Minimizing a Regular Function on Uniform Machines with Ordered Completion Times

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

October 22, 2010

Abstract

The scheduling problem we are dealing with is the following one. A set of n jobs has to be scheduled on a set of uniform machines. Each machine can handle at most one job at a time. Each job $J_j, j = 1, \ldots, n$, has an arbitrary due date d_j. Job J_j becomes available for processing at its release date r_j and has to be scheduled by its deadline D_j. 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 with a given order of the completion times that minimizes a non-decreasing function F. Thus, we consider problem $Q \mid r_j,\ \text{pmtn}, D_j \mid F$ and want to find an optimal schedule among the schedules with a given order of completion times. We show that problem $Q \mid r_j,\ \text{pmtn}, D_j \mid F$ with a given order of completion times is equivalent to the problem of minimizing a function F subject to linear constraints.

Keywords: uniform machines, linear programming

[*]Corresponding author. Tel.:+49 391 6712025; fax:+49 391 6711171.
E-mail address: frank.werner@mathematik.uni-magdeburg.de (F.Werner).

1 Introduction

Linear programming is a powerful tool for solving optimization problems. In the area of scheduling, it is sufficient to mention the results from [2] and [3] for problem $R \mid\mid \sum C_j$ and from [4] for problem $R \mid$ pmtn $\mid C_{max}$ to demonstrate the power and elegance of linear programming (in this paper, we always use the standard 3-parameter classification scheme $\alpha \mid \beta \mid \gamma$ for scheduling problems introduced in [1]).

For scheduling problems with equal processing times (for a survey on such problems see [8]), the idea of using linear programming was exploited in [6], where a polynomial algorithm for problem $Q \mid r_j, p_j = p,$ pmtn $\mid \sum C_j$ has been derived. In spite of the fact that the order of the completion times in an optimal schedule is known in advance, the complexity status of this problem remained open for a long time. A subsequent step was the consideration of problem $Q \mid r_j, p_j = p,$ pmtn $\mid \sum w_j T_j$ in [7]. It appeared that problem $Q \mid r_j, p_j = p,$ pmtn $\mid \sum T_j$ is NP-hard whereas problem $Q \mid p_j = p,$ pmtn $\mid \sum T_j$ can be polynomially solved. Note that for the latter problem, the order of the completion times in an optimal schedule is known in advance. In this chapter, we generalize the ideas from [6] and [7] to a more general case.

The problem considered can be stated as follows. There are n independent jobs and m uniform machines. For each job J_j, $j = 1, \ldots, n$, there is given its processing time p_j, its due date $d_j \geq 0$, its release time $r_j \geq 0$, i.e., the processing of any job can be started only at or after its release date, and its deadline $D_j \geq 0$, i.e., no part of the job can be processed after its deadline. Each machine M_q, $q = 1, \ldots, m$, has some speed $s_q \geq 1$, i.e., the execution of job J_j on machine M_q requires p_j/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. For a schedule s, let $F = F(C_1(s), \ldots, C_n(s))$ denote a non-decreasing function depending on the variables $C_j(s)$, $j = 1, \ldots, n$, 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 F within the class of schedules with a fixed order of completion times. The described problem can be denoted as $Q \mid r_j, \text{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid F$. We show that this problem is equivalent to the problem of minimizing a function F subject to linear constraints.

This chapter is organized as follows. In Section 2, we describe a polynomial reduction of problem $Q \mid r_j, \text{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid F$ to the problem of minimizing a non-decreasing function F subject to linear constraints. In Section 3, we apply the derived model to problem $Q \mid r_j, \text{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid \sum w_j T_j$ and show that this problem can be polynomially solved.

2 Problem $Q \mid r_j, \mathbf{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid F$

In this section, we show that problem $Q \mid r_j, \text{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid F$, where F is a non-decreasing function, can be reduced to the problem of minimizing a non-decreasing function F subject to linear constraints. The idea of the reduction is taken from [6], where a polynomial algorithm for problem $Q \mid r_j, p_j = p,\ \text{pmtn} \mid \sum C_j$ has been derived.

Suppose that all jobs are numbered according to non-decreasing completion times, i.e., $i < j$ implies $C_i \leq C_j$ for any pair of jobs J_i and J_j. Thus, we will look for an optimal schedule among the class of schedules for which $C_1 \leq \ldots \leq C_n$ holds.

Note that, if $i < j$ and $D_i > D_j$, then $C_i \leq C_j$ and $C_j \leq D_j$. Therefore, we can set $D_i := D_j$. Thus, in the following we suppose that $D_1 \leq \ldots \leq D_n$ holds.

Let $\{b_1, \ldots, b_z\}$ with $b_1 < \ldots < b_z$ be the set of release times, due dates and deadlines, i.e., $\{b_1, \ldots, b_z\} = \{r_1, \ldots, r_n\} \cup \{d_1, \ldots, d_n\} \cup \{D_1, \ldots, D_n\}$. Furthermore, we set $b_0 = 0$ and suppose that $b_z < b_{z+1} = \max\{r_1, \ldots, r_n\} + \sum_{j=1}^{n} p_j$, i.e., $[b_0, b_{z+1}]$ 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 [9].

Now, for each job $j \in \{1, \ldots, n\}$ and for each interval $[b_i, b_{i+1}]$ such that $[b_i, b_{i+1}] \subseteq [r_j, D_j]$, we define the 'completion' time of job J_j : For each job J_j with $j = 1, \ldots, n$ and for each interval $[b_i, b_{i+1}] \subseteq [r_j, D_j]$ 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 $[C_j^i, b_{i+1}]$. So, for each job $j \in \{1, \ldots, n\}$ and for each interval $[b_i, b_{i+1}]$, $i = 0, \ldots, n$, such that $[b_i, b_{i+1}] \subseteq [r_j, D_j]$, the values C_j^i define a partition of the interval $[b_i, b_{i+1}]$. It may happen that there is a job J_j such that $[b_i, b_{i+1}] \not\subseteq [r_j, D_j]$ holds. In this case, we set $C_j^i = C_{j-1}^i$. Moreover, we set $C_1^i \leq \ldots \leq C_n^i$.

For each $j \in \{1, \ldots, n\}$, denote by $v(j)$ the index such that $b_{v(j)} = r_j$ and by $u(j)$ the index such that $b_{u(j)+1} = D_j$.

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 $i = 0, \ldots, z$ such that $[b_i, b_{i+1}] \subseteq [r_j, D_j]$, see Figure 1.

Using equalities (2.1), we can calculate the value C_j by the formula

$$C_j = r_j + (C_j^{v(j)} - b_{v(j)}) + (C_j^{v(j)+1} - b_{v(j)+1}) + \ldots + (C_j^{u(j)} - b_{u(j)}).$$

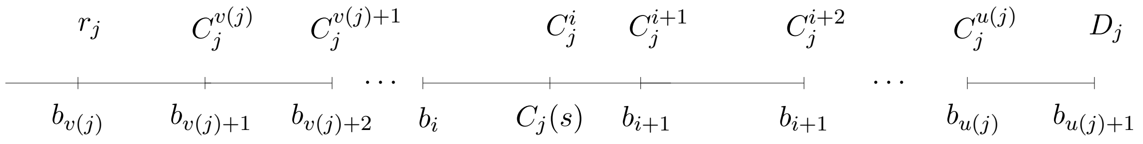

Figure 1: The structure of the interval $[r_j, D_j]$.

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

$$C_j = r_j + (C_j^{v(j)} - b_{v(j)}) + \ldots + (C_j^{i-1} - b_{i-1}) +$$

$$(C_j^i - b_i) + (C_j^{i+1} - b_{i+1}) + \ldots + (C_j^{u(j)} - b_{u(j)}).$$

Since $r_j = b_{v(j)}$, $C_j^{v(j)} = b_{v(j)+1}$, $\ldots$, $C_j^{i-1} = b_i$ and $C_j^{i+1} = b_{i+1}$, $\ldots$, $C_j^{u(j)} = b_{u(j)}$, we obtain

$$C_j = b_{v(j)} + (b_{v(j)+1} - b_{v(j)}) + \ldots + (b_i - b_{i-1}) +$$

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

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 2, 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

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

holds.

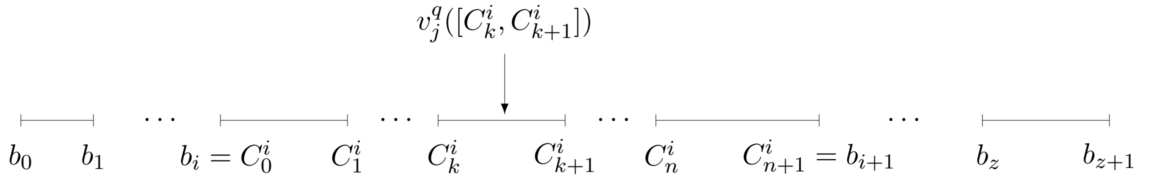

Figure 2: A partition of the interval $[b_i, b_{i+1}]$.

The values $\tilde{C}_j$, where $j = 0, \ldots, n$ and $\tilde{C}_0 = 0$, the values C_k^i, where $k = 0, \ldots, n+1$, $i = 0, \ldots, z$, 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:

Minimize

$$F(\tilde{C}_1, \ldots, \tilde{C}_n) \tag{2.2}$$

subject to

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

$$C_j^i = C_{j-1}^i \quad \text{if} \quad [b_i, b_{i+1}] \not\subseteq [r_j, D_j], \qquad i = 0, \dots, z, \quad j = 1, \dots, n \tag{2.4}$$

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

$$\sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \le C_{k+1}^i - C_k^i, \qquad i = 0, \dots, z, \quad q = 1, \dots, m, \quad k = 1, \dots, n \tag{2.6}$$

$$\sum_{i=0}^{z} \sum_{k=0}^{n} \sum_{q=1}^{m} v_j^q([C_k^i, C_{k+1}^i]) = p_j, \qquad j = 1, \dots, n \tag{2.7}$$

$$\begin{aligned} v_j^q([C_k^i, C_{k+1}^i]) = 0 \qquad \text{if} \qquad & [b_i, b_{i+1}] \not\subseteq [r_j, D_j] \text{ or } j \le k, \\ & i = 0, \dots, z, \quad j = 1, \dots, n, \\ & k = 0, \dots, n, \quad q = 1, \dots, m \end{aligned} \tag{2.8}$$

$$\tilde{C}_j = \max\{\tilde{C}_{j-1}, r_j + \sum_{i=v(j)}^{u(j)} (C_j^i - b_i)\}, \qquad j = 1, \dots, n, \quad \tilde{C}_0 = 0 \tag{2.9}$$

$$C_k^i \ge 0, \qquad i = 0, \dots, z, \quad k = 0, \dots, n \tag{2.10}$$

$$\begin{aligned} v_j^q([C_k^i, C_{k+1}^i]) \ge 0, \qquad & i = 0, \dots, z, \quad j = 1, \dots, n, \\ & k = 0, \dots, n, \quad q = 1, \dots, m \end{aligned} \tag{2.11}$$

$$\tilde{C}_j \ge 0, \qquad j = 1, \dots, n, \quad \tilde{C}_0 = 0 \tag{2.12}$$

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

Theorem 1 *For any feasible schedule s of problem $Q \mid r_j, pmtn, D_j, C_1 \le \ldots \le C_n \mid F$, there exists a corresponding feasible solution of problem (2.3)-(2.12) such that*

$$F(C_1(s), \dots, C_n(s)) = F(\tilde{C}_1, \dots, \tilde{C}_n)$$

holds.

Proof: Let s be a feasible schedule of problem $Q \mid r_j, \text{pmtn}, D_j, C_1 \leq \ldots \leq C_n \mid F$. Using schedule s, equalities (2.1), (2.4) and (2.8), we obtain the values of all variables C_k^i and $v_j^q([C_k^i, C_{k+1}^i])$.

Conditions (2.4) holds by definition.

Conditions (2.9) hold since on the one hand, $C_j(s) \geq C_{j-1}(s)$ holds and on the other hand, $C_j(s) = r_j + \sum_{i=v(j)}^{u(j)}(C_j^i - b_i)$ holds because the values C_j^i satisfy (2.1).

To prove that condition (2.3) holds, i.e., that $b_i = C_0^i \leq C_1^i \leq \cdots \leq C_n^i \leq C_{n+1}^i = b_{i+1}$, consider two jobs J_x and J_y such that $x < y$, i.e., $C_x \leq C_y$ holds.

Let $[b_i, b_{i+1}] \subseteq [r_x, D_x]$ and $[b_i, b_{i+1}] \subseteq [r_y, D_y]$. Consider all possible cases for C_x and C_y.
If $C_x \in [b_i, b_{i+1}]$ and $C_y \in [b_i, b_{i+1}]$ hold, then $C_x^i \leq C_y^i$ since $C_x \leq C_y$ by definition.
If $C_x \in [b_i, b_{i+1}]$ and $C_y \notin [b_i, b_{i+1}]$ hold, then $C_y^i = b_{i+1}$ and therefore, $C_x^i \leq C_y^i$ holds.
If $C_x \notin [b_i, b_{i+1}]$ and $C_y \in [b_i, b_{i+1}]$ hold, then $C_x < b_i$ and hence $C_x^i = b_i$ and therefore, $C_x^i \leq C_y^i$ holds.
If $C_x \notin [b_i, b_{i+1}]$ and $C_y \notin [b_i, b_{i+1}]$ hold, then

$$C_x^i = C_y^i = b_i \text{ if } C_x \leq b_i \text{ and } C_y \leq b_i;$$

$$C_x^i = b_i \text{ and } C_y^i = b_{i+1} \text{ if } C_x \leq b_i \text{ and } C_y \geq b_{i+1};$$

$$C_x^i = C_y^i = b_{i+1} \text{ if } C_x \geq b_{i+1} \text{ and } C_y \geq b_{i+1}.$$

Thus, for each case $C_x^i \leq C_y^i$ holds.

Let $[b_i, b_{i+1}] \not\subseteq [r_x, D_x]$ and $[b_i, b_{i+1}] \not\subseteq [r_y, D_y]$. In this case, $C_x^i = C_{x-1}^i$ and $C_y^i = C_{y-1}^i$ hold and to compare C_x^i and C_y^i, we have to consider C_{x-1}^i and C_{y-1}^i under the condition that $x - 1 < y - 1$ holds.

Let $[b_i, b_{i+1}] \not\subseteq [r_x, D_x]$ and $[b_i, b_{i+1}] \subseteq [r_y, D_y]$. In this case, $C_x^i = C_{x-1}^i$ and to compare C_x^i and C_y^i, we have to consider C_{x-1}^i and C_y^i under the condition that $x - 1 < y$ holds.

Let $[b_i, b_{i+1}] \subseteq [r_x, D_x]$ and $[b_i, b_{i+1}] \not\subseteq [r_y, D_y]$. In this case, $C_y^i = C_{y-1}^i$ and if $y - 1 = x$ holds, then $C_x^i = C_y^i$ but if $y - 1 > x$, then to compare C_x^i and C_y^i, we have to consider C_x^i and C_{y-1}^i under the condition that $x < y - 1$ holds.

Thus, condition (2.3) holds.

Inequalities (2.5) 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 (2.6) 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_j, the sum of all values $v_j^q([C_k^i, C_{k+1}^i])$ has to be equal to p_j since job J_j has to be processed completely. Therefore, equalities (2.7) must hold.

Furthermore, $\tilde{C}_j = C_j(s)$ since, due to (2.1),

$$C_j^i - b_i = \left\{ \begin{array}{ccc} C_j(s) - b_i & \text{if} & b_i < C_j(s) < b_{i+1} \\ 0 & \text{if} & C_j(s) \le b_i \\ b_{i+1} - b_i & \text{if} & C_j(s) \ge b_{i+1} \end{array} \right.$$

holds for any interval $[b_i, b_{i+1}]$ with $[b_i, b_{i+1}] \subseteq [r_j, D_j]$. Therefore,

$$C_j(s) = r_j + \sum_{i=v(j)}^{u(j)} (C_j^i - b_i).$$

Thus, minimizing $F(C_1(s), \dots, C_n(s))$ is equivalent to minimizing $F(\tilde{C}_1, \dots, \tilde{C}_n)$. □

Now we prove

Theorem 2 *Any feasible solution of problem (2.2)-(2.12) provides a feasible schedule s for the scheduling problem $Q \mid r_j, pmtn, D_j, C_1 \le \dots \le C_n \mid F$ such that*

$$F(C_1(s), \dots, C_n(s)) = F(\tilde{C}_1, \dots, \tilde{C}_n)$$

holds.

Proof: Let $v_j^q([C_k^i, C_{k+1}^i])$ and C_k^i, where $k = 0, \dots, n$, $i = 0, \dots, z$, $j = 1, \dots, n$, and $q = 1, \dots, m$, be a feasible solution of problem (2.2)-(2.12). For any interval $[C_k^i, C_{k+1}^i]$, it is possible to construct a feasible schedule with the length

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

having a finite number of preemptions, see [11]. Taking into account inequalities (2.5) and (2.6), 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 $[C_k^i, C_{k+1}^i]$, one can construct a feasible schedule. Therefore, for any feasible solution of problem (2.2)–(2.12), 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

$$C_k(\tilde{s}) = \left\{ \begin{array}{ccc} r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i) & \text{if} & C_{k-1}(\tilde{s}) \le r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i) \\ C_{k-1}(\tilde{s}) & \text{if} & C_{k-1}(\tilde{s}) > r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i). \end{array} \right.$$

In other words,

$$C_k(\tilde{s}) = \max \left\{ r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i), C_{k-1} \right\} = \tilde{C}_k,$$

and therefore,

$$F(C_1(s), \ldots, C_n(s)) = F(\tilde{C}_1, \ldots, \tilde{C}_n)$$

holds.

Now, suppose that for some job J_k, the values C_k^i do not satisfy equalities (2.1). Take the maximal value of k. Then there exist two intervals $[b_g, b_{g+1}]$ and $[b_h, b_{h+1}]$ such that $b_g \leq C_k^g < b_{g+1} \leq b_h < C_k^h \leq b_{h+1}$. Choose $[b_h, b_{h+1}]$ in such a way that $C_k^h = \tilde{C}_k$ holds. Now we describe a transformation of the schedule $\tilde{s}$. The transformation does not change the value of function $F(\tilde{C}_1, \ldots, \tilde{C}_n)$, but it changes schedule $\tilde{s}$.

Transform the schedule $\tilde{s}$ in the following way. Take the largest value of δ 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. Without loss of generality, we suppose that among all jobs processed in $[C_k^h - \delta, C_k^h]$ only one job, namely job J_k, is available but is not processed in $[C_k^g, b_{g+1}]$.

Case 1: $C_{k-1}^h < C_k^h$. 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 3). Set $C_k^g = C_k^g + \delta$ and $C_k^h = C_k^h - \delta$. Since in the interval $[b_h, b_{h+1}]$ inequality $C_l^h \geq C_k^h$ holds, it follows that after the described swapping, the completion time of job J_l is not changed.

Consider the F value before and after the swapping in $[b_g, b_{h+1}]$. Before the swapping, it was $F(\tilde{C}_1, \ldots, \tilde{C}_n)$ and after the swapping, only the value of $\tilde{C}_k$ can changes. Before the transformation according to (2.9), equality $\tilde{C}_k = \max\{\tilde{C}_{k-1}, r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i)\}$ holds, i.e.,

$$\tilde{C}_k = \max\{\tilde{C}_{k-1}, r_k + (C_k^{v(k)} - b_{v(k)}) + \ldots$$
$$+ (C_k^g - b_g) + (C_k^{g+1} - b_{g+1}) + \ldots + (C_k^h - b_h)\}.$$

After the swapping, we have

$$\tilde{C}_k = \max\{\tilde{C}_{k-1}, r_k + (C_k^{v(k)} - b_{v(k)}) + \ldots$$
$$+ (C_k^g + \delta - b_g) + (C_k^{g+1} - b_{g+1}) + \ldots + (C_k^h - \delta - b_h)\}.$$

Thus, one can see that the value of $\tilde{C}_k$ does also not change. It means that the value of $F(\tilde{C}_1, \ldots, \tilde{C}_n)$ does not change, too.

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 $M_q \neq M_z$, 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 $[C_k^g + \delta, b_{g+1}]$ 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.

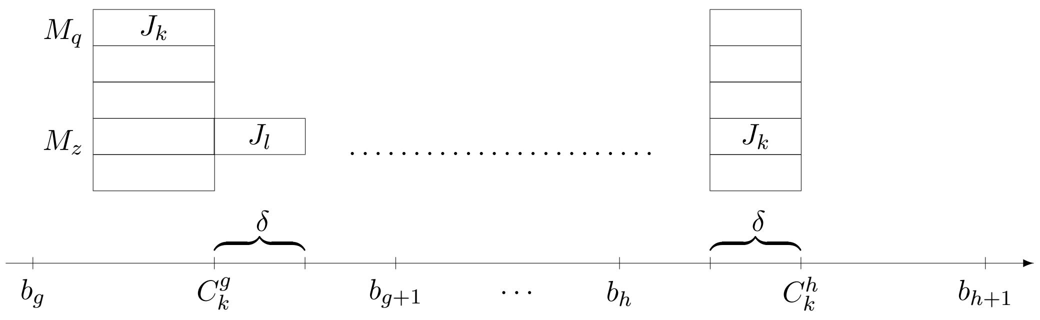

Figure 3: Swap of J_k from $[C_k^h - \delta, C_k^h]$ and J_l from $[C_k^g, C_k^g + \delta]$.

Case 2: $C_{k-1}^h = C_k^h$ and $r_{k-1} \geq b_{g+1}$ hold. Since $C_k^h = \tilde{C}_k$, equality $C_{k-1}^h = \tilde{C}_{k-1}$ holds. Apply the same transformation as it is described in case 1. After the swapping, we have

$$\tilde{C}_k = \max\{\tilde{C}_{k-1}, r_k + (C_k^{v(k)} - b_{v(k)}) + \dots$$

$$+(C_k^g + \delta - b_g) + (C_k^{g+1} - b_{g+1}) + \dots + (C_k^h - b_h)\}.$$

However, since

$$r_k + (C_k^{v(k)} - b_{v(k)}) + \dots + (C_k^g + \delta - b_g) + (C_k^{g+1} - b_{g+1}) + \dots + (C_k^h - b_h) < \tilde{C}_{k-1}$$

holds, we obtain that after the swapping $\tilde{C}_k = \tilde{C}_{k-1}$ holds.

Thus, any swapping does not change the value of function F. Therefore, the schedule can be transformed in such a way that for any two intervals $[b_g, b_{g+1}]$ and $[b_h, b_{h+1}]$ with $b_g \leq C_k^g < b_{g+1} \leq b_h < C_k^h \leq b_{h+1}$, equality $C_{k-1}^h = C_k^h$ holds.

Thus, we obtain schedule $\tilde{s}$ and the set of values C_k^i. For each $k = 1, \dots, n$, the values C_k^i either satisfy equalities (2.1), or they do not satisfy equalities (2.1). If the values C_k^i satisfy equalities (2.1), then

$$C_k(\tilde{s}) = r_k + \sum_{i=v(k)}^{u(k)} (C_k^i - b_i) = \tilde{C}_k$$

holds. However, if the values C_k^i do not satisfy equalities (2.1), then

$$C_k(\tilde{s}) = C_{k-1}(\tilde{s}) = \tilde{C}_{k-1} = \tilde{C}_k$$

holds.

Thus,

$$F(C_1(s), \dots, C_n(s)) = F(\tilde{C}_1, \dots, \tilde{C}_n)$$

holds. □

Since the described transformation does not change the value $(C_j^{v(j)} - b_{v(j)}) + \cdots + (C_j^{u(j)} - b_{u(j)})$ for each job J_j, we do not need to apply the transformation. As a result of solving problem (2.2)-(2.12), we obtain the values

$$\tilde{C}_1 = r_1 + (C_1^{v(1)} - b_{v(1)}) + \cdots + (C_1^{u(1)} - b_{u(1)}),$$

$$\tilde{C}_2 = \max\{\tilde{C}_1, r_2 + (C_2^{v(2)} - b_{v(2)}) + \cdots + (C_2^{u(2)} - b_{u(2)})\},$$

$$\ldots,$$

$$\tilde{C}_n = \max\{\tilde{C}_{n-1}, r_n + (C_n^{v(n)} - b_{v(n)}) + \cdots + (C_n^{u(n)} - b_{u(n)})\},$$

and we can reconstruct an optimal schedule using the known values $C_1 = \tilde{C}_1, \ldots, C_n = \tilde{C}_n$ by solving the corresponding network flow problem, see pages 255–256 in [9] and [10].

Thus, to solve problem $Q \mid r_j, \text{pmtn}, D_j, C_1 \le \ldots \le C_n \mid F$, one has to do the following:

1. Solve the corresponding problem (2.2) - (2.12).
2. Using the values $\tilde{C}_j$, reconstruct an optimal schedule by solving the corresponding network flow problem.

3 Problem $Q \mid r_j$, pmtn, $D_j, C_1 \le \ldots \le C_n \mid \sum w_j T_j$

In [5], it has been proven that problem $Q \mid r_j, p_j = p$, pmtn $\mid \sum w_j T_j$ is NP-hard in the strong sense. Using the results presented in the previous section, we can derive a polynomial algorithm for problem $Q \mid r_j, p_j = p, \text{pmtn} \mid \sum w_j T_j$ with a fixed order of the completion times. Throughout this section, we suppose that the jobs are numbered in such a way that $C_1 \le \cdots \le C_n$ holds. Thus, we have to find an optimal schedule among the class of schedules for which $C_1 \le \ldots \le C_n$ holds.

Remind that $b_1 < \ldots < b_z$ is the set of release times, due dates and deadlines, i.e., $\{b_1, \ldots, b_z\} = \{r_1, \ldots, r_n\} \cup \{d_1, \ldots, d_n\} \cup \{D_1, \ldots, D_n\}$.

For each $j \in \{1, \ldots, n\}$, denote by $v(j)$ the index such that $b_{v(j)} = d_j$ and by $u(j)$ the index such that $b_{u(j)+1} = D_j$.

We apply the mathematical programming model (2.2)-(2.12) to problem $Q \mid r_j$, pmtn, D_j, $C_1 \le \ldots \le C_n \mid \sum w_j T_j$. This yields:

Minimize

$$\sum_{j=1}^{n} w_j \max\{0, \tilde{C}_j - d_j\} \tag{3.1}$$

subject to

$$b_i = C_0^i \le C_1^i \le \cdots \le C_n^i \le C_{n+1}^i = b_{i+1}, \qquad i = 0, \ldots, z \tag{3.2}$$

$$C_j^i = C_{j-1}^i \quad \text{if} \quad [b_i, b_{i+1}] \not\subseteq [r_j, D_j], \qquad i = 0, \ldots, z, \quad j = 1, \ldots, n \tag{3.3}$$

$$\sum_{q=1}^{m} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \leq C_{k+1}^i - C_k^i, \qquad i = 0, \ldots, z, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n \tag{3.4}$$

$$\sum_{j=1}^{n} \frac{v_j^q([C_k^i, C_{k+1}^i])}{s_q} \leq C_{k+1}^i - C_k^i, \qquad i = 0, \ldots, z, \quad q = 1, \ldots, m, \quad k = 1, \ldots, n \tag{3.5}$$

$$\sum_{i=0}^{z} \sum_{k=0}^{n} \sum_{q=1}^{m} v_j^q([C_k^i, C_{k+1}^i]) = p_j, \qquad j = 1, \ldots, n \tag{3.6}$$

$$v_j^q([C_k^i, C_{k+1}^i]) = 0 \qquad \text{if} \qquad \begin{array}{l} [b_i, b_{i+1}] \not\subseteq [r_j, D_j] \text{ or } j \leq k, \\ i = 0, \ldots, z, \quad j = 1, \ldots, n, \\ k = 0, \ldots, n, \quad q = 1, \ldots, m \end{array} \tag{3.7}$$

$$\tilde{C}_j = \max\{\tilde{C}_{j-1}, r_j + \sum_{i=v(j)}^{u(j)} (C_j^i - b_i)\}, \qquad j = 1, \ldots, n, \quad \tilde{C}_0 = 0 \tag{3.8}$$

$$C_k^i \geq 0, \qquad i = 0, \ldots, z, \quad k = 0, \ldots, n \tag{3.9}$$

$$v_j^q([C_k^i, C_{k+1}^i]) \geq 0, \qquad \begin{array}{l} i = 0, \ldots, z, \quad j = 1, \ldots, n, \\ k = 0, \ldots, n, \quad q = 1, \ldots, m \end{array} \tag{3.10}$$

$$\tilde{C}_j \geq 0, \qquad j = 1, \ldots, n, \quad \tilde{C}_0 = 0 \tag{3.11}$$

Since $w_j T_j = w_j \max\{0, C_j - d_j\}$ holds by definition, we obtain that

$$w_j T_j = \begin{cases} 0 & \text{if} \quad r_j < C_j(s) < d_j \\ w_j(C_j(s) - d_j) & \text{if} \quad d_j \leq C_j(s) \leq D_j \end{cases}$$

for each $j = 1, \ldots, n$.

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

$$C_j^i = \begin{cases} C_j(s) & \text{if} \quad b_i < C_j(s) < b_{i+1} \\ b_i & \text{if} \quad C_j(s) \leq b_i \\ b_{i+1} & \text{if} \quad C_j(s) \geq b_{i+1} \end{cases} \tag{3.12}$$

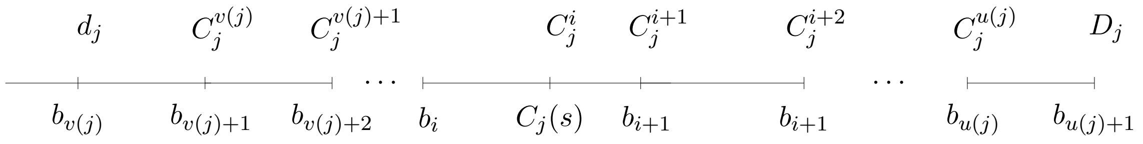

Figure 4: The structure of the interval $[d_j, D_j]$.

for each $j = 1, \dots, n$, and $i = 0, \dots, z$ such that $[b_i, b_{i+1}] \subseteq [d_j, D_j]$, see Figure 4. Using equalities (3.12), we can calculate the value w_jT_j by the formula

$$w_jT_j = w_j(C_j^{v(j)} - b_{v(j)}) + w_j(C_j^{v(j)+1} - b_{v(j)+1}) + \dots + w_j(C_j^{u(j)} - b_{u(j)}).$$

Indeed, let $C_j(s) \in [b_i, b_{i+1}]$, then using (3.12), we obtain

$$\begin{aligned} w_jT_j &= w_j(C_j^{v(j)} - b_{v(j)}) + w_j(C_j^{v(j)+1} - b_{v(j)+1}) + \dots + w_j(C_j^{u(j)} - b_{u(j)}) \\ &= w_j(b_{v(j)+1} - b_{v(j)}) + w_j(b_{v(j)+2} - b_{v(j)+1}) \\ &\quad + w_j(b_{v(j)+3} - b_{v(j)+2}) + \dots + w_j(C_j^i - b_i) \\ &\quad + w_j(b_i - b_i) + w_j(b_{i+1} - b_{i+1}) + \dots + w_j(b_{u(j)} - b_{u(j)}) \\ &= w_jC_j^i - w_jb_{v(j)} \\ &= w_j(C_j - d_j). \end{aligned}$$

Thus, to minimize $\sum_{j=1}^n w_jT_j$, we need to minimize $\sum_{j=1}^n w_j \max\{0, \tilde{C}_j - d_j\}$. So, problem $Q \mid r_j,\ \text{pmtn}, D_j,\ C_1 \leq \dots \leq C_n \mid \sum w_jT_j$ can be reduced to a linear programming problem and therefore, it can be polynomially solved.

4 Concluding Remarks

In this paper, we have considered scheduling problems with ordered completion times. We have shown that a wide class of scheduling problems with preemptions can be polynomially solved if the order of the completion times is known in advance. Note that in contrast to [6, 7], here we did not restrict to equal processing times but considered the general case of arbitrary processing times.

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] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 1979; 5: 287 - 326.

[2] Horn W.A. Minimizing average flow time with parallel machines. Oper. Res. 1973; 21: 846–847.

[3] Bruno J.L., Coffman E.J., Sethi R. Scheduling independent tasks to reduce mean finishing time. Comm. ACM 1974; 17: 382–387.

[4] Lawler E.L., Labetoulle J. On preemptive scheduling of unrelated parallel processors by linear programming. J. Assoc. Comput. Mach. 1978; 25: 612–619.

[5] Du J., Leung J. Y.-T. Minimizing mean flow time with release time constraint. Theoretical Computer Science 1990; 75: 347–355.

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

[7] Kravchenko S.A., Werner F. Minimizing a separable convex function on parallel machines with preemptions. Otto-von-Guericke-Universität Magdeburg. Fakultät für Mathematik, Preprint 22/09, 2009, 20 p.

[8] Kravchenko S.A., Werner F. Parallel machine problems with equal processing times. Proceedings of the 4th MISTA conference, Dublin/Ireland, 2009, 458 – 468.

[9] 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.

[10] Lawler E.L. Combinatorial optimization: Networks and matroids. New York: Holt, Rinehart and Winston, 1976.

[11] 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, 202–234.