages 255–256 in [6] and [8].

Thus, to solve problem $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$, one has to do the following:

1. Solve the corresponding problem (2.2)-(2.8),
   and calculate the values $C_j = \sum_{i=0}^{z} (C_j^i - b_i)$ for each job $J_j$.
2. Reconstruct an optimal schedule by solving the corresponding network flow problem.

Note that the problem of minimizing separable convex objectives subject to linear constraints was considered in [2], where an equivalent linear program was proposed.

# 3 A polynomial algorithm for problem $Q \mid p_j = p,$ pmtn $\mid \sum T_j$

In this section, we derive a polynomial algorithm for problem $Q \mid p_j = p, \text{pmtn} \mid \sum T_j$. The algorithm is analogous to the algorithm from the previous section. Throughout the section, we suppose that the jobs are numbered in such a way that $d_1 \leq \cdots \leq d_n$ holds.

**Statement 2** *For problem $Q \mid p_j = p, pmtn \mid \sum T_j$, an optimal schedule can be found in the class of schedules for which*

$$C_1 \leq C_2 \leq \cdots \leq C_n$$

*holds, i.e., there exists an optimal solution in which the completion times are in the same order as the due dates.*

**Proof:** The proof follows from simple interchange arguments. Suppose that in some optimal schedule, there are two jobs $J_i$ and $J_j$ such that $d_i < d_j$ and $C_i > C_j$ holds. Swap jobs $J_i$ and $J_j$. It is easy to see that the value of $T_i + T_j$ does not increase. □

Thus, we will look for an optimal schedule among the class of schedules for which $C_1 \leq \cdots \leq C_n$ holds. Furthermore, we set $d_0 = 0$ and $d_{n+1} = n \cdot p$, i.e., $[d_0, d_{n+1}]$ is the time interval within which all jobs have to be processed. Note that the set of all due dates $\{d_0, d_1, \ldots, d_{n+1}\}$ together with the set of all completion times $\{C_1, \ldots, C_n\}$ define a partition of the time interval. If we know both sets, then an optimal schedule can be easily found using a reduction to a network flow problem, see [6].

Now, for each job $j \in \{1, \ldots, n\}$, we define the 'completion' time of job $J_j$ for each interval $[d_i, d_{i+1}]$ : For each job $J_j$ with $j = 1, \ldots, n$ and for each interval $[d_i, d_{i+1}]$ with $i = 0, \ldots, n$, we define the value $C_j^i$ such that, if some part of job $J_j$ is scheduled in $[d_i, d_{i+1}]$, then this part has to be scheduled in $[d_i, C_j^i]$, i.e., there is no part of job $J_j$ processed within the interval $[C_j^i, d_{i+1}]$. Moreover, we set $C_1^i \leq C_2^i \leq \ldots \leq C_n^i$.

Thus, if we know a feasible schedule $s$, we can set

$$C_j^i = \begin{cases} C_j(s) & \text{if} \quad d_i < C_j(s) < d_{i+1} \\ d_i & \text{if} \quad C_j(s) \leq d_i \\ d_{i+1} & \text{if} \quad C_j(s) \geq d_{i+1} \end{cases} \tag{3.1}$$

for each $j = 1, \ldots, n$, and $i = j, \ldots, n$. Using equalities (3.1), we can calculate the value $T_j$ by the formula

$$T_j = (C_j^j - d_j) + \ldots + (C_j^n - d_n).$$

Indeed, let $C_j(s) \in [d_i, d_{i+1}]$, then we have the following properties:

if $d_{i+1} \leq d_j$, then $C_j^j = d_j, \ldots, C_j^n = d_n$ and we obtain $T_j = 0$;

if $d_{i+1} > d_j$, then $C_j^j = d_{j+1}, \ldots, C_j^{i-1} = d_i, C_j^i = C_j(s), C_j^{i+1} = d_{i+1}, \ldots, C_j^n = d_n$, i.e., $(C_j^j - d_j) + \ldots + (C_j^n - d_n) = (d_{j+1} - d_j) + (d_{j+2} - d_{j+1}) + \ldots + (d_i - d_{i-1}) + (C_j(s) - d_i) + (d_{i+1} - d_{i+1}) + \ldots + (d_n - d_n) = C_j(s) - d_j = T_j$ holds.

**Example 1.** Consider the schedule given in Figure 3. If we define $C_j^i$ as in (3.1), then we obtain
$C_0^0 = 0$, $C_1^0 = C_2^0 = C_3^0 = C_4^0 = 3$, $C_1^1 = C_2^1 = C_3^1 = C_4^1 = 3$, $C_1^2 = 5$, $C_2^2 = 6$, $C_3^2 = 6$, $C_4^2 = 6$, $C_1^3 = C_2^3 = C_3^3 = C_4^3 = 6$, and $C_1^4 = 6$, $C_2^4 = 6$, $C_3^4 = 7$, $C_4^4 = 9$.
One can see that
$T_1 = 2$ and $(C_1^1 - d_1) + (C_1^2 - d_2) + (C_1^3 - d_3) + (C_1^4 - d_4) = 2$,
$T_2 = 3$ and $(C_2^2 - d_2) + (C_2^3 - d_3) + (C_2^4 - d_4) = 3$,
$T_3 = 1$ and $(C_3^3 - d_3) + (C_3^4 - d_4) = 1$, and
$T_4 = 3$ and $(C_4^4 - d_4) = 3$, hold.

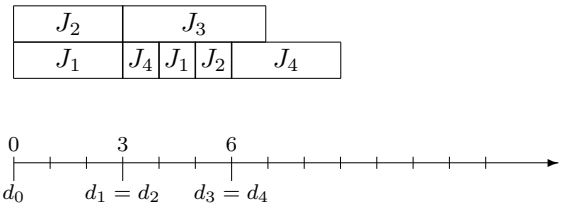


Figure 3: A feasible schedule with $C_1 < C_2 < C_3 < C_4$.

So, for each $i = 0, \ldots, n$, the values

$$d_i = C_0^i \leq C_1^i \leq \cdots \leq C_n^i \leq C_{n+1}^i = d_{i+1}$$

define a partition of the interval $[d_i, d_{i+1}]$.

In turn, each interval $[C_k^i, C_{k+1}^i]$ is completely defined by the jobs processed in it. Thus, we denote by $v_j^q([C_k^i, C_{k+1}^i])$ the part (amount) of job $J_j$ processed in the interval $[C_k^i, C_{k+1}^i]$ on machine $M_q$, see Figure 4, i.e., the total processing time of job $J_j$ on machine $M_q$ in the interval $[C_k^i, C_{k+1}^i]$ equals $\frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q}$ and for any job $J_j$, equality

$$\sum_{q=1}^{m} \sum_{k=0}^{n} \sum_{i=0}^{n} v_j^q([C_k^i, C_{k+1}^i]) = p$$

holds.

The values $C_k^i$, where $k = 0, \ldots, n+1, \quad i = 0, \ldots, n$, and the values $v_j^q([C_k^i, C_{k+1}^i])$, where $j = 1, \ldots, n, \quad i = 0, \ldots, n, \; k = 0, \ldots, n+1, \quad q = 1, \ldots, m$, define a feasible solution of the following linear program.

Minimize

$$\sum_{k=1}^{n} \sum_{i=k}^{n} (C_k^i - d_i) \tag{3.2}$$

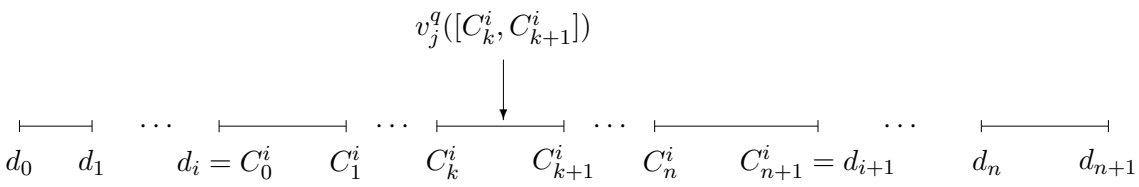


Figure 4: A partition of the interval $[d_i, d_{i+1}]$.

subject to

$$d_i = C_0^i \le C_1^i \le \cdots \le C_n^i \le C_{n+1}^i = d_{i+1}, \qquad i = 0, \ldots, n \tag{3.3}$$

$$\sum_{q=1}^{m} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \le C_{k+1}^i - C_k^i, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n \tag{3.4}$$

$$\sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \le C_{k+1}^i - C_k^i, \qquad i = 1, \ldots, n, \quad q = 1, \ldots, m, \quad k = 1, \ldots, n \tag{3.5}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{q=1}^{m} v_j^q([C_k^i, C_{k+1}^i]) = p, \qquad j = 1, \ldots, n \tag{3.6}$$

$$v_j^q([C_k^i, C_{k+1}^i]) = 0, \qquad i = 0, \ldots, n, \quad j = 1, \ldots, n, \quad q = 1, \ldots, m, \quad k = j, \ldots, n \tag{3.7}$$

$$C_k^i \ge 0, \qquad i = 0, \ldots, n, \quad k = 0, \ldots, n+1 \tag{3.8}$$

$$v_j^q([C_k^i, C_{k+1}^i]) \ge 0, \qquad i = 0, \ldots, n, \quad j = 1, \ldots, n, \quad k = 0, \ldots, n, \quad q = 1, \ldots, m. \tag{3.9}$$

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

**Theorem 3** *For any feasible schedule $s$ of problem $Q \mid p_j = p, pmtn \mid \sum T_j$, where the completion times $C_1(s) \le \cdots \le C_n(s)$ are in the same order as the due dates $d_1 \le \cdots \le d_n$, there exists a corresponding feasible solution of problem (3.3)-(3.9) such that*

$$\sum_{j=1}^{n} T_j(s) = \sum_{k=1}^{n} \sum_{i=k}^{n} (C_k^i - d_i)$$

*holds.*

**Proof:** Let $s$ be a feasible schedule of problem $Q \mid p_j = p, \text{pmtn} \mid \sum T_j$ with $d_1 \leq \cdots \leq d_n$ and $C_1 \leq \cdots \leq C_n$. Using schedule $s$ and equalities (3.1), we obtain the values of all variables $C_k^i$ and $v_j^q([C_k^i, C_{k+1}^i])$.

First, condition (3.3) holds since for any two jobs $x$ and $y$ such that $d_x \leq d_y$ with $C_x \in [d_a, d_{a+1}]$ and $C_y \in [d_b, d_{b+1}]$, we obtain

$$C_x^0 = d_1, \cdots, \ C_x^{a-1} = d_a, \ C_x^a \in [d_a, d_{a+1}], \ C_x^{a+1} = d_{a+1}, \cdots, \ C_x^n = d_n,$$

and

$$C_y^0 = d_1, \cdots, \ C_y^{b-1} = d_b, \ C_y^b \in [d_b, d_{b+1}], \ C_y^{b+1} = d_{b+1}, \cdots, \ C_y^n = d_n.$$

Since $a \leq b$, we have $C_x^i \leq C_y^i$ for any $i = 0, \ldots, n$.

Inequalities (3.4) hold since all parts $v_j^1([C_k^i, C_{k+1}^i]), \cdots, v_j^m([C_k^i, C_{k+1}^i])$ of job $j$ have to be scheduled in the interval $[C_k^i, C_{k+1}^i]$ on different machines without overlapping.

Inequalities (3.5) hold since the parts $v_1^q([C_k^i, C_{k+1}^i]), \cdots, v_n^q([C_k^i, C_{k+1}^i])$ of all jobs have to be scheduled in the interval $[C_k^i, C_{k+1}^i]$ on machine $q$ without overlapping.

For each job $j$, the sum of all values $v_j^q([C_k^i, C_{k+1}^i])$ has to be equal to $p$ since job $j$ has to be processed completely. Therefore, equalities (3.6) must hold.

Equalities (3.7) hold since

$$v_j^q([C_j^i, C_{j+1}^i]) = \cdots = v_j^q([C_n^i, C_{n+1}^i]) = 0,$$

i.e., no part of any job can be processed after its completion time.

Furthermore, function (3.2) corresponds to the optimality criterion $\sum_{j=1}^n T_j(s)$ because

$$C_j^i - d_i = \begin{cases} C_j(s) - d_i & \text{if} \quad d_i < C_j(s) < d_{i+1} \\ 0 & \text{if} \quad C_j(s) \leq d_i \\ d_{i+1} - d_i & \text{if} \quad C_j(s) \geq d_{i+1} \end{cases}$$

and therefore, $T_j(s) = C_j(s) - d_j = \sum_{i=j}^n (C_j^i - d_i) = (C_j^j - d_j) + \cdots + (C_j^n - d_n)$. □

Now we prove

**Theorem 4** *Any feasible solution of problem (3.2)-(3.9) provides a feasible schedule $s$ for the scheduling problem $Q \mid p_j = p, pmtn \mid \sum T_j$ such that*

$$\sum_{j=1}^n T_j(s) = \sum_{j=1}^n \sum_{i=j}^n (C_j^i - d_i)$$

*holds.*

**Proof:** Let $v_j^q([C_k^i, C_{k+1}^i])$ and $C_k^i$, where $k = 0, \ldots, n$, $i = 0, \ldots, n$, $j = 1, \ldots, n$, and $q = 1, \ldots, m$, be a feasible solution of problem (3.2)-(3.9). For any interval $[C_k^i, C_{k+1}^i]$, it is possible to construct a feasible schedule with the length

$$\max \left\{ \max_{1 \le j \le n} \sum_{q=1}^{m} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q}, \max_{1 \le q \le m} \sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \right\},$$

with a finite number of preemptions, see [9]. Taking into account inequalities (3.4) and (3.5), we obtain

$$\max \left\{ \max_{1 \le j \le n} \sum_{q=1}^{m} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q}, \max_{1 \le q \le m} \sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \right\} \le C_{k+1}^i - C_k^i.$$

Thus, for any interval $[C_{k+1}^i, C_k^i]$, one can construct a feasible schedule. Therefore, for any feasible solution of problem (3.2)–(3.9), one can construct a feasible schedule $\tilde{s}$.

Now, if for any $k = 0, \ldots, n$, the values $C_k^i$ satisfy equalities (3.1), then $T_k(\tilde{s}) = \sum_{i=k}^{n}(C_k^i - d_i)$ and therefore, $\sum_{k=1}^{n}\sum_{i=k}^{n}(C_k^i - d_i) = T_k(\tilde{s})$ holds. Thus, $\tilde{s}$ is an optimal schedule.

Now, suppose that for some job $J_k$, the values $C_k^i$ do not satisfy equalities (3.1). Then there exist two intervals $[d_g, d_{g+1}]$ and $[d_h, d_{h+1}]$ such that $d_g \le C_k^g < d_{g+1} \le d_h < C_k^h \le d_{h+1}$ holds. Transform the schedule $\tilde{s}$ in the following way. Take the largest value of $\delta$ such that in the intervals $[C_k^g, C_k^g + \delta]$ and $[C_k^h - \delta, C_k^h]$, each machine is either idle or processes exactly one job. Now, we swap $J_k$ from the interval $[C_k^h - \delta, C_k^h]$ and $J_l$ (if any) from the interval $[C_k^g, C_k^g + \delta]$ on the same machine, say $M_z$ (see Figure 5). Set $C_k^g = C_k^g + \delta$ and $C_k^h = C_k^h - \delta$. Since in the interval $[d_g, d_{g+1}]$ inequality $C_g^l > C_k^j$ holds, it follows that inequality $d_g \le d_l$ holds and therefore, $C_l^h \ge C_l^h$ holds. This implies that after the described swapping the completion time of job $J_l$ is not changed.

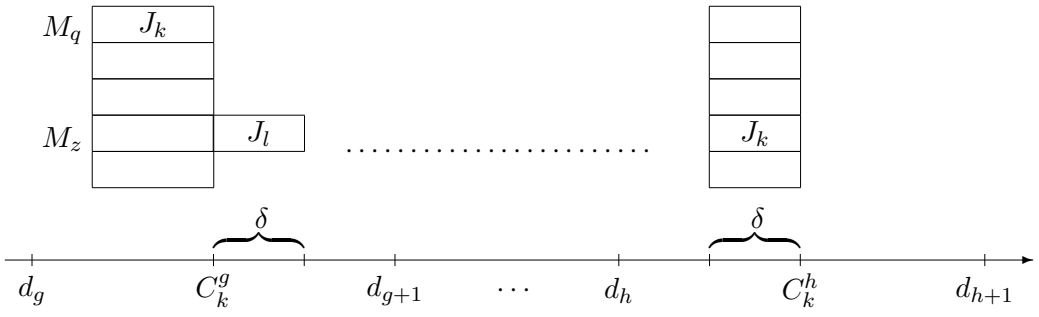


Figure 5: Swap of $J_k$ from $[C_k^h - \delta, C_k^h]$ and $J_l$ from $[C_k^g, C_k^g + \delta]$.

Now, if it happens that after such a swapping the schedule becomes infeasible, i.e., $J_l$ is processed in $[C_k^h - \delta, C_k^h]$ on some other machine, say $M_q \ne M_z$, then we swap job $J_l$

from $[C_k^h - \delta, C_k^h]$ and $J_f$ (if any) from $[C_k^g, C_k^g + \delta]$ on machine $M_q$. Since inequalities $C_l^g \geq C_k^g + \delta$ and $C_f^g \geq C_k^g + \delta$ hold, also inequalities $C_l^h \geq C_k^h$ and $C_f^h \geq C_k^h$ hold. Since this swapping does not influence the jobs within the intervals $[C_k^g + \delta, d_{g+1}]$ and $[C_k^h, d_{h+1}]$, the completion times of the jobs $J_l$ and $J_f$ are not changed.

We will continue with this swapping as long as the schedule remains infeasible. The maximal number of equired swaps is determined by the number of different due dates, by the number of different $C_k^i$ values for $k = 1, \ldots, n$ and $i = 0, \ldots, n$, and by the number of preemptions.

Note that any swapping does not change the value of function (3.2). □

Since the described transformation does not change the value $(C_j^j - d_j) + \cdots + (C_j^n - d_n)$ for each job $J_j$, we do not need to apply the transformation. As a result of solving problem (3.2)-(3.9), we obtain the values

$$C_1 = d_1 + (C_1^1 - d_1) + \cdots + (C_1^n - d_n),$$

$$\cdots,$$

$$C_n = d_n + (C_n^n - d_n),$$

and we can reconstruct an optimal schedule using the known values $C_1, \ldots, C_n$ by solving the corresponding network flow problem, see pages 255–256 in [6] and [8].

Thus, to solve problem $Q \mid p_j = p, \text{pmtn} \mid \sum T_j$, one has to do the following:

1. Solve the corresponding linear program (3.2)-(3.9),
   and calculate the values $C_j = d_j + \sum_{i=j}^{n} (C_j^i - d_i)$ for each job $J_j$.
2. Reconstruct an optimal schedule by solving the corresponding network flow problem.

In fact, now one can easily derive a polynomial algorithm for problem $Q \mid p_j = p,\ \text{pmtn} \mid \sum w_j C_j$, since it is quite analogous to the algorithm for problem $Q \mid p_j = p,\ \text{pmtn} \mid \sum T_j$.

# 4 NP-hardness of problem $P \mid \text{pmtn} \mid \sum T_j$

In [3], it has been shown that problem $1 \mid\mid \sum T_j$ is NP-hard in the ordinary sense. Since preemptions do not reduce the total tardiness on one machine, problem $1 \mid \text{pmtn} \mid \sum T_j$ is NP-hard in the ordinary sense, too. For the case of parallel machines, preemptions can reduce the total tardiness. In this section, we show that problem $P \mid \text{pmtn} \mid \sum T_j$ is NP-hard (in the ordinary sense). The proof is obtained by a reduction from PARTITION:

Given a set of positive integers $a_1, \ldots, a_k, b$ with $\sum_{i=1}^{k} a_i = 2b$. Does there exist a subset $I \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in I} a_i = b$?

Given any instance of PARTITION, we define a corresponding instance of problem $P \mid \text{pmtn} \mid \sum T_j$ as follows. Let $n = 2k^2 + k + 1$ and $m = k$. We set $L = \frac{4kb^3+2b}{k}$. There are three classes of jobs:

- the class of $a$-jobs comprises $k$ jobs with $p_i = a_i$ and $d_i = L$ for $i = 1, \ldots, k$;
- the class of $ba$-jobs comprises
  - $2k$ jobs with processing time $b^2 a_1$ and due date $L - a_1$;
  - $\cdots$;
  - $2k$ jobs with processing time $b^2 a_k$ and due date $L - a_k$;
- one *long* job with processing time $b^3$ and due date $b^3$.

We claim that PARTITION has a solution if and only if there exists a schedule with $\sum T_j \leq b^3 + b$.

First, we show that, if PARTITION has a solution, then for the corresponding instance of problem $P \mid \text{pmtn} \mid \sum T_j$ a schedule exists with $\sum T_j \leq b^3 + b$. Without loss of generality, suppose that the solution for PARTITION is $\{a_1, \ldots, a_z\}$, i.e., we suppose that $\sum_{i=1}^{z} a_i = b$ holds. Then:

- we schedule the $ba$-jobs with the processing times $b^2 a_1, \ldots, b^2 a_z$ in the intervals $[L, L + b^2 a_1], \ldots, [L, L + b^2 a_z]$, correspondingly;
- we schedule the $a$-jobs with the processing times $a_1, \ldots, a_k$ in the intervals $[L - a_1, L], \ldots, [L - a_k, L]$, correspondingly;
- we schedule the *long* job in the interval $[0, b^3]$, see Figure 6; and
- we schedule the remaining $ba$-jobs with preemptions in the interval $[0, L]$.

Note that after scheduling all $a$-jobs and one *long* job, the idle time in the interval $[0, L]$ on $k$ machines is $Lk - b^3 - 2b$. At the same time, all unscheduled $ba$-jobs require $2k \sum_{i=1}^{k} b^2 a_i - \sum_{i=1}^{z} b^2 a_i = 4kb^3 - b^3$ time. Taking into account that $L = \frac{4kb^3+2b}{k}$, we conclude that the obtained schedule is feasible. Moreover, for the constructed schedule $\sum T_j = b^3 + b$ holds.

Next, we show that, if there exists a schedule for which $\sum T_j \leq b^3 + b$ holds, then PARTITION has a solution. Consider a schedule with $\sum T_j \leq b^3 + b$. The *long* job has to be scheduled within the interval $[0, L]$ since otherwise the tardiness of the *long* job is larger than $L - b^3$, i.e., inequality $\sum T_j > b^3 + b$ holds.

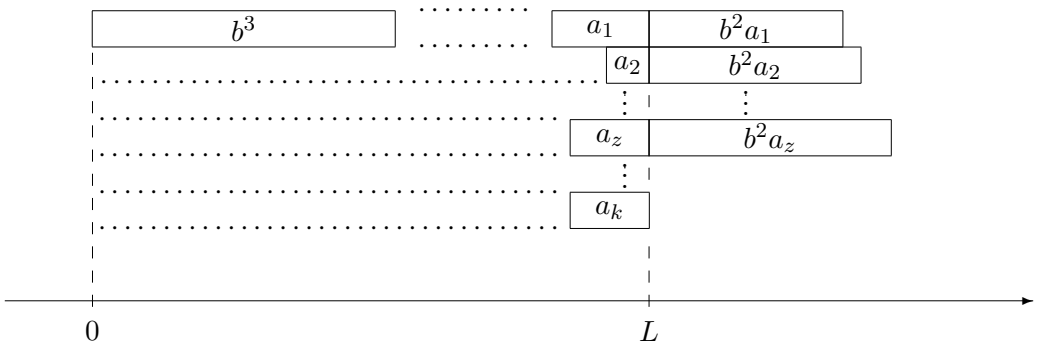


Figure 6: Scheduling jobs for the given partition.

Take all jobs that are completed after the time point $L$. Without loss of generality, we denote this set of jobs by $\{1, \ldots, u\}$ for the $ba$-jobs and by $\{1, \ldots, v\}$ for the $a$-jobs. Let each $ba$-job $i = 1, \ldots, u$ be processed for $b^2\beta_i$ time units after time point $L$, and each $a$-job $j = 1, \ldots, v$ is processed for $\alpha_j$ time units after time point $L$. Then we have

$$\sum_{i=1}^{n} T_i \geq b^2 \sum_{i=1}^{u} \beta_i + \sum_{i=1}^{u} a_i + \sum_{i=1}^{v} \alpha_i,$$

i.e., by assumption on the $\sum T_j$-value, we obtain

$$b^2 \sum_{i=1}^{u} \beta_i + \sum_{i=1}^{u} a_i + \sum_{i=1}^{v} \alpha_i \leq \sum_{i=1}^{n} T_i \leq b^3 + b. \tag{4.1}$$

Note that

$$b^2 \sum_{i=1}^{u} \beta_i + \sum_{i=1}^{v} \alpha_i \geq b^3. \tag{4.2}$$

Suppose that

$$\sum_{i=1}^{u} a_i > b \tag{4.3}$$

holds. Then, substituting (4.3) and (4.2) into inequality (4.1), we obtain that

$$b^3 + b < \sum_{i=1}^{n} T_i \leq b^3 + b$$

holds. Therefore, $\sum_{i=1}^{u} a_i \leq b$ holds.

Now, suppose that $\sum_{i=1}^{u} a_i \leq b - 1$ holds. Then $\sum_{i=1}^{u} \beta_i \leq b - 1$ holds. Substituting it into inequality (4.2), we obtain

$$b^3 \leq b^2 \sum_{i=1}^{u} \beta_i + \sum_{i=1}^{v} \alpha_i \leq b^2(b-1) + 2b \leq b^3 - b^2 + 2b,$$

which is also impossible.

Thus, we obtain $\sum_{i=1}^{u} a_i = b$, i.e., the set of $ba$-jobs completed after time point $L$ defines the solution to PARTITION.

# 5 NP-hardness of problem $P \mid r_j, p_j = p, \mathbf{pmtn} \mid \sum T_j$

In [4], it has been shown that problem $P \mid r_j, p_j = p \mid \sum f_j(C_j)$, where $f_j$ is a non-decreasing function such that for any $i$ and $j$ function $f_i - f_j$ is monotonic, can be solved in polynomial time. Therefore, problem $P \mid r_j, p_j = p \mid \sum T_j$ can be solved in polynomial time, too. In this section, we show that problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum T_j$ is NP-hard in the ordinary sense. The proof is analagous to the NP-hardness proof for problem $P \mid \text{pmtn} \mid \sum T_j$, and it is obtained by a reduction from PARTITION: Given a set of positive integers $a_1, \ldots, a_k, b$ with $\sum_{i=1}^{k} a_i = 2b$. Does there exist a subset $I \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in I} a_i = b$?

Given any instance of PARTITION, we define a corresponding instance of problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum T_j$ as follows. Let $n = 8(2k^2 + 2k + 1)(2k^2 + k + 1) + 2(2k^2 + k + 1)$, $p = k^2 L^2$, where $L = \frac{4b^3 k + 2b}{k}$ and $m = 2k^2 + 2k + 1$. There are four classes of jobs:

- the class of $a$-jobs comprises $k$ jobs with $r_i = a_i$ and $d_i = p + L$ for $i = 1, \ldots, k$;
- the class of $ba$-jobs comprises
  - $2k$ jobs with release date $b^2 a_1$ and due date $p + L - a_1$;
  - $\cdots$;
  - $2k$ jobs with release date $b^2 a_k$ and due date $p + L - a_k$;
- one *tight* job with release date $b^3$ and due date $p + b^3$; and
- a class of *frame* jobs:
  - $2k^2 + k + 1$ *frame* jobs with release date $p$ and due date $2p$, see Figure 7;
  - $2k^2 + 2k + 1$ *frame* jobs with release date $2p$ and due date $3p$;
  - $\cdots$
  - $2k^2 + 2k + 1$ *frame* jobs with release date $(16k^2 + 8k + 9)p$ and due date $(16k^2 + 8k + 10)p$.

We claim that PARTITION has a solution if and only if there exists a schedule with $\sum T_j \leq b^3 + b$.

First, we show that, if PARTITION has a solution, then for the corresponding instance of problem $P \mid r_j, p_j = p, \text{pmtn} \mid \sum T_j$ a schedule exists with $\sum T_j \leq b^3 + b$. Without loss of generality, suppose that a solution for PARTITION is $\{a_1, \ldots, a_z\}$, i.e., suppose that $\sum_{i=1}^{z} a_i = b$ holds. Then we construct the following schedule:

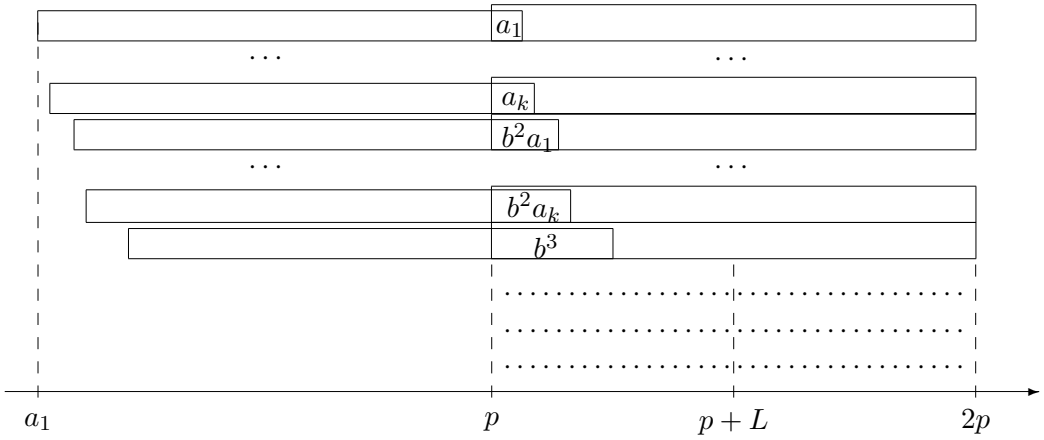


Figure 7: The lengths of the overlapping parts are indicated within the boxes.

- we schedule $2k^2 + k + 1$ *frame* jobs with release date $p$ and due date $2p$ in the interval $[p, 2p]$ on machines $M_{k+1}, \ldots, M_{2k^2+2k+1}$, see Figure 7;
- we schedule $2k^2 + 2k + 1$ *frame* jobs with release date $2p$ and due date $3p$ in the interval $[2p, 3p]$ on machines $M_1, \ldots, M_{2k^2+2k+1}$;
- $\ldots$
- we schedule $2k^2+2k+1$ *frame* jobs with release date $(16k^2+8k+9)p$ and due date $(16k^2 + 8k + 10)p$ in the interval $[(16k^2 + 8k + 9)p, (16k^2 + 8k + 10)p]$ on machines $M_1, \ldots, M_{2k^2+2k+1}$;
- we partially schedule the $a$-jobs with release dates $a_1, \ldots, a_k$ for $p - a_1, \ldots, p - a_k$ time units in the intervals $[a_1, p], \ldots, [a_k, p]$, correspondingly, see Figure 7;
- we partially schedule the $ba$-jobs with release dates $b^2a_1, \ldots, b^2a_k$ for $p-b^2a_1, \ldots, p-b^2a_k$ time units in the intervals $[b^2a_1, p], \ldots, [b^2a_k, p]$, correspondingly, see Figure 7;
- we partially schedule the *tight* job for $p - b^3$ time in the interval $[b^3, p]$, see Figure 7.

We schedule the remaining parts of jobs in the interval $[p, 2p]$ in the following way:

- we schedule the $a$-jobs with the release dates $a_1, \ldots, a_k$ in the intervals $[p + L - a_1, p + L], \ldots, [p + L - a_k, p + L]$, correspondingly;
- we schedule the *tight* job for $b^3$ time units in the interval $[p, p + b^3]$;
- we schedule the $ba$-jobs with the release dates $b^2a_1, \ldots, b^2a_z$ for $b^2a_1, \ldots, b^2a_z$ time units in the intervals $[p+L, p+L+b^2a_1], \ldots, [p+L, p+L+b^2a_z]$, correspondingly; and

- we schedule the remaining parts of the *ba*-jobs with preemptions in the interval $[p, p + L]$.

Note that after scheduling all *frame* jobs, *a*-jobs and one *tight* job, the idle time in the interval $[p, p + L]$ on all machines is $Lk - b^3 - 2b$. At the same time, all parts of *ba*-jobs scheduled in $[p, p + L]$ require $2k \sum_{i=1}^{k} b^2 a_i - \sum_{i=1}^{z} b^2 a_i = 4kb^3 - b^3$ time. Taking into account that $L = \frac{4kb^3+2b}{k}$, we conclude that the obtained schedule is feasible. Besides, for the constructed schedule we have $\sum T_j = b^3 + b$.

Now, we show that, if there is a schedule for which $\sum T_j \leq b^3+b$ holds, then PARTITION has a solution. Let $s$ be a schedule for which $\sum T_j \leq b^3+b$ holds. Note that all *a*-jobs, *ba*-jobs, and the *tight* job have to be completed before time $2p$, since otherwise $\sum T_j > b^3+b$ holds. First, we show that all *frame* jobs with release time $p$ and due date $2p$ have to be scheduled in $[p, 2p]$ in schedule $s$. Suppose that one of the *frame* jobs with release date $p$ is not completely processed in $[p, 2p]$. Let $x$ denotes the largest time within $[p, 2p]$ such that at least one *frame* job, say $J_f$, with release date $p$ is not processed in $[x - \Delta, x]$ for some $\Delta$. Now, we reschedule the part of job $J_f$ from $[2p, \infty[$ to $[x - \Delta, x]$. After rescheduling at least $16k^2 + 8k + 8$ *frame* jobs whose release time is greater than $2p$ will be shifted left by $\Delta$ and at most $2k^2 + k + 1$ jobs, i.e., all *a*-jobs, all *ba*-jobs and one *tight* job have to be shifted right by $\Delta$. Therefore, the $\sum T_j$ value will decrease by at least $(16k^2 + 8k + 8)\Delta - (2k^2 + k + 1)\Delta$.

Thus, we can transform schedule $s$ in such a way that all frame jobs with release time $p$ and due date $2p$ have to be scheduled in $[p, 2p]$ in the schedule $s$. Moreover, it follows that all *frame* jobs are scheduled before their due dates. Thus, in $s$ all parts of *a*-jobs with lengths $a_1, \ldots, a_k$, all parts of *ba*-jobs with lengths $b^2a_1, \ldots, b^2a_k$ and the part of the *tight* job with length $b^3$ have to be scheduled in the interval $[p, 2p]$ on $k$ machines in such a way that the sum of tardiness for all *a*-jobs, *ba*-jobs and *tight* job is less than or equal to $b^3 + b$. From Section 4, we know that this is possible only if PARTITION has a solution.

# 6 Concluding Remarks

In this paper, we proposed a polynomial reduction of problem $Q \mid p_j = p, \text{pmtn} \mid \sum f_j$ for a special class of functions $f_j$, to the problem of minimizing a convex-separable function subject to linear constraints. We used this reduction to derive polynomial algorithms for problems $Q \mid p_j = p, \text{ pmtn} \mid \sum T_j$ and $Q \mid p_j = p, \text{ pmtn} \mid \sum w_j C_j$. We have shown that both the problem $P \mid \text{ pmtn} \mid \sum T_j$ of minimizing total tardiness on a set of parallel machines with allowed preemptions and the problem $P \mid r_j, p_j = p, \text{ pmtn} \mid \sum T_j$ of minimizing total tardiness on a set of parallel machines with release dates, equal processing times and allowed preemptions are NP-hard.

Note that we have proved NP-hardness only in the ordinary sense. The complexity status of both problems $P \mid \text{ pmtn} \mid \sum T_j$ and $P \mid r_j, p_j = p, \text{ pmtn} \mid \sum T_j$ under an unary encoding is still an open question.

ACKNOWLEDGEMENTS

The results presented in this paper were attained during a visit of S.A. Kravchenko at the Otto-von-Guericke-Universität of Magdeburg which was supported by a fellowship of the Alexander von Humboldt Foundation.

# References

[1] Baptiste Ph., Brucker P., Chrobak M., Dürr C., Kravchenko S.A., Sourd F. The complexity of mean flow time scheduling problems with release times. Journal of Scheduling 2007; 10: 139-146.

[2] Dantzig G.B. Linear programming and extensions. Princeton University Press, Princeton, New Jersey, 1998.

[3] Du J., Leung J. Y.-T. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 1990; 15: 483–495.

[4] Kravchenko S.A., Werner F. On a parallel machine scheduling problem with equal processing times. Discrete Applied Mathematics 2009; 157: 848–852.

[5] Kravchenko S.A., Werner F. Preemptive scheduling on uniform machines to minimize mean flow time. Computers & Operations Research 2009; 36: 2816–2821.

[6] Labetoulle J., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Preemptive scheduling of uniform machines subject to release dates. In: Pulleyblank HR (Ed.). Progress in Combinatorial Optimization. Academic Press: New York, 1984, p. 245-261.

[7] Lawler E.L. A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. 1977; 1: 331–342.

[8] Lawler E.L. Combinatorial Optimization: Networks and Matroids. New York: Holt, Rinehart and Winston, 1976.

[9] Lawler E.L. Recent results in the theory of machine scheduling. In: Bachem A, Grötschel M, Korte B (Eds.). Mathematical Programming: The State of the Art. Springer-Verlag: Berlin, 1983, p. 202–234.

[10] Tian Z., Ng C.T., Cheng T.C.E. An $O(n^2)$ algorithm for scheduling equal-length preemptive jobs on a single machine to minimize total tardiness. Journal of Scheduling 2006; 9: 343–364.