RWTH Aachen

Department of Computer Science
Technical Report

Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

George B. Mertzios and Walter Unger

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2008-10

RWTH Aachen · Department of Computer Science · May 2008

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

George B. Mertzios and Walter Unger
Department of Computer Science
RWTH Aachen University
{mertzios, 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, the problem is polynomially solvable 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. 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 [1,2], while it is unary NP-hard for arbitrary processing times [2]. The problem of maximizing the weighted throughput, or equivalently of minimizing the weighted number of late jobs on a single machine, is NP-hard [3] and pseudo-polynomially solvable [4] 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 [5,6]. For the problem of minimizing the total tardiness there is also a polynomial algorithm for equal-length jobs [7]. Furthermore, minimizing the sum of completion times [8] or the number of late jobs [4,9] 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 [10, 11].

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 [13] are polynomial for equal-length jobs, while the corresponding problems in the general case are both NP-hard, also on a single machine [3, 14]. Moreover, polynomial time algorithms are presented in [15] 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 [2, 11, 16, 17]. The non-preemptive version of this problem is known to be polynomially solvable on a fixed number of identical parallel machines [16]. On the other hand, the preemptive version of this problem is known to be NP-hard if the processing times are arbitrary on a single machine [18], or even for equal processing times on identical parallel machines [19]. 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 $C_i < C_k$.

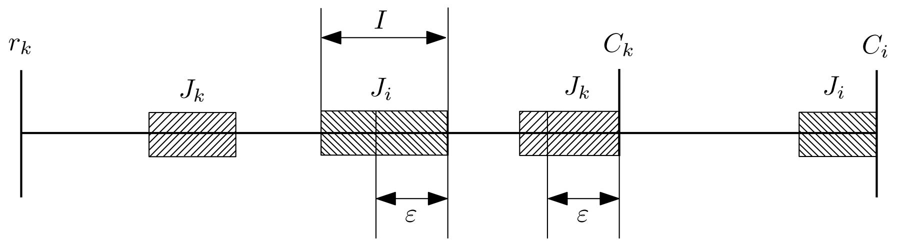

Fig. 1. The impossible case $C_i > C_k$, where job J_i is partially executed in $[r_k, C_k)$.

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 $\mathcal{J}^i = \{J_1^i, J_2^i, \ldots, J_{n_i}^i\}$, $i \in \{1, \ldots, k\}$, such that job J_ℓ^i has weight α_i for every $\ell \in \{1, \ldots, n_i\}$. Assume without loss of generality that for every i, the jobs J_ℓ^i are sorted with respect to ℓ in non-decreasing order according to their release times r_ℓ^i, i.e.

$$r_1^i \leq r_2^i \leq \ldots \leq r_{n_i}^i \tag{3}$$

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 $t_i > 0$. Furthermore, let $\mathcal{R} = \{r_\ell^i \mid 1 \leq i \leq k, 1 \leq \ell \leq n_i\}$ be the set of all release times of the jobs and

$$\mathcal{R}(\mathbf{t}) = \{r_\ell^i \mid i \in \mathcal{P}(\mathbf{t}), 1 \leq \ell \leq t_i\} \tag{8}$$

Denote now by

$$Q(\mathbf{t}, x, y, z) \tag{9}$$

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

$$r_\ell^i \in \begin{cases} [x, z), \text{ if } i = \mathbf{t}_{\max} \text{ and } \ell = t_i \\ [y, z), \qquad \text{otherwise} \end{cases} \tag{10}$$

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

$$Q(\mathbf{0}, x, y, z) = \emptyset \tag{11}$$

for all values $x \leq y < z$. Moreover, we define for every vector $\mathbf{t}$ and every triple $\{x, y, z\}$, such that $x \leq y$ and $y \geq z$

$$Q(\mathbf{t}, x, y, z) = \emptyset \tag{12}$$

Definition 1. *The set $Q(\mathbf{t}, x, y, z) \neq \emptyset$ of jobs is called* feasible*, if there exists a feasible schedule of these jobs in the interval $[y, z)$.*

For the case of a feasible set $Q(\mathbf{t}, x, y, z) \neq \emptyset$, denote now by

$$F(\mathbf{t}, x, y, z) \tag{13}$$

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 (9) and (13) to take values only from the set T. Also, due to (10), since every job is released not earlier than x, it suffices to consider that $x \in \mathcal{R}$. For an arbitrary $y \in T$, let

$$r(y) = \min\{r \in \mathcal{R} \mid r \geq y\} \tag{14}$$

be the smallest release time that equals at least y. For simplicity reasons, we define $r(y) = \max T$ in the case where there exists no release time $r \in \mathcal{R}$ with $r \geq y$, where $\max T$ is the greatest value of the set T, cf. (2). In the case where $Q(\mathbf{t}, x, y, z) \neq \emptyset$ is *not* feasible, we define $F(\mathbf{t}, x, y, z) = \infty$. In the case where $Q(\mathbf{t}, x, y, z) = \emptyset$, we define $F(\mathbf{t}, x, y, z) = 0$.

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*

$$\begin{aligned} C_1 &= \max\{\widetilde{r}_1, y\} + p \\ C_\ell &= \max\{\widetilde{r}_\ell, C_{\ell-1}\} + p \end{aligned} \tag{15}$$

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 (15) 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

Consider a vector $\mathbf{t} \neq \mathbf{0}$ and a feasible set $Q(\mathbf{t}, x, y, z) \neq \emptyset$ of jobs. Then, $y < z$ by the definition of $Q(\mathbf{t}, x, y, z)$. Furthermore, for every index $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$, if $r^i_{t_i} \notin [y, z)$, it follows that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, x, y, z) \tag{16}$$

Indeed, in this case $J^i_{t_i} \notin Q(\mathbf{t}, x, y, z)$ by (10), and thus we can ignore job $J^i_{t_i}$, i.e. we can replace t_i by $t_i - 1$. Then, all jobs of $Q(\mathbf{t}, x, y, z)$ have release times according to (10) and they are scheduled in the interval $[y, z)$. Therefore, (16) follows.

On the other hand, for $i = \mathbf{t}_{\max}$, if $r^i_{t_i} \notin [x, z)$, then

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, r(y), r(y), z) \tag{17}$$

Indeed, in this case again $J^i_{t_i} \notin Q(\mathbf{t}, x, y, z)$ by (10), and thus we can ignore job $J^i_{t_i}$, i.e. we can replace again t_i by $t_i - 1$. Then, all jobs of $Q(\mathbf{t}, x, y, z)$ 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, (17) follows. Note here that in the extreme case where $r(y) \geq z$, no job of $Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ is released in $[y, z)$, and thus $Q(\mathbf{t}, x, y, z) = \emptyset$ by (10), which is a contradiction to the assumption that $Q(\mathbf{t}, x, y, z) \neq \emptyset$.

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 $\ell \in \{1, \ldots, n_i\}$. Consider now the vector of the completion times $(C^1_1, C^1_2, \ldots, C^k_{n_k})$ and the feasible set $Q(\mathbf{t}, x, y, z) \neq \emptyset$. Let $\mathcal{C}(\mathbf{t}, x, y, z)$ be the restriction of the vector $(C^1_1, C^1_2, \ldots, C^k_{n_k})$ on those values j and ℓ, for which $J^j_\ell \in Q(\mathbf{t}, x, y, z)$. Denote now by $\mathcal{S}(\mathbf{t}, x, y, z)$ the optimal schedule of the jobs of $Q(\mathbf{t}, x, y, z)$ that lexicographically minimizes the vector $\mathcal{C}(\mathbf{t}, x, y, z)$ among all other optimal schedules. In the sequel, we denote $\mathcal{S}(\mathbf{t}, x, y, z)$ by $\mathcal{S}$, whenever the values $\mathbf{t}, x, y, z$ are clear from the context. Next, we compute in Theorems 1 and 2 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 these theorems. Denote by s_i and e_i the start and completion time of job $J^i_{t_i}$ in $\mathcal{S} = \mathcal{S}(\mathbf{t}, x, y, z)$, respectively. Also, for $i = \mathbf{t}_{\max}$, denote for simplicity $J^i_{t_i}$ and $r^i_{t_i}$ by $J_{\mathbf{t}_{\max}}$ and $r_{\mathbf{t}_{\max}}$, respectively.

Lemma 5. *Suppose that $Q(\mathbf{t}, x, y, z) \neq \emptyset$ is feasible and 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

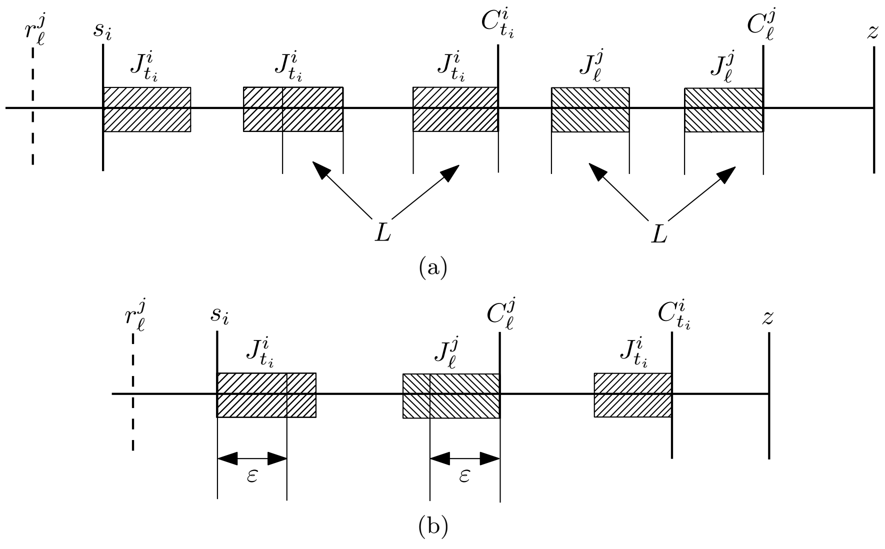

Fig. 2. The impossible case $r_\ell^j \leq s_i$, where $j \leq i$ and $C_\ell^j > s_i$.

jobs $J_{t_i}^i$ and J_ℓ^j in these intervals, obtaining a completion time of J_ℓ^j at most $C_\ell^j - \varepsilon$, 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 to its optimality. This implies again that job J_ℓ^j is released not earlier than s_i, i.e. $r_\ell^j > s_i$.

Corollary 1. *Suppose that $Q(\mathbf{t}, x, y, z) \neq \emptyset$ is feasible and that $J_{t_i}^i \in Q(\mathbf{t}, x, y, z)$ for some $i \in \mathcal{P}(\mathbf{t})$. Then, every other job $J_\ell^i \in Q(\mathbf{t}, x, y, z) \setminus \{J_{t_i}^i\}$ is completed in $\mathcal{S}$ at a point $C_\ell^i \leq s_i$.*

Proof. Consider such a job J_ℓ^i, with $\ell < t_i$ and suppose that J_ℓ^i is completed at a point $C_\ell^i > s_i$. Then, Lemma 5 implies that $r_\ell^i > s_i$. On the other side, it holds due to (3) that $r_\ell^i \leq r_{t_i}^i \leq s_i$, which is a contradiction.

Theorem 1. *Let $Q(\mathbf{t}, x, y, z) \neq \emptyset$ be feasible and $J_{t_i}^i \in Q(\mathbf{t}, x, y, z)$ for every $i \in \mathcal{P}(\mathbf{t})$. Suppose that $r_{\mathbf{t}_{\max}} > y$. Then,*

$$F(\mathbf{t}, x, y, z) = F_1 = \min_{\substack{s \in (y,z) \cap T \\ s \notin \mathcal{R}(\mathbf{t}'_{\mathbf{t}_{\max}})}} \left\{ F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s) + F(\mathbf{t}, x, s, z) \right\} \tag{18}$$

Proof. First, recall that s_i and e_i denote the start and completion times of the job $J_{t_i}^i \in Q(\mathbf{t}, x, y, z)$ in $\mathcal{S} = \mathcal{S}(\mathbf{t}, x, y, z)$, for every $i \in \mathcal{P}(\mathbf{t})$. Due to the assumption that $r_{\mathbf{t}_{\max}} > y$, it follows that also $s_{\mathbf{t}_{\max}} > y$.

For every job $J_\ell^j \in Q(\mathbf{t}, x, y, z)$ it holds $j \leq \mathbf{t}_{\max}$, due to (7). Thus, Lemma 5 implies that all jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ with release times $r_\ell^j \leq s_{\mathbf{t}_{\max}}$ are scheduled completely in the interval $[y, s_{\mathbf{t}_{\max}})$, while all jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ with release times $r_\ell^j > s_{\mathbf{t}_{\max}}$ are scheduled in $\mathcal{S}$ completely in the interval $[s_{\mathbf{t}_{\max}}, z)$. Note that the extreme case $r_\ell^j = s_{\mathbf{t}_{\max}}$ is impossible for any job $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$, since otherwise job J_ℓ^j must be scheduled in the empty interval $[s_{\mathbf{t}_{\max}}, s_{\mathbf{t}_{\max}})$, which is a contradiction. That is, $s_{\mathbf{t}_{\max}} \notin \mathcal{R}(\mathbf{t}'_{\mathbf{t}_{\max}})$.

Since $J_{\mathbf{t}_{\max}}$ is scheduled in the second part $[s_{\mathbf{t}_{\max}}, z)$ of $\mathcal{S}$, it follows that every job J_ℓ^j, which is scheduled in the first part $[y, s_{\mathbf{t}_{\max}})$ of $\mathcal{S}$, has release time $r_\ell^j \geq y$, i.e. $r_\ell^j \geq r(y)$. Thus, the value of this first part of $\mathcal{S}$ equals $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}})$. Note here that in the extreme case where $r(y) \geq s_{\mathbf{t}_{\max}}$, no job of $Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ is released in $[y, s_{\mathbf{t}_{\max}})$, and thus no job is scheduled in the first part of $\mathcal{S}$, i.e. the value of this part equals zero. However, in this case, where $r(y) \geq s_{\mathbf{t}_{\max}}$, it holds $Q(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}}) = \emptyset$ by (12), and thus $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}}) = 0$. Thus, in any case, the value of the first part of $\mathcal{S}$ equals $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}})$.

On the other hand, in the second part $[s_{\mathbf{t}_{\max}}, z)$ of $\mathcal{S}$, exactly $J_{\mathbf{t}_{\max}}$ and the jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ with release times $r_\ell^j > s_{\mathbf{t}_{\max}}$ are scheduled. Thus, since $s_{\mathbf{t}_{\max}} \notin \mathcal{R}(\mathbf{t}'_{\mathbf{t}_{\max}})$, we can state equivalently that in the second part $[s_{\mathbf{t}_{\max}}, z)$ of $\mathcal{S}$, exactly $J_{\mathbf{t}_{\max}}$ and the jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ with release times $r_\ell^j \geq s_{\mathbf{t}_{\max}}$ are scheduled. Therefore, 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

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), s_{\mathbf{t}_{\max}}) + F(\mathbf{t}, x, s_{\mathbf{t}_{\max}}, z) \tag{19}$$

Conversely, if the value of (19) is finite, then it corresponds to a feasible schedule of the jobs of $Q(\mathbf{t}, x, y, z)$ in the interval $[y, z)$. Thus, since $\mathcal{S}$ is assumed to be optimal, the value $F(\mathbf{t}, x, y, z)$ is the minimum of the expression in (19) over all possible values $s = s_{\mathbf{t}_{\max}} \in (y, z) \cap T$, such that $s_{\mathbf{t}_{\max}} \notin \mathcal{R}(\mathbf{t}'_{\mathbf{t}_{\max}})$.

Theorem 2. *Let $Q(\mathbf{t}, x, y, z) \neq \emptyset$ be feasible and $J_{t_i}^i \in Q(\mathbf{t}, x, y, z)$ for every $i \in \mathcal{P}(\mathbf{t})$. Suppose that $r_{\mathbf{t}_{\max}} \leq y$ and let $e = y + p \cdot |Q(\mathbf{t}, x, y, z)|$. If $Q(\mathbf{t}, r(e), r(e), z) \neq \emptyset$, then*

$$F(\mathbf{t}, x, y, z) = \min_{\substack{s \in (y,z) \cap T \\ i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\} \\ s \geq r(y),\ s \notin \mathcal{R}(\mathbf{t}'_i)}} \left\{ F_1, F(\mathbf{t}'_i, x, y, s) + F(\mathbf{t}''_i, r(y), s, z) \right\} \tag{20}$$

Otherwise, if $Q(\mathbf{t}, r(e), r(e), z) = \emptyset$, then

$$F(\mathbf{t}, x, y, z) = \min_{\substack{s \in (y,z) \cap T \\ i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\} \\ s \geq r(y),\ s \notin \mathcal{R}(\mathbf{t}'_i)}} \left\{ \begin{array}{c} F_1, \\ F(\mathbf{t}'_i, x, y, s) + F(\mathbf{t}''_i, r(y), s, z), \\ F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e) + e \cdot \alpha_{\mathbf{t}_{\max}} \end{array} \right\} \tag{21}$$

where F_1 is the value computed in (18).

Proof. Similarly to the proof of Theorem 1, let job $J_{t_i}^i \in Q(\mathbf{t}, x, y, z)$ start at point s_i and complete at point e_i in $\mathcal{S} = \mathcal{S}(\mathbf{t}, x, y, z)$, for every $i \in \mathcal{P}(\mathbf{t})$. In the case where $s_{\mathbf{t}_{\max}} > y$, Theorem 1 implies that $F(\mathbf{t}, x, y, z) = F_1$, where F_1 is the value computed in (18). Suppose in the sequel of the proof that $s_{\mathbf{t}_{\max}} = y$. We distinguish in the following two cases.

Case 1. Suppose that there exists an index $i \in \mathcal{P}(\mathbf{t})$, such that $s_i \geq e_{\mathbf{t}_{\max}}$, and let i be the greatest among them. Then, $i < \mathbf{t}_{\max}$ and $y < s_i < z$. That is, for every index $j \in \mathcal{P}(\mathbf{t})$ with $j > i$, job $J_{t_j}^j$ starts at a point $s_j \in [s_{\mathbf{t}_{\max}}, e_{\mathbf{t}_{\max}})$ in $\mathcal{S}$, 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 $j \in \mathcal{P}(\mathbf{t})$ (where $j > i$), 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 release times $r_\ell^j \leq s_i$ are scheduled completely in the interval $[y, s_i)$, while all jobs $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{t_i}^i\}$ with release times $r_\ell^j > s_i$ are scheduled in $\mathcal{S}$ completely in the interval $[s_i, z)$. Note that the extreme case $r_\ell^j = s_i$ is impossible for any job $J_\ell^j \in Q(\mathbf{t}, x, y, z) \setminus \{J_{t_i}^i\}$, since otherwise job J_ℓ^j must be scheduled in the empty interval $[s_i, s_i)$, which is a contradiction. That is, $s_i \notin \mathcal{R}(\mathbf{t}'_i)$. Furthermore, since the release time of $J_{t_i}^i$ is assumed to be $r_{t_i}^i \geq y$, i.e. $r_{t_i}^i \geq r(y)$, and since $s_i \geq r_{t_i}^i$, it follows that $s_i \geq r(y)$.

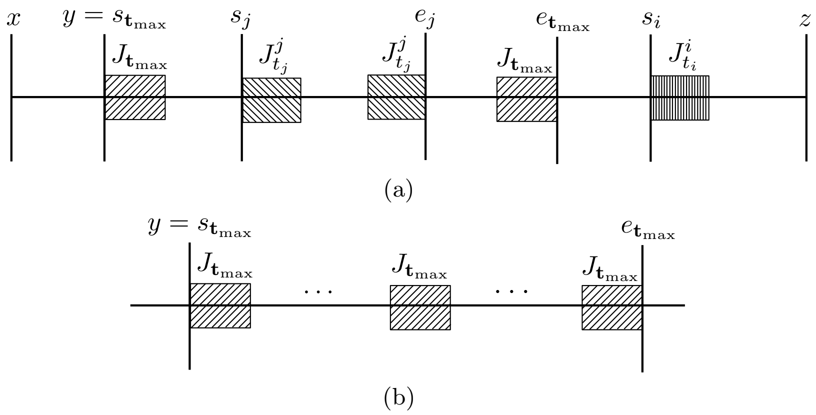

Fig. 3. The case $s_{\mathbf{t}_{\max}} = y$.

Note that $J_{\mathbf{t}_{\max}}$ is scheduled in the first part $[y, s_i)$ of $\mathcal{S}$, since we assumed that $y = s_{\mathbf{t}_{\max}}$, while $J^i_{t_i}$ is scheduled in the second part $[s_i, z)$ of $\mathcal{S}$. Thus, since $J_{\mathbf{t}_{\max}}$ is released not earlier than x, the value of the first part $[y, s_i)$ of $\mathcal{S}$ equals $F(\mathbf{t}'_i, x, y, s_i)$.

In the second part $[s_i, z)$ of $\mathcal{S}$, exactly $J^i_{t_i}$ and the jobs $J^j_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ with $j \leq i$ and release times $r^j_\ell > s_i$ are scheduled. Thus, since $s_i \notin \mathcal{R}(\mathbf{t}'_i)$, we can state equivalently that in the second part $[s_i, z)$ of $\mathcal{S}$, exactly $J^i_{t_i}$ and the jobs $J^j_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J^i_{t_i}\}$ with $j \leq i$ and release times $r^j_\ell \geq s_i$ are scheduled. Since the release time of $J^i_{t_i}$ is assumed to be $r^i_{t_i} \geq y$, i.e. $r^i_{t_i} \geq r(y)$, the value of the second part of $\mathcal{S}$ equals $F(\mathbf{t}''_i, r(y), s_i, z)$. Note here that, since $r(y) \leq s_i < z$, the value $F(\mathbf{t}''_i, r(y), s_i, z)$ is well defined. It follows that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_i, x, y, s_i) + F(\mathbf{t}''_i, r(y), s_i, z) \tag{22}$$

Conversely, if the value of (22) is finite, then it corresponds to a feasible schedule of the jobs of $Q(\mathbf{t}, x, y, z)$ in the interval $[y, z)$. Thus, since $\mathcal{S}$ is assumed to be optimal, the value $F(\mathbf{t}, x, y, z)$ equals (in Case 1) to the minimum of the expression in (22) over all possible values of $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$ and $s = s_i \in (y, z) \cap T$, such that $s \notin \mathcal{R}(\mathbf{t}'_i)$ and $s \geq r(y)$.

Case 2. Suppose that $s_i < e_{\mathbf{t}_{\max}}$ for every $i \in \mathcal{P}(\mathbf{t})$. Then, Corollary 1 implies that for every $i \in \mathcal{P}(\mathbf{t})$, all jobs $J^i_\ell \in Q(\mathbf{t}, x, y, z)$ with $\ell < t_i$ are completed at most at point s_i in $\mathcal{S}$. 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 hand, 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,

$$e_{\mathbf{t}_{\max}} = y + p \cdot |Q(\mathbf{t}, x, y, z)| \tag{23}$$

Note that, since $Q(\mathbf{t}, x, y, z)$ is assumed to be feasible, there exists a feasible schedule of the jobs of $Q(\mathbf{t}, x, y, z)$ in the interval $[y, z)$, and thus, $z \geq e_{\mathbf{t}_{\max}} = y + p \cdot |Q(\mathbf{t}, x, y, z)|$. Furthermore, since all jobs of $Q(\mathbf{t}, x, y, z)$ are scheduled completely in the interval $[y, e_{\mathbf{t}_{\max}})$, it follows in particular that all jobs of $Q(\mathbf{t}, x, y, z)$ are released strictly before $e_{\mathbf{t}_{\max}}$, and thus $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$. Note here that, in the extreme case where $r(e_{\mathbf{t}_{\max}}) \geq z$, again $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$ by (12).

Now, Lemma 1 implies that no part of $J_{\mathbf{t}_{\max}}$ is executed in any time interval $[r^i_\ell, C^i_\ell)$, where $J^i_\ell \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$, since otherwise $J_{\mathbf{t}_{\max}}$ would complete before J^i_ℓ, 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 $\mathcal{S}$. Recall that all jobs $J_\ell^i \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ have release times $r_\ell^i \geq y$, i.e. $r_\ell^i \geq r(y)$. Thus, since the weight of $J_{\mathbf{t}_{\max}}$ is $\alpha_{\mathbf{t}_{\max}}$ and its completion time is $e_{\mathbf{t}_{\max}}$, it follows in this case that

$$F(\mathbf{t}, x, y, z) = F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) + e_{\mathbf{t}_{\max}} \cdot \alpha_{\mathbf{t}_{\max}} \tag{24}$$

Note here that in the extreme case where $r(y) \geq e_{\mathbf{t}_{\max}}$, no job of $Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ is released in $[y, e_{\mathbf{t}_{\max}})$, and thus no job except $J_{\mathbf{t}_{\max}}$ is scheduled in $\mathcal{S}$, i.e. $F(\mathbf{t}, x, y, z) = e_{\mathbf{t}_{\max}} \cdot \alpha_{\mathbf{t}_{\max}}$. In this case, where $r(y) \geq e_{\mathbf{t}_{\max}}$, it holds $Q(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) = \emptyset$ by (12), and thus $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) = 0$. Thus, in any case, the value of $F(\mathbf{t}, x, y, z)$ is given by (24).

Conversely, suppose that $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$ and that the value of $F(\mathbf{t}, x, y, z)$ in (24) is finite, or equivalently, the value $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}})$ is finite, where $e_{\mathbf{t}_{\max}}$ is given by (23). Then, since $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$, all jobs $J_\ell^i \in Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ have release times r_ℓ^i, such that $r(y) \leq r_\ell^i < e_{\mathbf{t}_{\max}}$.

If $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) = 0$, then $Q(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) = \emptyset$. Therefore, since also $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$, it follows that $Q(\mathbf{t}, x, y, z) = \{J_{\mathbf{t}_{\max}}\}$, and thus $F(\mathbf{t}, x, y, z) = e_{\mathbf{t}_{\max}} \cdot \alpha_{\mathbf{t}_{\max}}$ corresponds to a feasible schedule of $Q(\mathbf{t}, x, y, z)$ in $[y, z)$.

In the opposite case, where $F(\mathbf{t}'_{\mathbf{t}_{\max}}, r(y), r(y), e_{\mathbf{t}_{\max}}) \neq 0$, this value corresponds to a feasible schedule $\mathcal{S}_0$ of the jobs of the set $Q(\mathbf{t}, x, y, z) \setminus \{J_{\mathbf{t}_{\max}}\}$ in the interval $[y, e_{\mathbf{t}_{\max}})$. Since the processing time of each job is p, the total processing time of these jobs in $[y, e_{\mathbf{t}_{\max}})$ is $p \cdot (|Q(\mathbf{t}, x, y, z)| - 1)$. Thus, due to (23), the machine has idle periods in the interval $[y, e_{\mathbf{t}_{\max}})$ of total length p (in the schedule $\mathcal{S}_0$). Therefore, since $r_{\mathbf{t}_{\max}} \leq y$ by the assumption, we can schedule the job $J_{\mathbf{t}_{\max}}$ in these idle periods, obtaining a feasible schedule of all jobs of $Q(\mathbf{t}, x, y, z)$ in the interval $[y, e_{\mathbf{t}_{\max}})$ with value $F(\mathbf{t}, x, y, z)$, as it is expressed in (24). That is, if $Q(\mathbf{t}, r(e_{\mathbf{t}_{\max}}), r(e_{\mathbf{t}_{\max}}), z) = \emptyset$, and if the value of (24) is finite, then this value corresponds to a feasible schedule of the jobs of $Q(\mathbf{t}, x, y, z)$ in the interval $[y, z)$. Thus, since $\mathcal{S}$ is assumed to be optimal, the value $F(\mathbf{t}, x, y, z)$ equals (in Case 2) to the expression in (24) for $e_{\mathbf{t}_{\max}} = y + p \cdot |Q(\mathbf{t}, x, y, z)|$.

Summarizing now Cases 1 and 2, and since $\mathcal{S}$ is optimal, it follows that the optimal value $F(\mathbf{t}, x, y, z)$ is the minimum among the value F_1 (computed in (18)) and the values of the expressions in (22) and (24), over all possible values $s = s_i \in (y, z) \cap T$ and $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$, such that $s \notin \mathcal{R}(\mathbf{t}'_i)$ and $s \geq r(y)$. This completes 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

$$F(\mathbf{t}^*, \min T, \min T, \max T) \tag{25}$$

where

$$\mathbf{t}^* = (n_1, n_2, \ldots, n_k) \tag{26}$$

and $\min T$, $\max T$ denote the smallest and the greatest value of the set T, respectively, cf. (2). Note that $\min T$ coincides with the smallest release time. The dynamic programming Algorithm 1 follows now by Lemma 4 and Theorems 1 and 2. The correctness and the complexity of this algorithm is proved in the next theorem.

Note that, as a preprocessing step, we partition the n jobs into the sets $\mathcal{J}^i = \{J_1^i, J_2^i, \ldots, J_{n_i}^i\}$, $i \in \{1, \ldots, k\}$, such that job J_ℓ^i has weight α_i for every $\ell \in \{1, \ldots, n_i\}$, and that, for every i, the jobs J_ℓ^i are sorted with respect to ℓ according to (3). This can be done clearly in $O(n \log n)$ time.

Theorem 3. *An optimal schedule can be computed in $O((\frac{n}{k}+1)^k n^8)$ time and $O((\frac{n}{k}+1)^k n^6)$ space.*

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

1: **for** every $x \in \mathcal{R}$ and $y, z \in T$, with $x \leq y < z$ **do**
2: $\quad F(\mathbf{0}, x, y, z) \leftarrow 0$ {initialization}
3: **for** every $\mathbf{t}$ between $\mathbf{0}$ and $\mathbf{t}^*$, $x \in \mathcal{R}$ and $y, z \in T$, with $x \leq y$ and $y \geq z$ **do**
4: $\quad F(\mathbf{t}, x, y, z) \leftarrow 0$ {initialization}
5: **for** every $\mathbf{t}$ between $\mathbf{0}$ and $\mathbf{t}^*$ in lexicographical order **do**
6: $\quad$ **for** every $x \in \mathcal{R}$ and $z \in T$ with $x < z$ **do**
7: $\quad\quad$ **for** $y = z$ downto x (with $y \in T$ and $y \neq z$) **do**
8: $\quad\quad\quad$ **if** $Q(\mathbf{t}, x, y, z) = \emptyset$ **then**
9: $\quad\quad\quad\quad F(\mathbf{t}, x, y, z) \leftarrow 0$
10: $\quad\quad\quad$ **else if** $Q(\mathbf{t}, x, y, z)$ is not feasible **then**
11: $\quad\quad\quad\quad F(\mathbf{t}, x, y, z) \leftarrow \infty$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad$ **for** every $i \in \mathcal{P}(\mathbf{t})$ **do**
14: $\quad\quad\quad\quad\quad$ **if** $i = \mathbf{t}_{\max}$ **then**
15: $\quad\quad\quad\quad\quad\quad$ **if** $r^i_{t_i} \notin [x, z)$ **then**
16: $\quad\quad\quad\quad\quad\quad\quad F(\mathbf{t}, x, y, z) \leftarrow F(\mathbf{t}'_i, r(y), r(y), z)$
17: $\quad\quad\quad\quad\quad$ **else** $\{i \neq \mathbf{t}_{\max}\}$
18: $\quad\quad\quad\quad\quad\quad$ **if** $r^i_{t_i} \notin [y, z)$ **then**
19: $\quad\quad\quad\quad\quad\quad\quad F(\mathbf{t}, x, y, z) \leftarrow F(\mathbf{t}'_i, x, y, z)$
20: $\quad\quad\quad\quad$ **if** $F(\mathbf{t}, x, y, z)$ has not been computed in lines 16 or 19 **then**
21: $\quad\quad\quad\quad\quad$ Compute $F(\mathbf{t}, x, y, z)$ by Theorems 1 and 2
22: **return** $F(\mathbf{t}^*, \min T, \min T, \max T)$

Proof. We present Algorithm 1 that computes the value of an optimal schedule of the given n jobs. A slight modification of this algorithm returns an optimal schedule, instead of its value only. In lines 1-4, Algorithm 1 initializes $F(\mathbf{0}, x, y, z) = 0$ for all possible values of x, y, z, such that $x \leq y < z$, as well as $F(\mathbf{t}, x, y, z) = 0$ for all possible values of $\mathbf{t}, x, y, z$, such that $x \leq y$ and $y \geq z$, cf. (11) and (12). It iterates further for every $\mathbf{t}$ between $\mathbf{0}$ and $\mathbf{t}^*$ in lexicographical order and for every possible x, y, z, such that $x \leq y < z$. 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 8. If this set is empty, it defines $F(\mathbf{t}, x, y, z) = 0$. Otherwise, it checks in line 10 its feasibility, using Lemma 4 and, if it is not feasible, it defines $F(\mathbf{t}, x, y, z) = \infty$. In the case of feasibility of the set $Q(\mathbf{t}, x, y, z)$, the algorithm checks in lines 13-19 the release times of the jobs $J^i_{t_i}$ for all $i \in \mathcal{P}(\mathbf{t})$. 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 16 and 19, due to (17) and (16), respectively. Finally, if all jobs $J^i_{t_i}$, $i \in \mathcal{P}(\mathbf{t})$ belong to $Q(\mathbf{t}, x, y, z)$, i.e. if the value $F(\mathbf{t}, x, y, z)$ has not been computed in the lines 16 or 19, the algorithm computes $F(\mathbf{t}, x, y, z)$ in line 21 by Theorems 1 and 2.

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 $F(\mathbf{t}'_i, \cdot, \cdot, \cdot)$ and $F(\mathbf{t}''_i, \cdot, \cdot, \cdot)$, which are used in lines 16 and 19, as well as in equations (18), (20), and (21), have been already computed at a previous iteration of the algorithm. Furthermore, since we iterate for y in line 7 from the value z downwards to the value x, the values $F(\mathbf{t}, x, s, z)$, for every s with $y < s < z$, cf. equation (18), have been also computed at a previous iteration of the algorithm. Thus, all recursive values that are used by Theorems 1 and 2, cf. equations (18), (20), and (21), have been already computed at a previous iteration of the algorithm. This completes the correctness of Algorithm 1.

The running time of the algorithm can be computed as follows. For each vector $\mathbf{t} = (t_k, t_{k-1}, \ldots, t_1)$, the set $\mathcal{P}(\mathbf{t}) = \{i \mid t_i > 0, 1 \leq i \leq k\}$ and the value $\mathbf{t}_{\max} = \max \mathcal{P}(\mathbf{t})$ can be computed in linear $O(n)$ time, since $k \leq n$. Thus, the computation of the set $Q(\mathbf{t}, x, y, z)$ in line 8 can be done in linear time as well. Indeed, since $y < z$, we can check in linear time whether $\mathbf{t} = \mathbf{0}$, cf. (11), while we can check also in linear time in (10) the release times of the jobs $\bigcup_{i \in \mathcal{P}(\mathbf{t})} \bigcup_{\ell=1}^{t_i} J_\ell^i$. The feasibility of $Q(\mathbf{t}, x, y, z)$ in line 10 can be checked 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. The execution of lines 13-19 can be simply done in linear time, by checking the release times of the jobs $J_{t_i}^i$, for all $i \in \mathcal{P}(\mathbf{t})$.

For the computation of $F(\mathbf{t}, x, y, z)$ by Theorems 1 and 2, the algorithm uses for at most every $s \in T$ and every $i \in \mathcal{P}(\mathbf{t}) \setminus \{\mathbf{t}_{\max}\}$ the values of one or two smaller instances that have been already computed at a previous iteration. This takes $O(n^3)$ time, since T has at most n^2 elements and $\mathcal{P}(\mathbf{t})$ has at most n elements. Furthermore, the sets $\mathcal{R}(\mathbf{t}'_{\mathbf{t}_{\max}})$ and $\mathcal{R}(\mathbf{t}'_i)$ in the statements of these theorems can be computed in linear $O(n)$ time by (8). Moreover, the set $Q(\mathbf{t}, r(e), r(e), z)$ in the statement of Theorem 2 can be computed in linear $O(n)$ time. Indeed, we can check in linear time whether $\mathbf{t} = \mathbf{0}$ or whether $r(e) \geq z$, cf. (11) and (12), while we can check also in linear time in (10) the release times of the jobs $\bigcup_{i \in \mathcal{P}(\mathbf{t})} \bigcup_{\ell=1}^{t_i} J_\ell^i$. Thus, the algorithm needs $O(n^3)$ time for the execution of the lines 8-21.

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 $(n_i + 1) = \frac{n+k}{k}$ for every $i = 1, \ldots, k$. Thus, there are in total at most $O((\frac{n}{k} + 1)^k)$ vectors $\mathbf{t}$ and $O((\frac{n}{k} + 1)^k n^5)$ possible tuples $(\mathbf{t}, x, y, z)$, since $x \in \mathcal{R}$ can take at most $O(n)$ possible values and $y, z \in T$ can take at most $O(n^2)$ possible values each. Since the lines 8-21 are executed for all these tuples, the algorithm needs for the lines 5-21 $O((\frac{n}{k} + 1)^k n^8)$ time. Furthermore, the initialization of the values $F(\mathbf{0}, x, y, z)$ for all possible x, y, z in lines 1-2 takes $O(n^5)$ time. Finally, the initialization of the values $F(\mathbf{t}, x, y, z)$ in lines 3-4 takes $O((\frac{n}{k} + 1)^k n^5)$ time, since it is executed for at most all possible tuples $(\mathbf{t}, x, y, z)$. Summarizing, the running time of Algorithm 1 is $O((\frac{n}{k} + 1)^k n^8)$.

The space complexity of Algorithm 1 can be computed as follows. 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 Conclusion

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. L.A. Herrbach and J.Y.-T. Leung. Preemptive scheduling of equal length jobs on two machines to minimize mean flow time. *Operations Research*, 38:487–494, 1990.
2. P. Baptiste, P. Brucker, M. Chrobak, C. Durr, Svetlana A. Kravchenko, and F. Sourd. The complexity of mean flow time scheduling problems with release times. *Journal of Scheduling*, 10(2):139–146, 2007.
3. M.R. Garey and D.S. Johnson. *Computers and intractability, a guide to the theory of NP-completeness.* W.H. Freeman and Company, 1979.

4. E.L. 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.
5. 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.
6. P. Baptiste, M. Chrobak, Ch. Dürr, W. Jawor, and N. Vakhania. Preemptive scheduling of equal-length jobs to maximize weighted throughput. *Operation Research Letters*, 32(3):258–264, 2004.
7. Z. Tian, C.T. Ng, and T.C.E. 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.
8. K.R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley and Sons, New York, 1974.
9. 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.
10. P. Brucker. *Scheduling algorithms*. Springer Verlag, Heidelberg, 5 edition, 2007.
11. P. Brucker and S. Knust. Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/OR/class/.
12. M.R. Garey, D.R. Johnson, B.B. Simons, and R.E. Tarjan. Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. Comput.*, 10:256–269, 1981.
13. B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM Journal of Computing*, 12:294–299, 1983.
14. J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
15. M.I. Dessouky, B.J. Lageweg, J.K. Lenstra, and S.L. van de Velde. Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica*, 44:115–123, 1990.
16. P. Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103:21–32, 2000.
17. P. Baptiste and C. Dürr. http://www.lix.polytechnique.fr/ durr/ OpenProblems/1_rj_pmtn_pjp_sumWjCj/.
18. J. Labetoulle, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Preemptive scheduling of uniform machines subject to release dates. In *Progress in Combinatorial Optimization*, pages 245–261. Academic Press, Toronto, 1984.
19. J.Y.-T. Leung and G.H. Young. Preemptive scheduling to minimize mean weighted flow time. *Information Processing Letters*, 34:47–50, 1990.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: biblio@informatik.rwth-aachen.de

2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15 Uwe Naumann: The Complexity of Derivative Computation
2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”
2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-01 * Fachgruppe Informatik: Jahresbericht 2005
2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems
2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation
2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F
2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color
2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations
2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning
2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"
2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01 * Fachgruppe Informatik: Jahresbericht 2006
2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations
2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase
2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation
2007-05 Uwe Naumann: On Optimal DAG Reversal
2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking
2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications
2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René :Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.