A Parameterized Algorithm for the Preemptive Scheduling of Equal-Length Jobs

George B. Mertzios
Department of Computer Science
RWTH Aachen University
52056 Aachen, Germany
Email: mertzios@cs.rwth-aachen.de

Walter Unger
Department of Computer Science
RWTH Aachen University
52056 Aachen, Germany
Email: quax@cs.rwth-aachen.de

Abstract

We study the preemptive scheduling problem of a set of n jobs with release times and equal processing times on a single machine. The objective is to minimize the sum of the weighted completion times $\sum_{i=1}^{n} w_i C_i$ of the jobs. We propose for this problem the first parameterized algorithm on the number k of different weights. The runtime of the proposed algorithm is $O((\frac{n}{k}+1)^k n^8)$ and hence, this is the first polynomial algorithm for any fixed number k of different weights.

1. Introduction

In this paper we consider the preemptive scheduling of n jobs $J_1, J_2, \ldots, J_n$ with equal processing time p on a single machine. Here, *preemption* means job splitting, i.e. the execution of a job J_i may be interrupted for the execution of another job J_j, while the execution of J_i will be resumed later on [8]. Every job J_i has a release time r_i, after which J_i is available and a positive weight $w_i \in \{\alpha_j\}_{j=1}^{k}$. A schedule of these jobs is called *feasible*, if every job J_i starts not earlier than its release time r_i. The objective is to find a feasible schedule of these jobs that minimizes the weighted sum $\sum_{i=1}^{n} w_i C_i$, where C_i is the completion time of job J_i.

The preemptive scheduling has attracted many research efforts. Several problems, which are NP-hard in the general case, admit polynomial algorithms under the assumption of equal-length jobs. In particular, the problem of minimizing the sum of completion times on identical parallel machines is polynomially solvable for equal-length jobs [5], [13], while it is unary NP-hard for arbitrary processing times [5]. The problem of maximizing the weighted throughput, or equivalently of minimizing the weighted number of late jobs on a single machine, is NP-hard [11] and pseudo-polynomially solvable [15] in the general case. On the contrary, its restriction to equal-length jobs is solvable in polynomial time in the preemptive, as well as in the non-preemptive case [3], [6]. For the problem of minimizing the total tardiness there is also a polynomial algorithm for equal-length jobs [19]. Furthermore, minimizing the sum of completion times [1] or the number of late jobs [2], [15] on a single machine can be done in polynomial time also for arbitrary processing times. More detailed complexity results on machine scheduling can be found in [8], [9].

In the non-preemptive case, the problems of minimizing the number of late jobs on a single machine [12] and minimizing the sum of the completion times on identical parallel machines [18] are polynomial for equal-length jobs, while the corresponding problems in the general case are both NP-hard, also on a single machine [11], [16]. Moreover, polynomial time algorithms have been presented in [10] for the case of equal-length jobs on uniform parallel machines.

The complexity status of the problem we focus on in this paper has been stated as an open question for equal-length jobs and arbitrary weights on a single machine [4], [5], [7], [9]. This problem is known to be NP-hard, if the processing times are arbitrary on a single machine [14], or even for equal processing times on identical parallel machines [17]. We propose the first polynomial algorithm for arbitrary release times r_i, which is parameterized on the number k of different weights w_i. The runtime of the proposed algorithm is $O((\frac{n}{k}+1)^k n^8)$, while its space complexity is $O((\frac{n}{k}+1)^k n^6)$.

Several real-time applications of this problem can be found. In the context of service management,

vehicles may arrive in predefined appointments for regular check. This process is preemptive, while the service time of each vehicle is the same. In addition, special purpose vehicles, such as ambulances, have higher priority than others. In the context of logistics, products that need special conditions, such as humidity and temperature, have to be stored with higher priority than other products.

In Section 2 we provide some properties of an optimal schedule, in order to determine the possible start and completion times of the jobs. By using these results, we construct a polynomial dynamic programming algorithm in Section 3. Finally, some conclusions and open questions are discussed in Section 4.

2. Properties of an optimal schedule

In this section we provide some properties of an optimal preemptive schedule $\mathcal{S}$, in order to determine the set of all possible start and completion times of the n jobs in $\mathcal{S}$. For every job J_i let r_i be its release time and C_i be its completion time in $\mathcal{S}$. As a first step, we prove the technical Lemma 1 that will be used several times in the remaining part of the article.

Lemma 1: For every job J_i that is at least partially executed in an optimal schedule $\mathcal{S}$ in the time interval $[r_k, C_k)$, it holds $C_i < C_k$.

Proof: The proof will be done by contradiction. Suppose that job J_i is partially executed in at least one time interval $I \subset [r_k, C_k)$ and that $C_i > C_k$, as it is illustrated in Figure 1. Since J_k is completed at time C_k in $\mathcal{S}$, there is a sufficient small positive $\varepsilon \leq |I|$, such that J_k is executed during the interval $[C_k - \varepsilon, C_k)$. We can exchange now a part of length ε of the interval I with the interval $[C_k - \varepsilon, C_k)$. In this modified schedule $\mathcal{S}'$, the completion time of J_k becomes at most $C_k - \varepsilon$, while the completion times of all other jobs remain the same. This is a contradiction to the assumption that $\mathcal{S}$ is optimal. It follows that $C_i < C_k$. □

The following Lemma 2 restricts the possible values of the makespan $C_{\max}$ of any optimal schedule, i.e. the completion time of the last completed job.

Lemma 2: The makespan $C_{\max}$ in an optimal schedule $\mathcal{S}$ equals

$$C_{\max} = r_i + \ell p \tag{1}$$

for some $i, \ell \in \{1, 2, \ldots, n\}$.

Proof: Let t be the end of the last idle period in $\mathcal{S}$, i.e. the machine is working continuously between t and $C_{\max}$. Let also that job J_i is executed directly after t, for some $i \in \{1, 2, \ldots, n\}$. Then, t equals the release time r_i of J_i, since otherwise J_i could be scheduled to complete earlier, resulting thus to a better schedule, which is a contradiction. Furthermore, every job J_k that is at least partially executed after t, has release time $r_k \geq t$, since otherwise J_k could be scheduled to complete earlier, which is again a contradiction. Thus, since the machine is working continuously between t and $C_{\max}$, it holds that $C_{\max} = r_i + \ell p$, where $1 \leq \ell \leq n$ is the number of jobs executed in the interval $[t, C_{\max})$. □

Now, Lemma 3 determines the possible start and completion times of the jobs $J_1, J_2, \ldots, J_n$ in $\mathcal{S}$.

Lemma 3: The start and completion times of the jobs in an optimal schedule $\mathcal{S}$ take values from the set

$$T := \{r_i + \ell p : 1 \leq i \leq n, 0 \leq \ell \leq n\} \tag{2}$$

Proof: Consider an arbitrary job J_k and let $\mathcal{J} = \{J_i : C_i \leq C_k\}$ be the set of all jobs that are completed not later than J_k in $\mathcal{S}$. Consider now a job $J_m \notin \mathcal{J}$. Then, Lemma 1 implies that no part of J_m is executed at all in any time interval $[r_i, C_i)$, where $J_i \in \mathcal{J}$, since otherwise it would be $C_m < C_i \leq C_k$, i.e. $J_m \in \mathcal{J}$, which is a contradiction. It follows that the completion time C_k of job J_k remains the same if we remove from schedule $\mathcal{S}$ all jobs $J_m \notin \mathcal{J}$.

Thus, it holds due to Lemma 2 that $C_k = r_i + \ell p$, for some $J_i \in \mathcal{J}$ and $\ell \in \{1, 2, \ldots, |\mathcal{J}|\}$. Since $|\mathcal{J}| \leq n$, it follows that for the completion time of an arbitrary job J_k it holds $C_k \in T$. Furthermore, due to the optimality of $\mathcal{S}$, an arbitrary job J_i starts either at its release time r_i, or at the completion time C_k of another job J_k. Thus, all start points of the jobs belong to T as well. □

3. The dynamic programming algorithm

3.1. Definitions and boundary conditions

In this section we propose a polynomial dynamic programming algorithm that computes the value of an optimal preemptive schedule on a single machine, where the weights of the jobs take k possible values $\{\alpha_i : 1 \leq i \leq k\}$, with $\alpha_1 > \ldots > \alpha_k > 0$. We partition the jobs into k sets $\mathcal{J}^i = \{J^i_1, J^i_2, \ldots, J^i_{n_i}\}$, $i \in \{1, \ldots, k\}$, such that job J^i_ℓ has weight α_i for every $\ell \in \{1, \ldots, n_i\}$. Assume without loss of generality that for every i, the jobs J^i_ℓ are sorted with respect to ℓ in non-decreasing order according to their release times r^i_ℓ, i.e.

$$r^i_1 \leq r^i_2 \leq \ldots \leq r^i_{n_i} \tag{3}$$

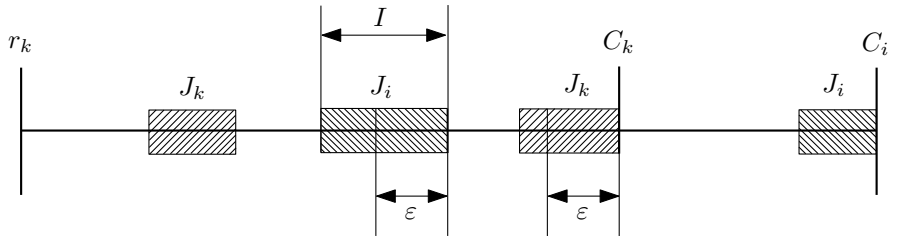

Figure 1: The impossible case $C_i > C_k$, where job J_i is partially executed in $[r_k, C_k)$.

Denote now by

$$\mathbf{t} = (t_k, t_{k-1}, \ldots, t_1) \tag{4}$$

a vector $\mathbf{t} \in \mathbb{N}_0^k$, where for its coordinates it holds $0 \leq t_i \leq n_i$ for every $i \in \{1, \ldots, k\}$. Let $\mathcal{P}(\mathbf{t}) = \{i : t_i > 0, 1 \leq i \leq k\}$ be the set of indices that corresponds to strictly positive coordinates of $\mathbf{t}$. For every vector $\mathbf{t} \neq \mathbf{0} = (0, \ldots, 0)$ and every $i \in \mathcal{P}(\mathbf{t})$ define the vectors

$$\mathbf{t}'_i = (t_k, \ldots, t_{i+1}, t_i - 1, t_{i-1}, \ldots, t_1) \tag{5}$$

$$\mathbf{t}''_i = (0, \ldots, 0, t_i, t_{i-1}, \ldots, t_1) \tag{6}$$

and let

$$\mathbf{t}_{\max} = \max \mathcal{P}(\mathbf{t}) \tag{7}$$

be the maximum index i, for which $t_i > 0$. Denote by

$$Q(\mathbf{t}, x, y, z) \tag{8}$$

where $\mathbf{t} \neq \mathbf{0}$ and $x \leq y < z$, the set of all jobs among $\bigcup_{i \in \mathcal{P}(\mathbf{t})} \bigcup_{\ell=1}^{t_i} J_\ell^i$ that have release times

$$r_\ell^i \in \begin{cases} [x, z), & \text{if } i = \mathbf{t}_{\max} \text{ and } \ell = t_i \\ [y, z), & \text{otherwise} \end{cases} \tag{9}$$

We define for $\mathbf{t} = \mathbf{0}$

$$Q(\mathbf{0}, x, y, z) = \emptyset \tag{10}$$

for all values $x \leq y < z$. Denote now by

$$F(\mathbf{t}, x, y, z) \tag{11}$$

the value of an optimal schedule of all jobs of the set $Q(\mathbf{t}, x, y, z)$ in the interval $[y, z)$. Due to Lemma 3, we allow the variables y, z in (8) and (11) to take values only from the set T. Also, due to (9), since every job is released not earlier than x, it suffices to consider that $x \in \{r_\ell^i : 1 \leq i \leq k, 1 \leq \ell \leq n_i\}$. For an arbitrary $y \in T$, let

$$r(y) = \min\{r_\ell^i : r_\ell^i \geq y, 1 \leq i \leq k, 1 \leq \ell \leq n_i\} \tag{12}$$

be the smallest release time that equals at least y. In the case $Q(\mathbf{t}, x, y, z) = \emptyset$, we define $F(\mathbf{t}, x, y, z) = 0$.

Definition 1: The set $Q(\mathbf{t}, x, y, z)$ of jobs is called *feasible*, if there exists a feasible schedule of these jobs in the interval $[y, z)$. If $Q(\mathbf{t}, x, y, z)$ is *not* feasible, then we define $F(\mathbf{t}, x, y, z) = \infty$.

The following lemma uses the release times of the jobs of $Q(\mathbf{t}, x, y, z)$ in order to decide whether it is feasible, i.e. whether there exists a feasible schedule of these jobs in the interval $[y, z)$.

Lemma 4 (feasibility test): Let $\widetilde{r}_1 \leq \widetilde{r}_2 \leq \ldots \leq \widetilde{r}_q$ be the release times of the jobs of $Q(\mathbf{t}, x, y, z)$ and let

$$\begin{aligned} C_1 &= \max\{\widetilde{r}_1, y\} + p \\ C_\ell &= \max\{\widetilde{r}_\ell, C_{\ell-1}\} + p \end{aligned} \tag{13}$$

for every $\ell \in \{2, 3, \ldots, q\}$. It holds that $Q(\mathbf{t}, x, y, z)$ is feasible if and only if $C_q \leq z$.

Proof: The proof is straightforward. The set $Q(\mathbf{t}, x, y, z)$ of jobs is feasible if and only if there exists a schedule of these jobs with makespan $C_{\max}$ not greater than z. Without loss of generality, in a schedule that minimizes $C_{\max}$, every job is scheduled without preemption at the earliest possible point. In particular, the job with the earliest release time $\widetilde{r}_1$ starts at $\max\{\widetilde{r}_1, y\}$. Suppose that the $\ell - 1$ first jobs complete at point $C_{\ell-1}$, for some $\ell \in \{2, 3, \ldots, q\}$. If the ℓ^{th} job has release time $\widetilde{r}_\ell > C_{\ell-1}$, then this job starts obviously at $\widetilde{r}_\ell$. In the opposite case $\widetilde{r}_\ell \leq C_{\ell-1}$, it starts at $C_{\ell-1}$. Since every job has processing time p, we obtain (13) for the completion times of the scheduled jobs and thus the minimum makespan is C_q. It follows that $Q(\mathbf{t}, x, y, z)$ is feasible, i.e. $F(\mathbf{t}, x, y, z) \neq \infty$, if and only if $C_q \leq z$. □

3.2. The recursive computation

For every index $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$ of a vector $\mathbf{t} \neq \mathbf{0}$ and any possible values x, y, z, if $r_{t_i}^i \notin [y, z)$, it holds that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, x, y, z) \tag{14}$$

while for $i = \mathbf{t}_{\max}$, if $r_{t_i}^i \notin [x, z)$, it holds that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, r(y), r(y), z) \tag{15}$$

Indeed, in both cases it holds $J^i_{t_i} \notin Q(\mathbf{t}, x, y, z)$ due to (9) and thus we may remove job $J^i_{t_i}$ from the schedule, i.e. we replace t_i by $t_i - 1$. In the first case $i \neq \mathbf{t}_{\max}$, all the remaining jobs have release times according to (9) and they are scheduled in the interval $[y, z)$. In the second case $i = \mathbf{t}_{\max}$, all the remaining jobs are released not earlier than y, i.e. not earlier than $r(y)$ and thus they are all scheduled in the interval $[r(y), z)$. Therefore, suppose in the following without loss of generality that $J^i_{t_i} \in Q(\mathbf{t}, x, y, z)$ for every $i \in \mathcal{P}(\mathbf{t})$.

Let C^i_ℓ denote the completion time of job J^i_ℓ, where $i \in \{1, \ldots, k\}$ and $\ell \in \{1, \ldots, n_i\}$. Denote in the following by $\mathcal{S}$ the optimal schedule that lexicographically minimizes the vector of the completion times $(C^1_1, C^1_2, \ldots, C^k_{n_k})$ among all other optimal schedules. Next, we compute in the main Theorem 1 the values $F(\mathbf{t}, x, y, z)$. To this end, we provide first the technical Lemma 5 and Corollary 1 that will be used in the proof of the theorem. Denote by s_i and e_i the start and completion time of job $J^i_{t_i}$ in $\mathcal{S}$ respectively. Also, for $i = \mathbf{t}_{\max}$, denote for simplicity $J^i_{t_i}$ and α_i by $J_{\mathbf{t}_{\max}}$ and $\alpha_{\mathbf{t}_{\max}}$, respectively.

Lemma 5: Suppose that $J^i_{t_i} \in Q(\mathbf{t}, x, y, z)$ for some $i \in \mathcal{P}(\mathbf{t})$. For every other job $J^j_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ with $j \leq i$, if J^j_ℓ is completed in $\mathcal{S}$ at a point $C^j_\ell > s_i$, then its release time is $r^j_\ell > s_i$.

Proof: The proof will be done by contradiction. Consider a job $J^j_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ with $j \leq i$ and suppose that J^j_ℓ is completed in $\mathcal{S}$ at a point $C^j_\ell > s_i$. We distinguish the cases $C^j_\ell > C^i_{t_i}$ and $C^j_\ell < C^i_{t_i}$, respectively.

Suppose that $C^j_\ell > C^i_{t_i}$ and that J^j_ℓ is executed in $[C^i_{t_i}, z)$ for a time period of total length $L \leq p$, as it is illustrated in Figure 2(a). If $r^j_\ell \leq s_i$, then we can exchange the execution of J^j_ℓ in the interval $[C^i_{t_i}, z)$ with the last part of total length L of the execution of $J^i_{t_i}$ in the interval $[s_i, C^i_{t_i})$. In the resulting schedule $\mathcal{S}'$, the completion times C^j_ℓ and $C^i_{t_i}$ exchange values, while the completion times of all other jobs remain the same. Since $j \leq i$, it holds $\alpha_j \geq \alpha_i$ and therefore the schedule $\mathcal{S}'$ is not worse than $\mathcal{S}$. Thus, since $\mathcal{S}$ is optimal, $\mathcal{S}'$ is also optimal. However, $\mathcal{S}'$ is lexicographically smaller than $\mathcal{S}$, which is a contradiction to the assumption on $\mathcal{S}$. It follows that job J^j_ℓ is released not earlier than s_i, i.e. $r^j_\ell > s_i$.

Suppose now that $C^j_\ell < C^i_{t_i}$, as it is illustrated in Figure 2(b). Then, there exists a sufficiently small time period $\varepsilon > 0$, such that during the time intervals $[s_i, s_i + \varepsilon)$ and $[C^j_\ell - \varepsilon, C^j_\ell)$ the jobs $J^i_{t_i}$ and J^j_ℓ are executed, respectively. If $r^j_\ell \leq s_i$, we can now exchange the execution of the jobs $J^i_{t_i}$ and J^j_ℓ in these intervals, obtaining a completion time of J^j_ℓ at most $C^j_\ell - \varepsilon$, while the completion times of all other jobs remain the same. Since all weights are positive, the resulting schedule is better than $\mathcal{S}$, which is a contradiction. This implies again that job J^j_ℓ is released not earlier than s_i, i.e. $r^j_\ell > s_i$. □

Corollary 1: Suppose that $J^i_{t_i} \in Q(\mathbf{t}, x, y, z)$ for some $i \in \mathcal{P}(\mathbf{t})$. Then, every other job $J^i_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ is completed in $\mathcal{S}$ at a point $C^i_\ell \leq s_i$.

Proof: Consider such a job J^i_ℓ, with $\ell < t_i$ and suppose that J^i_ℓ is completed at a point $C^i_\ell > s_i$. Then, Lemma 5 implies that $r^i_\ell > s_i$. On the other side, it holds due to (3) that $r^i_\ell \leq r^i_{t_i} \leq s_i$, which is a contradiction. □

Theorem 1: Suppose that $J^i_{t_i} \in Q(\mathbf{t}, x, y, z)$ for every $i \in \mathcal{P}(\mathbf{t})$. Let $e = y + p \cdot |Q(\mathbf{t}, x, y, z)|$. If $Q_k(x, y, z)$ is feasible, it holds that

$$F(\mathbf{t}, x, y, z) = \min_{\substack{s \in (y,z) \cap T \\ i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}}} \left\{ \begin{array}{l} F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s) \\ \qquad + F(\mathbf{t}, x, s, z), \\ F(\mathbf{t}'_i, x, y, s) \\ \qquad + F(\mathbf{t}''_i, r(y), s, z), \\ F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e) \\ \qquad + e \cdot \alpha_{\mathbf{t}_{\max}} \end{array} \right\} \quad (16)$$

Proof: Let $\mathcal{S}$ be the optimal schedule that lexicographically minimizes the vector of the completion times $(C^1_1, C^1_2, \ldots, C^k_{n_k})$ among all other optimal schedules. Let also job $J^i_{t_i} \in Q(\mathbf{t}, x, y, z)$ start at point s_i and complete at point e_i in $\mathcal{S}$, for every $i \in \mathcal{P}(\mathbf{t})$. We distinguish in the following the cases $s_{\mathbf{t}_{\max}} > y$ and $s_{\mathbf{t}_{\max}} = y$.

Case $s_{\mathbf{t}_{\max}} > y$. For every job $J^j_\ell \in Q(\mathbf{t}, x, y, z)$ it holds $j \leq \mathbf{t}_{\max}$, due to (7). Thus, Lemma 5 implies that all jobs $J^j_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ with $r^j_\ell \leq s_{\mathbf{t}_{\max}}$ are scheduled completely in the interval $[y, s_{\mathbf{t}_{\max}})$, while all jobs J^j_ℓ with release times $r^j_\ell > s_{\mathbf{t}_{\max}}$ are clearly scheduled in $\mathcal{S}$ completely in the interval $[s_{\mathbf{t}_{\max}}, z)$. Note that the extreme case $r^j_\ell = s_{\mathbf{t}_{\max}}$ is impossible, since otherwise job J^j_ℓ must be scheduled in the empty interval $[s_{\mathbf{t}_{\max}}, s_{\mathbf{t}_{\max}})$, which is a contradiction.

In the first part $[y, s_{\mathbf{t}_{\max}})$ of $\mathcal{S}$, only jobs of the set $Q(\mathbf{t}'_{\mathbf{t}_{\max}}, x, y, z) = Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ may be scheduled. These jobs are assumed to be released not earlier than y, i.e. not earlier than $r(y)$ and thus the value of this first part of $\mathcal{S}$ equals

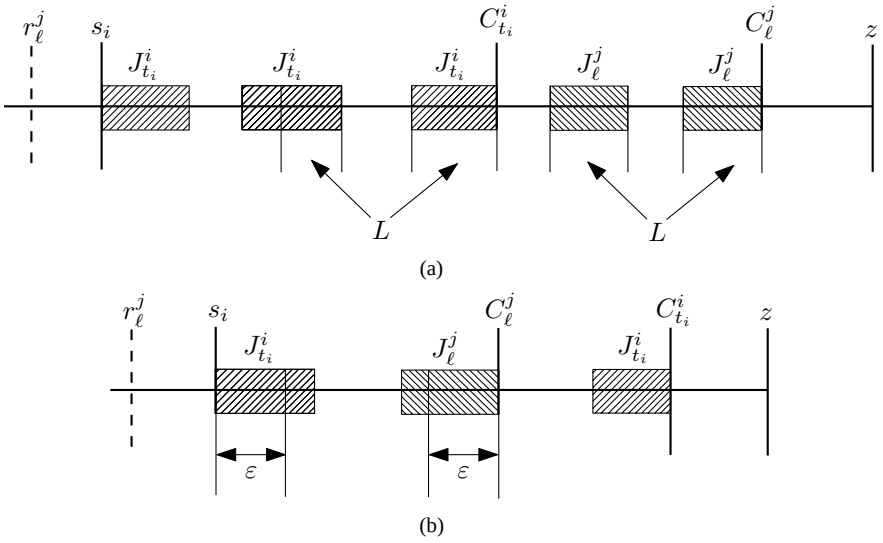

Figure 2: The impossible case $r_\ell^j \leq s_i$, where $j \leq i$ and $C_\ell^j > s_i$.

$F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}})$. On the other side, in the second part $[s_{\mathbf{t}_{\max}}, z)$ of $\mathcal{S}$, each job of $Q(\mathbf{t}, x, y, z)$ with release time greater than $s_{\mathbf{t}_{\max}}$ may be scheduled. Since $J_{\mathbf{t}_{\max}}$ is released not earlier than x, the value of the second part of $\mathcal{S}$ equals $F(\mathbf{t}, x, s_{\mathbf{t}_{\max}}, z)$. It follows that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}}) + F(\mathbf{t}, x, s_{\mathbf{t}_{\max}}, z) \quad (17)$$

Case $s_{\mathbf{t}_{\max}} = y$. Let i be the greatest index of $\mathcal{P}(\mathbf{t})$, such that $s_i \geq e_{\mathbf{t}_{\max}}$, if one exists. That is, for every index $j \in \mathcal{P}(\mathbf{t})$ with $j > i$, job $J_{t_j}^j$ starts at a point $s_j \in [s_{\mathbf{t}_{\max}}, e_{\mathbf{t}_{\max}})$, as it is illustrated in Figure 3(a). Then, Lemma 1 implies that this job completes also in this interval, i.e. $e_j \in [s_{\mathbf{t}_{\max}}, e_{\mathbf{t}_{\max}})$. Furthermore, Corollary 1 implies that for every such index $j \in \mathcal{P}(\mathbf{t})$ with $j > i$, all jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{t_j}^j\}$ are completed at a point $C_\ell^j \leq s_j$. Then, since $s_j < s_i$, we obtain that $C_\ell^j < s_i$. It follows that for every job J_ℓ^j that is completed at a point $C_\ell^j > s_i$, it holds $j \leq i$. Thus, Lemma 5 implies that all jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{t_i}^i\}$ with $r_\ell^j \leq s_i$ are scheduled completely in the interval $[y, s_i)$, while all jobs J_ℓ^j with release times $r_\ell^j > s_i$ are clearly scheduled in $\mathcal{S}$ completely in the interval $[s_i, z)$. Note that the extreme case $r_\ell^j = s_i$ is impossible, since otherwise job J_ℓ^j must be scheduled in the empty interval $[s_i, s_i)$, which is a contradiction.

In the first part $[y, s_i)$ of $\mathcal{S}$, only jobs of the set $Q(\mathbf{t}'_i, x, y, z) = Q(\mathbf{t}, x, y, z) \setminus \{J_{t_i}^i\}$ may be scheduled. Since $J_{\mathbf{t}_{\max}}$ is released not earlier than x, the value of this first part of $\mathcal{S}$ equals $F(\mathbf{t}'_i, x, y, s_i)$. In the second part $[s_i, z)$ of $\mathcal{S}$, only jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z)$ with $j \leq i < \mathbf{t}_{\max}$ and release time greater than s_i may be scheduled. Since all these jobs are assumed to be released not earlier than y, i.e. not earlier than $r(y)$, the value of the second part of $\mathcal{S}$ equals $F(\mathbf{t}''_i, r(y), s_i, z)$. It follows that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, x, y, s_i) + F(\mathbf{t}''_i, r(y), s_i, z) \quad (18)$$

Consider now the remaining case that for every $i \in \mathcal{P}(\mathbf{t})$ it holds $s_i < e_{\mathbf{t}_{\max}}$. Corollary 1 implies that for every $i \in \mathcal{P}(\mathbf{t})$, all jobs J_ℓ^i with $\ell < t_i$ are completed at most at point s_i. Thus, in this case all jobs of $Q(\mathbf{t}, x, y, z)$ are scheduled completely in the interval $[y, e_{\mathbf{t}_{\max}})$, as it is illustrated in Figure 3(b). Since the processing time of every job equals p, the total processing time of all jobs equals $p \cdot |Q(\mathbf{t}, x, y, z)|$. On the other side, there is no idle period between y and $e_{\mathbf{t}_{\max}}$, since otherwise $J_{\mathbf{t}_{\max}}$ would be scheduled to complete earlier, resulting thus to a better schedule, which is a contradiction to the optimality of $\mathcal{S}$. Therefore, it holds

$$e_{\mathbf{t}_{\max}} = y + p \cdot |Q(\mathbf{t}, x, y, z)| \quad (19)$$

Lemma 1 implies that no part of $J_{\mathbf{t}_{\max}}$ is executed in any time interval $[r_\ell^i, C_\ell^i)$, where $J_\ell^i \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$, since otherwise $J_{\mathbf{t}_{\max}}$ would complete before J_ℓ^i, which is a contradiction. Thus, the completion times of all these jobs remain the same if we remove $J_{\mathbf{t}_{\max}}$ from the schedule $\mathcal{S}$. Since the weight of $J_{\mathbf{t}_{\max}}$ is $\alpha_{\mathbf{t}_{\max}}$ and its completion

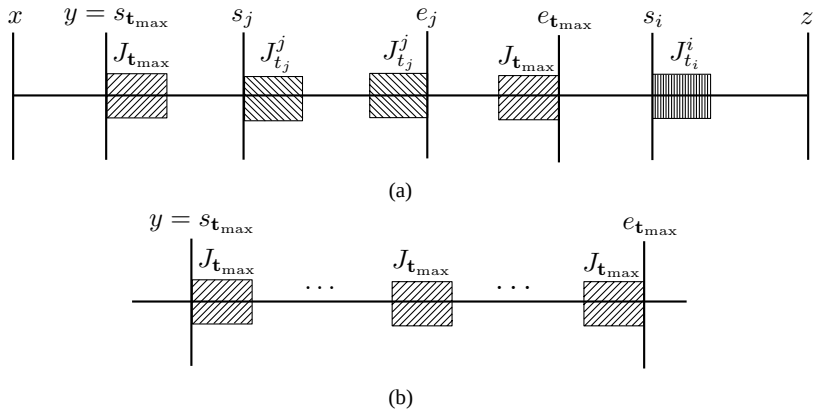

Figure 3: The case $s_{\mathbf{t}_{\max}} = y$.

time is $e_{\mathbf{t}_{\max}}$, it follows in this case that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) + e_{\mathbf{t}_{\max}} \cdot \alpha_{\mathbf{t}_{\max}} \tag{20}$$

Summarizing, since $\mathcal{S}$ is optimal, the value $F(\mathbf{t}, x, y, z)$ is the minimum of the expressions in (17), (18) and (20) over all possible values $s_{\mathbf{t}_{\max}}, s_i \in (y, z) \cap T$ and $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$ respectively. This proves the theorem. □

3.3. The algorithm

Since the start and endpoints of the jobs in an optimal schedule belong to T, the value of such a schedule equals

$$F(\mathbf{t}^*, \min T, \min T, \max T) \tag{21}$$

where

$$\mathbf{t}^* = (n_1, n_2, \ldots, n_k) \tag{22}$$

and $\min T$ coincides with the smallest release time, due to (2). The dynamic programming Algorithm 1 follows now from Lemma 4 and Theorem 1.

Note that, as a preprocessing step, we partition the n jobs into the sets $\mathcal{J}^i = \{J^i_1, J^i_2, \ldots, J^i_{n_i}\}$, $i \in \{1, \ldots, k\}$, such that job J^i_ℓ has weight α_i for every $\ell \in \{1, \ldots, n_i\}$, and that, for every i, the jobs J^i_ℓ are sorted with respect to ℓ according to (3). This can be done clearly in $O(n \log n)$ time. Note furthermore that, for each vector $\mathbf{t} = (t_k, t_{k-1}, \ldots, t_1)$, the set $\mathcal{P}(\mathbf{t}) = \{i : t_i > 0, 1 \le i \le k\}$ and the value $\mathbf{t}_{\max} = \max \mathcal{P}(\mathbf{t})$ can be computed in $O(n)$ time, since $k \le n$.

In the first two lines, Algorithm 1 initializes $F(\mathbf{0}, x, y, z) = 0$ for all possible values of x, y, z. This takes $O(n^5)$ time, since x can take $O(n)$ possible values and y, z can take at most $O(n^2)$ possible values each. It iterates further for every $\mathbf{t}$ between $\mathbf{0}$ and $\mathbf{t}^*$ in lexicographical order and for every possible x, y, z. For every such tuple $(\mathbf{t}, x, y, z)$, the algorithm computes the value $F(\mathbf{t}, x, y, z)$ as follows. At first, it computes the set $Q(\mathbf{t}, x, y, z)$ in line 6. Since for each vector $\mathbf{t}$, the set $\mathcal{P}(\mathbf{t})$ and the value $\mathbf{t}_{\max}$ can be computed in linear $O(n)$ time, the computation of $Q(\mathbf{t}, x, y, z)$ can be done in linear time as well,

Algorithm 1 Compute the value of an optimal schedule with n jobs

```
1:  for each x = r^i_ℓ and y, z ∈ T, with x ≤ y < z do
2:      F(0, x, y, z) ← 0
3:  for every t in lexicographical order do
4:      for every x = r^i_ℓ and z ∈ T with x < z do
5:          for y = z downto x (with y ≠ z) do
6:              if Q(t, x, y, z) = ∅ then
7:                  F(t, x, y, z) ← 0
8:              else if Q(t, x, y, z) is not feasible then
9:                  F(t, x, y, z) ← ∞
10:             else
11:                 for every i ∈ P(t) do
12:                     if i = t_max then
13:                         if r^i_{t_i} ∉ [x, z) then
14:                             F(t, x, y, z) ← F(t'_i, r(y), r(y), z)
15:                     else {i ≠ t_max}
16:                         if r^i_{t_i} ∉ [y, z) then
17:                             F(t, x, y, z) ← F(t'_i, x, y, z)
18:                 if F(t, x, y, z) has not been computed in
                    lines 14 or 17 then
19:                     Compute F(t, x, y, z) by Theorem 1
20: return F(t*, min T, min T, max T)
```

by checking in (9) the release times of the jobs $\bigcup_{i\in\mathcal{P}(\mathbf{t})}\bigcup_{\ell=1}^{t_i} J_\ell^i$. If the set $Q(\mathbf{t},x,y,z)$ is empty, it defines $F(\mathbf{t},x,y,z)=0$. Otherwise, if $Q(\mathbf{t},x,y,z)$ is not empty, the algorithm checks its feasibility in line 8. This can be done in $O(n\log n)$ time using Lemma 4, by sorting first increasingly the release times $\widetilde{r}_1,\widetilde{r}_2,\ldots,\widetilde{r}_q$ of the jobs in $Q(\mathbf{t},x,y,z)$ and then, by computing in linear time the value C_q. If it is not feasible, it defines $F(\mathbf{t},x,y,z)=\infty$ by Definition 1.

In the case of feasibility of the set $Q(\mathbf{t},x,y,z)$, the algorithm checks in linear time the release times of the jobs $J_{t_i}^i$ for all $i\in\mathcal{P}(\mathbf{t})$ in lines 11-17. If at least one of these jobs does not belong to $Q(\mathbf{t},x,y,z)$, it computes $F(\mathbf{t},x,y,z)$ recursively in lines 14 and 17, due to (15) and (14), respectively. Finally, if all jobs $J_{t_i}^i$, $i\in\mathcal{P}(\mathbf{t})$ belong to $Q(\mathbf{t},x,y,z)$, i.e. the value $F(\mathbf{t},x,y,z)$ has not been computed in the lines 14 or 17, the algorithm computes $F(\mathbf{t},x,y,z)$ in line 19 by Theorem 1. For this computation, the algorithm uses for every $s\in(y,z)\cap T$ and every $i\in\mathcal{P}(\mathbf{t})\setminus\{\mathbf{t}_{\max}\}$ the values of one or two smaller instances. This takes time $O(n^3)$, since T has at most n^2 elements and $\mathcal{P}(\mathbf{t})$ has at most n elements. Thus, the algorithm needs for the lines 6-19 time $O(n^3)$.

Note here that, for every $i\in\mathcal{P}(\mathbf{t})$, the vectors $\mathbf{t}_i'$ and $\mathbf{t}_i''$ are lexicographically smaller than $\mathbf{t}$. Thus, the values $F(\mathbf{t}_i',r(y),r(y),z)$ and $F(\mathbf{t}_i',x,y,z)$, which are used in lines 14 and 17, have been already computed at a previous iteration of the algorithm. Furthermore, since we iterate for y in line 5 from the value z downwards to the value x, the values $F(\mathbf{t},x,s,z)$, for every $s\in T$ with $y<s<z$, have been computed at a previous iteration of the algorithm. Thus, all recursive values that are used by Theorem 1, cf. equation (16), have been already computed at a previous iteration of the algorithm as well. This completes the correctness of Algorithm 1.

There are in total $\prod_{i=1}^{k}(n_i+1)$ possible values of the vector $\mathbf{t}$, where it holds $\sum_{i=1}^{k}(n_i+1)=n+k$. The product $\prod_{i=1}^{k}(n_i+1)$ is maximized, when $(n_i+1)=\frac{n+k}{k}$ holds for every $i=1,\ldots,k$. Thus, there are in total at most $O((\frac{n}{k}+1)^k n^5)$ possible tuples $(\mathbf{t},x,y,z)$. Since the lines 6-19 are executed for all these tuples, it follows that the runtime of Algorithm 1 is $O((\frac{n}{k}+1)^k n^8)$.

For the computation of the optimal value, the algorithm stores for every tuple $(\mathbf{t},x,y,z)$ the value $F(\mathbf{t},x,y,z)$ in an array of size $O((\frac{n}{k}+1)^k n^5)$. The storage of the release and completion times in Lemmas 4 and Theorem 1 can be done in an array of linear size $O(n)$. In order to build the optimal schedule, instead of its value, we need to store at every entry of these arrays the corresponding schedule. For each one of them we store the start and completion times of the jobs in an array of size $O(n)$. Then, the optimal schedule can be easily computed by sorting these start and completion times in non-decreasing order, storing the interrupted jobs in a stack. This implies space complexity $O((\frac{n}{k}+1)^k n^6)$.

4. Concluding remarks

In this paper we presented the first polynomial algorithm for the preemptive scheduling of equal-length jobs on a single machine, parameterized on the number k of different weights. The objective is to minimize the sum of the weighted completion times $\sum_{i=1}^{n} w_i C_i$ of the jobs, where w_i and C_i is the weight and the completion time of job J_i. The complexity status of the generalized version with an arbitrary number of positive weights on a single machine remains an interesting open question for further research.

References

[1] K. Baker. *Introduction to Sequencing and Scheduling*. John Wiley and Sons, New York, 1974.

[2] P. Baptiste. An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Operations Research Letters*, 24:175–180, 1999.

[3] P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal. *Journal of Scheduling*, 2:245–252, 1999.

[4] P. Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103:21–32, 2000.

[5] P. Baptiste, P. Brucker, M. Chrobak, C. Durr, S. A. Kravchenko, and F. Sourd. The complexity of mean flow time scheduling problems with release times. *Journal of Scheduling*, 10(2):139–146, 2007.

[6] P. Baptiste, M. Chrobak, C. Durr, W. Jawor, and N. Vakhania. Preemptive scheduling of equal-length jobs to maximize weighted throughput. *Operation Research Letters*, 32(3):258–264, 2004.

[7] P. Baptiste and C. Dürr. http://www.lix.polytechnique.fr/ durr/OpenProblems/1_rj_pmtn_pjp_sumWjCj/.

[8] P. Brucker. *Scheduling algorithms*. Springer Verlag, Heidelberg, 5 edition, 2007.

[9] P. Brucker and S. Knust. Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

[10] M. Dessouky, B. Lageweg, J. Lenstra, and S. van de Velde. Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica*, 44:115–123, 1990.

[11] M. Garey and D. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979.

[12] M. Garey, D. Johnson, B. Simons, and R. Tarjan. Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. Comput.*, 10:256–269, 1981.

[13] L. Herrbach and J.-T. Leung. Preemptive scheduling of equal length jobs on two machines to minimize mean flow time. *Operations Research*, 38:487–494, 1990.

[14] J. Labetoulle, E. Lawler, J. Lenstra, and A. R. Kan. Preemptive scheduling of uniform machines subject to release dates. In *Progress in Combinatorial Optimization*, pages 245–261. Academic Press, Toronto, 1984.

[15] E. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Discrete Mathematics*, 26(1–4):125–133, 1990.

[16] J. Lenstra, A. R. Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[17] J.-T. Leung and G. Young. Preemptive scheduling to minimize mean weighted flow time. *Information Processing Letters*, 34:47–50, 1990.

[18] B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM Journal of Computing*, 12:294–299, 1983.

[19] Z. Tian, C. Ng, and T. Cheng. An $O(n^2)$ algorithm for scheduling equal-length preemptive jobs on a single machine to minimize total tardiness. *SIAM Journal of Computing*, 9(4):343–364, 2006.