SCHEDULING UNIT EXECUTION TIME TASKS ON TWO PARALLEL MACHINES WITH THE CRITERIA OF MAKESPAN AND TOTAL COMPLETION TIME

Yakov Zinder and Van Ha Do
Department of Mathematical Sciences, University of Technology, Sydney, Australia
Yakov.Zinder@uts.edu.au and vhdo@it.uts.edu.au

Abstract Two extensions of the classical scheduling model with two parallel identical machines and a partially ordered set of unit execution time tasks are considered. It is well known that the Coffman–Graham algorithm constructs for this model a so-called ideal schedule: that is, a schedule which is optimal for both makespan and total completion time criteria simultaneously. The question of the existence of such a schedule for the extension of this model, where each task has a release time, has remained open over several decades. The paper gives a positive answer to this question and presents the corresponding polynomial-time algorithm. Another straightforward generalization of the considered classical model is obtained by the introduction of multiprocessor tasks. It is shown that, despite the fact that a slightly modified Coffman–Graham algorithm solves the makespan problem with multiprocessor tasks for arbitrary precedence constraints, generally an ideal schedule does not exist and the problem with the criterion of total completion time turns out to be NP-hard in the strong sense even for in-trees.

Keywords: scheduling, parallel machines, precedence constraints, multiprocessor tasks.

1. INTRODUCTION

This paper is concerned with two extensions of the classical scheduling model which can be stated as follows. Suppose that a set of n tasks (jobs, operations) $N = \{1, \ldots, n\}$ is to be processed by two parallel identical machines. The restrictions on the order in which tasks can be processed are given in the form of an anti-reflexive, anti-symmetric and transitive relation on the set of tasks and will be referred to as precedence constraints. If task i precedes task k, denoted $i \rightarrow k$, then the processing of i must be completed before the processing of k begins. If $i \rightarrow k$, then k is called a successor of i and i is called a predecessor of k. Each machine can process at most one task at a

time, and each task can be processed by any machine. If a machine starts to process a task, then it continues until its completion, i.e. no preemptions are allowed. The processing time of each task i is equal to one unit of time. Both machines become available at time $t = 0$. Since preemptions are not allowed and the machines are identical, to specify a schedule σ it suffices to define for each task i its completion time $C_i(\sigma)$. The goal is to find a schedule which minimizes some criterion $\gamma(\sigma)$.

In the standard three-field notation this problem is denoted by

$$P2 \,|\, prec, p_j = 1 \,|\, \gamma, \tag{1}$$

where $P2$ specifies that tasks are to be processed on two identical machines, $prec$ indicates presence of precedence constraints, and $p_j = 1$ reflects the fact that each task is a unit execution time (UET) task, i.e. requires one unit of processing time.

The criteria of makespan

$$C_{max}(\sigma) = \max_{1 \le i \le n} C_i(\sigma) \tag{2}$$

and total completion time

$$C_{\Sigma}(\sigma) = \sum_{1 \le i \le n} C_i(\sigma) \tag{3}$$

are among the most frequently used in scheduling theory. Several algorithms have been developed for the $P2 \,|\, prec, p_j = 1 \,|\, C_{max}$ problem. One of them, known as the Coffman–Graham algorithm (Coffman and Graham, 1972), also solves the $P2 \,|\, prec, p_j = 1 \,|\, C_{\Sigma}$ problem (see Lawler *et al.*, 1993; Coffman *et al.*, 2003). Following Coffman *et al.* (2003), we will say that a schedule is ideal if it is optimal for both (2) and (3) simultaneously.

A straightforward generalization of (1) is the $P2 \,|\, prec, r_j, p_j = 1 \,|\, \gamma$ problem, which differs from the former one only by the assumption that the processing of each task j cannot commence before the given integer release time r_j. Section 2 gives a positive answer to the open question of the existence of an ideal schedule for the model with release times. A polynomial-time algorithm, presented in Section 2, combines the ideas of the Coffman–Graham algorithm and the Garey–Johnson algorithm (Garey and Johnson, 1977), which solves the $P2 \,|\, prec, r_j, p_j = 1 \,|\, L_{max}$ problem with the criterion of maximum lateness, where

$$L_{\max}(\sigma) = \max_{1 \le i \le n} [C_i(\sigma) - d_i]$$

and d_i is a due date associated with task i. A relevant result can also be found in Zinder (1986), which is concerned with the problem of finding a schedule

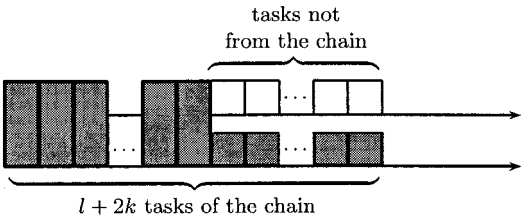

Figure 1. Schedule σ_1.

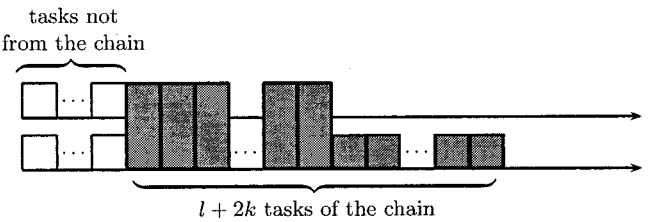

Figure 2. Schedule σ_2.

with the smallest value of C_Σ among all schedules optimal for the criterion of maximum lateness.

Another way of generalizing (1) is the introduction of multiprocessor tasks. The new problem, denoted by $P2\,|\,prec, p_j = 1, size_j\,|\,\gamma$ differs from the original one only by the assumption that each task requires for processing either one or two machines simultaneously, which is indicated by the parameter $size_j$. If $size_j = 1$, then task j needs only one machine. If $size_j = 2$, then task j can be processed only if two machines are used simultaneously. An arbitrary instance of the model with multiprocessor tasks does not necessarily allow an ideal schedule. Indeed, for arbitrary positive integers k and l, where $l > k$, consider the set of $4k + l$ tasks such that $l + 2k$ tasks form a chain and all remaining tasks do not precede or succeed any other tasks. The first l tasks in this chain require two machines simultaneously and all other tasks can be processed on one machine. It is easy to see that in any schedule, optimal for the criterion C_{max}, each task which does not belong to the chain must be processed in parallel with one of the last $2k$ tasks from the chain. Such a schedule, say σ_1, is depicted in Figure 1, where shaded tasks are the tasks constituting the chain.

Figure 2 depicts another schedule, say σ_2, where all tasks which do not belong to the chain are processed during the first k time units, followed by the tasks constituting the chain.

It is easy to see that σ_2 is not optimal for the criterion C_{max} and that

$$C_\Sigma(\sigma_1) - C_\Sigma(\sigma_2) = k(l-k) > 0$$

Section 3 shows that the $P2 \,|\, prec, p_j = 1, size_j \,|\, C_\Sigma$ problem is NP-hard in the strong sense even if the precedence constraints are restricted to in-trees. This result strengthens the result in Brucker *et al.* (2000), where NP-hardness has been proven for series-parallel graphs, and complements Zinder *et al.* (2002), where NP-hardness has been proven for out-trees.

2. IDEAL SCHEDULES WITH RELEASE TIMES

2.1 Algorithms

The model considered in this section assumes that each task j is assigned a non-negative integer r_j (referred to as a release time) and that processing of this task cannot commence before time r_j. Without loss of generality it will be assumed that the relation $i \rightarrow j$ implies the inequality $r_i \leq r_j - 1$ and that $\min_{i \in N} r_i = 0$.

The algorithm presented in this section uses the ideas introduced in Coffman and Graham (1972) and Garey and Johnson (1977). The central concept of Garey and Johnson (1977) is the notion of consistent due dates. In the process of constructing an ideal schedule, consistent due dates will be calculated for different subsets of the set N. Let $J \subseteq \{1, \ldots, n\}$. Let $\{D_i : i \in J\}$ be a set of arbitrary positive integers associated with tasks constituting J. We will refer to such integers as due dates. For any task $u \in J$ and any two numbers s and d such that

$$r_u \leq s \leq D_u \leq d, \tag{4}$$

$S(u, s, d, J)$ will denote the set of all tasks $k \in J$ such that $k \neq u$, $D_k \leq d$, and either $u \rightarrow k$ or $r_k \geq s$. Similar to Garey and Johnson (1977), we will say that the set of due dates $\{D_i : i \in J\}$ is consistent or that the due dates are consistent if $r_u \leq D_u - 1$, for all $u \in J$, and for every task $u \in J$ and any two integers s and d satisfying (4), either $|S(u, s, d, J)| = 2(d - s)$ and $D_u = s$, or $|S(u, s, d, J)| < 2(d - s)$.

Garey and Johnson (1977) developed a procedure which for any given set of positive due dates $\{d_i : i \in J\}$ either determines a set of consistent due dates $\{D_i : i \in J\}$ satisfying the inequalities $D_i \leq d_i$, for all $i \in J$, or determines that such consistent due dates do not exist at all. If due dates $\{d_i : i \in J\}$ are consistent, then the procedure returns these due dates as the set $\{D_i : i \in J\}$. The complexity of this procedure is $O(|J|^3)$. Some important results from Garey and Johnson (1977) are presented below in the form of three theorems. In what follows, the expression "schedule α for the set J" means that only tasks belonging to J are considered with precedence constraints which exist

between these tasks in the original problem. In other words, such α schedules only tasks from J and ignores the existence of other tasks.

Theorem 1 *Let $\{d_i : i \in J\}$ be a set of arbitrary positive due dates. If the procedure from Garey and Johnson (1977) fails to determine the set of consistent due dates, then there is no schedule α for the set J satisfying the inequalities $C_i(\alpha) \leq d_i$, for all $i \in J$.*

Theorem 2 *Let $\{d_i : i \in J\}$ be a set of arbitrary positive due dates and let $\{D_i : i \in J\}$ be the set of the corresponding consistent due dates calculated by the procedure from Garey and Johnson (1977). Then a schedule α for the set J satisfies the inequalities $C_i(\alpha) \leq d_i$, for all $i \in J$, if and only if it satisfies the inequalities $C_i(\alpha) \leq D_i$, for all $i \in J$.*

The third theorem shows how to construct a schedule that meets all due dates if such a schedule exists. The theorem uses the notion of list schedule. A list schedule is a schedule constructed by the following algorithm (known as the list algorithm). Suppose that all tasks are arranged in a list.

1. Among all tasks select a task j with the smallest release time. Let $T = r_j$.
2. Scan the list from left to right and find the first task available for processing in time interval $[T, T+1]$. Assign this task to the first machine and eliminate it from the list.
3. Continue to scan the list until either another task available for processing in the time interval $[T, T+1]$ has been found, or the end of the list has been reached. If another task has been found, assign this task to the second machine and eliminate it from the list. Otherwise leave the second machine idle.
4. If all tasks have been assigned, then halt. Otherwise among all tasks, which have not been assigned for processing, select task i with the smallest completion time. Set $T = \max\{T+1, r_i\}$ and go to Step 2.

Theorem 3 *Let $\{D_i : i \in J\}$ be a set of consistent due dates. Then any list schedule α for the set J, corresponding to a list where tasks are arranged in a nondecreasing order of consistent due dates D_i, meets these due dates, i.e. $C_i(\alpha) \leq D_i$, for all $i \in J$.*

The algorithm described in this section constructs an ideal schedule, denoted δ, in iterative manner. At each iteration the procedure determines a fragment of δ referred to as a block. A task can be included in a block only if all its predecessors have been already included in the same or previously constructed

blocks. Each block is a fragment of some list schedule. In constructing this list schedule the procedure associates with each task i, which has not been included in the previously constructed blocks, a positive integer μ_i, referred to as a priority, and forms a list by arranging tasks in the decreasing order of their priorities (no two tasks have the same priority). Although both criteria C_{max} and C_Σ do not assume any due dates, the calculation of priorities is based on some consistent due dates. The subroutine, which calculates priorities, will be referred to as the μ-algorithm. In order to describe the μ-algorithm it is convenient to introduce the following notation. For an arbitrary integer t and an arbitrary task i, $K(i,t)$ will denote the set of all tasks j such that $r_j < t$ and $i \rightarrow j$. Suppose that $K(i,t) \neq \emptyset$ and each task $j \in K(i,t)$ has been already assigned its priority μ_j. Denote by $\omega(i,t)$ the $|K(i,t)|$-dimensional vector $(\mu_{j_1}, \ldots, \mu_{j_{|K(i,t)|}})$, where $\mu_{j_1} > \cdots > \mu_{j_{|K(i,t)|}}$ and $j_k \in K(i,t)$ for all $1 \leq k \leq |K(i,t)|$. In other words, $\omega(i,t)$ lists all successors j of task i, satisfying the condition $r_j < t$, in the decreasing order of their priorities.

Let $J \subseteq \{1, \ldots, n\}$ be the set of all tasks that have not been included in the previously constructed blocks. In constructing a new block, only these tasks are considered and all tasks which have been already included in the previously constructed blocks are ignored. The construction of the first block and some other blocks will require calculating for each task $i \in J$ a priority μ_i using the following algorithm.

μ-algorithm

1. Set $a = 1$.
2. Among all tasks from J, which have not been assigned their priorities, select a task with the largest due date. Denote this due date by d. If several tasks from J, which have not been assigned their priorities, have due dates equal to d, select among them any task without successors. If every task $i \in J$ with $d_i = d$, which has not been assigned μ_i, has a successor, select among these tasks any task with the smallest in the lexicographical order vector $\omega(i,d)$. Let u be the selected task.
3. Set $\mu_u = a$ and $a = a + 1$. If there is a task in J, which has not been assigned its priority, go to step 2. Otherwise halt.

The idea behind the algorithm, constructing an ideal schedule, can be described as follows. Suppose that we know that for any instance of the considered scheduling problem there exists an ideal schedule. Since any ideal schedule is optimal for both C_{max} and C_Σ, even without knowing any ideal schedule, it is not difficult to find integers d_j such that

$$C_j(\nu) \leq d_j, \qquad \text{for some ideal schedule } \nu \text{ and all } j \in N \tag{5}$$

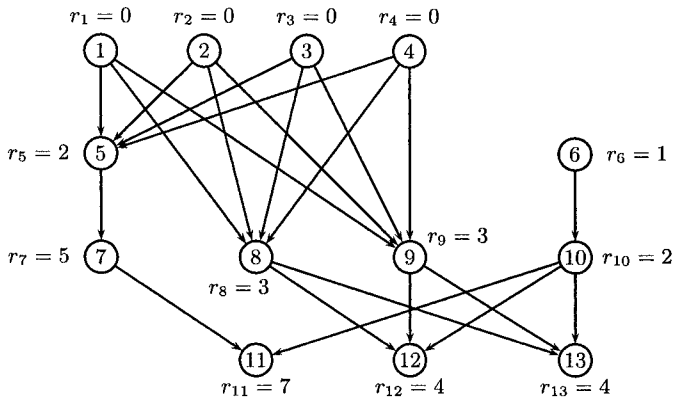

Figure 3. Partially ordered set of tasks.

By (5), the set of schedules that meet the due dates d_j is not empty. Therefore, using Garey and Johnson (1977), it is possible to construct a schedule, say σ, such that $C_j(\sigma) \leq d_j$ for all $j \in N$. If due dates d_j are sufficiently tight, σ coincides with an ideal schedule. The Main Algorithm, described below, is an iterative procedure that at each iteration either tightens the current due dates, or determines that these due dates are already tight enough and specifies a fragment of an ideal schedule.

The following example illustrates this idea. Consider the partially ordered set of tasks depicted in Figure 3, where nodes represent tasks and arrows represent precedence constraints. The corresponding release time is given next to each node.

It can be shown that for the considered problem there exists an ideal schedule (a proof that such a schedule exists for any instance of the problem with two parallel identical machines, precedence constraints, and unit execution time tasks with release times is given in the next section and is closely related to the analysis of the Main Algorithm). Denote this ideal schedule by ν. Observe that the partially ordered set of tasks is comprised of 13 tasks. Taking into account that ν is optimal for both C_{max} and C_Σ, it is easy to see that (5) holds if $d_j = r_j + 13$ for all $1 \leq j \leq 13$. The priorities μ_j, calculated for these d_j according to the μ-algorithm, are presented in the table below.

task j	1	2	3	4	5	6	7	8	9	10	11	12	13
μ_j	13	12	11	10	7	9	2	5	6	8	1	3	4

For example, we have $d_5 = 15$ and $d_{10} = 15$. Since according to the precedence constraints both 5 and 10 have successors, in order to determine μ_5 and

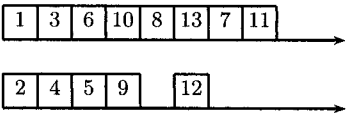

Figure 4.

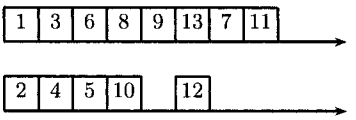

Figure 5.

μ_{10} we need to compare vectors $\omega(5, 15)$ and $\omega(10, 15)$. Because $\omega(5, 15) = (2, 1)$ is smaller in the lexicographical order than $\omega(10, 15) = (4, 3, 1)$, $\mu_5 = 7$ whereas $\mu_{10} = 8$.

Consider the list schedule σ, corresponding to the list 1, 2, 3, 4, 6, 10, 5, 9, 8, 13, 12, 7, 11, where tasks are arranged in the decreasing order of μ_j. Schedule σ is depicted in Figure 4.

The smallest integer $0 < t \leq C_{max}(\sigma)$ such that at most one task is processed in the time interval $[t-1, t]$ is $t = 5$. By the list algorithm, each task j, satisfying the inequality $C_8(\sigma) < C_j(\sigma)$, either is a successor of task 8 in the precedence constraints or has $r_j \geq 5$. Therefore, in any schedule, where task 8 has the completion time greater than $C_8(\sigma)$, only tasks 1, 2, 3, 4, 6, 5, 10 and 9 can be processed in the time interval $[0, C_8(\sigma)]$. Hence, this schedule cannot be optimal for the criterion C_Σ, because in σ task 8 is also processed in this time interval. Therefore $C_8(\nu) \leq C_8(\sigma) = 5$, and the replacement of $d_8 = 16$ by $d_8 = 5$ does not violate the inequality $C_8(\nu) \leq d_8$.

The table below presents consistent due dates D_j, calculated according to Garey and Johnson (1977) for the new set of due dates $\{d_j : 1 \leq j \leq 13\}$ that differs from the original one only by the value of d_8 which now equals 5. By Theorem 2, $C_j(\nu) \leq D_j$ for all $1 \leq j \leq 13$. The above-mentioned table also contains new μ_j, calculated according to the μ-algorithm for the set of consistent due dates $\{D_j : 1 \leq j \leq 13\}$.

task j	1	2	3	4	5	6	7	8	9	10	11	12	13
D_j	4	4	4	4	15	14	18	5	16	15	20	17	17
μ_j	13	12	11	10	6	8	2	9	5	7	1	3	4

Again, denote by σ the list schedule, corresponding to the list where tasks are arranged in the decreasing order of μ_j. This schedule is depicted in Figure 5.

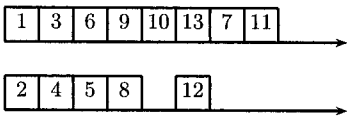

Figure 6.

Using σ and reasoning analogous to that presented above for the task 8, it is easy to see that $C_9(\nu) \leq C_9(\sigma) = 5$. Therefore, the set of due dates $\{d_j : 1 \leq j \leq 13\}$, where $d_j = D_j$ for all $j \neq 9$ and $d_9 = 5$, satisfies (5). The table below presents consistent due dates $\{D_j : 1 \leq j \leq 13\}$, calculated according to Garey and Johnson (1977) for this new set $\{d_j : 1 \leq j \leq 13\}$. By Theorem 2, these new consistent due dates satisfy the inequalities $C_j(\nu) \leq D_j$ for all $1 \leq j \leq 13$. Analogously to the previous iteration, the table below also contains new μ_j, calculated according to the μ-algorithm for the new set of consistent due dates $\{D_j : 1 \leq j \leq 13\}$.

task j	1	2	3	4	5	6	7	8	9	10	11	12	13
D_j	4	4	4	4	15	14	18	5	5	15	20	17	17
μ_j	13	12	11	10	5	7	2	8	9	6	1	3	4

Again, let σ be the list schedule, corresponding to the list where tasks are arranged in the decreasing order of μ_j. This schedule is depicted in Figure 6.

Analogously to the above, σ indicates that $C_{10}(\nu) \leq C_{10}(\sigma) = 5$. Therefore, the set of due dates $\{d_j : 1 \leq j \leq 13\}$, where $d_j = D_j$ for all $j \neq 10$ and $d_{10} = 5$, satisfies (5). The new set $\{d_j : 1 \leq j \leq 13\}$ leads to a new set of consistent due dates $\{D_j : 1 \leq j \leq 13\}$, calculated according to Garey and Johnson (1977). These new due dates D_j are presented in the following table.

task j	1	2	3	4	5	6	7	8	9	10	11	12	13
D_j	4	4	4	4	15	4	18	5	5	5	20	17	17

By Theorem 2, these new consistent due dates satisfy the inequalities $C_j(\nu) \leq D_j$ for all $1 \leq j \leq 13$. In other words, now we know that $C_8(\nu) \leq 5$, $C_9(\nu) \leq 5$, $C_{10}(\nu) \leq 5$, $C_1(\nu) \leq 4$, $C_2(\nu) \leq 4$, $C_3(\nu) \leq 4$, $C_4(\nu) \leq 4$ and $C_6(\nu) \leq 4$. In order to decide whether these consistent due dates are tight enough to determine a fragment of ν, we need to calculate new priorities μ_j and to construct a new list schedule σ. Using $\{D_j : 1 \leq j \leq 13\}$, the μ-algorithm first assigns $\mu_{11} = 1$, and then, in the decreasing order of D_j, $\mu_7 = 2$, $\mu_{12} = 3$, $\mu_{13} = 4$ and $\mu_5 = 5$. Since the next three tasks, 10, 9 and 8, have equal due dates and since $\omega(10,5) = \omega(9,5) = \omega(8,5)$, the priorities 6, 7 and 8 can be assigned to these three tasks in any order. Let $\mu_{10} = 6$, $\mu_9 = 7$, and $\mu_8 = 8$. Now tasks 1, 2, 3, 4 and 6 have equal due dates, but $\omega(1,4) = \omega(2,4) = \omega(3,4) = \omega(4,4) = (8,7,5)$ whereas $\omega(6,4) = (6)$.

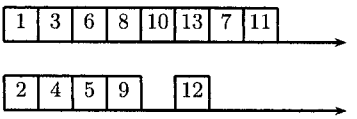

Figure 7.

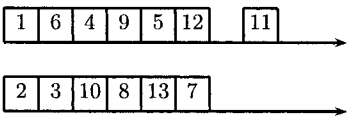

Figure 8.

Because in the lexicographical order (6) is smaller than $(8, 7, 5)$, $\mu_6 = 9$. The priorities 10, 11, 12, 13 can be assigned to the remaining tasks 1, 2, 3 and 4 in any order. Let $\mu_4 = 10$, $\mu_3 = 11$, $\mu_2 = 12$ and $\mu_1 = 13$.

Again, let σ be the list schedule, corresponding to the list where tasks are arranged in the decreasing order of μ_j. This schedule is depicted in Figure 7.

Now $C_{10}(\sigma) = d_{10} = 5$ and we cannot tighten d_{10}. Consider the largest integer $0 < t < C_{10}(\sigma)$, satisfying the condition that there exists a task j such that $C_j(\sigma) = t$ and $\mu_j < \mu_{10}$. It is easy to see that $t = 3$ and $j = 5$. Observe that for $i \in \{8, 9, 10\}$ either $6 \to i$ or $r_i \geq 3$. Hence, in any schedule, where task 6 has the completion time greater than $C_6(\sigma) = 3$, tasks 8–10 are processed in the time interval $[3, \infty]$ and therefore tasks 7, 11, 12 and 13 are processed in the time interval $[5, \infty]$ (see Figures 4–7). Hence, in this schedule only tasks 1–5 can be processed in the time interval $[0, C_6(\sigma)]$. Such a schedule cannot be optimal for the criterion C_Σ, because in σ task 6 is also processed in this time interval. Therefore, $C_6(\nu) \leq C_6(\sigma) = 3$ and the set of due dates $\{d_j : 1 \leq j \leq 13\}$, where $d_j = D_j$ for all $j \neq 6$ and $d_6 = 3$, satisfies (5).

Now the due dates $\{d_j : 1 \leq j \leq 13\}$ are tight enough. Indeed, consider the list schedule, depicted in Figure 8 and corresponding to the list, where tasks are arranged in a nondecreasing order of d_j. It is obvious that the part of this schedule, corresponding to the time interval $[0, 7]$, is a fragment of an ideal schedule (actually, in this example, the entire schedule, obtained in the process of constructing the first fragment, is ideal).

Reasoning in the above example was based on the assumption that an ideal schedule exists. As will be shown in the next section, this assumption is not necessary and can be replaced by two weaker conditions. The remaining part of this section presents the Main Algorithm, which constructs an ideal schedule block by block. The construction of each block commences with determining the starting time of this block, denoted by τ. So, in the above example, we

started with $\tau = 0$. The parameter τ increases from iteration to iteration. So, if τ' and τ'' are two successive values of τ, then the block is a fragment of δ corresponding to the time interval $[\tau', \tau'']$. In the above example, the next value of τ is 7. In what follows, for an arbitrary schedule α and an arbitrary integer t, $R(\alpha, t)$ will denote the set of all tasks i such that $C_i(\alpha) \leq t$.

Main Algorithm

1. Set $J = \{1, \ldots, n\}$, $\tau = \min_{1 \leq i \leq n} r_i$, and $d_i = r_i + n$, for all $1 \leq i \leq n$.

2. Using the procedure from Garey and Johnson (1977) and the set $\{d_i : i \in J\}$, determine the set of consistent due dates $\{D_i : i \in J\}$ such that $D_i \leq d_i$ for each $i \in J$.

3. Using the set of consistent due dates $\{D_i : i \in J\}$ and the μ-algorithm, calculate μ_i for every task $i \in J$. Arrange tasks, constituting the set J, in a list in the decreasing order of their priorities and construct from time point τ a list schedule σ.

4. Select the smallest integer $\tau < t \leq C_{max}(\sigma)$ such that at most one task is processed in the time interval $[t-1, t]$. Denote this integer by $t(\sigma, \tau)$. If $t(\sigma, \tau)$ does not exist or $t(\sigma, \tau) = C_{max}(\sigma)$, then construct the last block of δ by setting $C_i(\delta) = C_i(\sigma)$ for each task $i \in J$ and halt.

5. Among all tasks i satisfying the inequality $C_i(\sigma) > t(\sigma, \tau)$ select a task with the earliest release time and denote this release time by r. If $r \geq t(\sigma, \tau)$, then construct a new block of δ by setting $C_i(\delta) = C_i(\sigma)$ for each task i, satisfying the inequalities $\tau < C_i(\sigma) \leq t(\sigma, \tau)$. This block corresponds to the time interval $[\tau, r]$. Set $\tau = r$, $J = J - R(\delta, r)$ and return to step 4.

6. Set $i(\tau) = j$, where j is the task satisfying the equality $C_j(\sigma) = t(\sigma, \tau)$.

7. If $D_{i(\tau)} > C_{i(\tau)}(\sigma)$, then set $d_{i(\tau)} = C_{i(\tau)}(\sigma)$ and $d_k = D_k$, for all other tasks $k \in J$, and return to step 2. If $D_{i(\tau)} = C_{i(\tau)}(\sigma)$, then select the largest integer $\tau < t < C_{i(\tau)}(\sigma)$ such that at least one task, which is processed in the time interval $[t-1, t]$, has priority less than $\mu_{i(\tau)}$. Denote this integer by t'. If either t' does not exist, or both tasks processed in the time interval $[t'-1, t']$ have priorities less than $\mu_{i(\tau)}$, then construct a new block of δ by setting $C_i(\delta) = C_i(\sigma)$ for each task i, satisfying the inequalities $\tau < C_i(\sigma) \leq t(\sigma, \tau)$. This block corresponds to the time interval $[\tau, t(\sigma, \tau)]$. Set $J = J - R(\delta, t(\sigma, \tau))$. Assign a new release time to every $i \in J$ as the maximum between the old release time and $t(\sigma, \tau)$. Recalculate, if necessary, the release times of all tasks

to satisfy the condition that the relation $i \rightarrow j$ implies the inequality $r_i \leq r_j - 1$. In doing this, consider the release times in increasing order and for any i and j such that $i \rightarrow j$ and $r_i = r_j$ replace r_j by $r_j + 1$. Set $\tau = t(\sigma, \tau)$ and return to step 4.

8. Let task j be the task satisfying the conditions $C_j(\sigma) = t'$ and $\mu_j > \mu_{i(\tau)}$. Set $i(\tau) = j$ and return to step 7.

The validity of the algorithm is established by the following lemma.

Lemma 1 *For any set $\{d_i : i \in J\}$, utilized at step 2, the procedure from Garey and Johnson (1977) constructs a set of consistent due dates $\{D_i : i \in J\}$ satisfying the inequalities $D_i \leq d_i$ for all $i \in J$.*

Proof: By Theorem 1, if for some set of due dates $\{d_i : i \in J\}$ the procedure from Garey and Johnson (1977) fails to determine the set of consistent due dates, then there is no schedule α for the set J satisfying the inequalities $C_i(\alpha) \leq d_i$ for all $i \in J$. Hence, if such a schedule exists, then the procedure from Garey and Johnson (1977) is able to calculate a set of consistent due dates, and it remains to show that for each set of integers $\{d_i : i \in J\}$, utilized at step 2, there exists a schedule α such that $C_i(\alpha) \leq d_i$ for all $i \in J$. It is easy to see that this holds at the first execution of step 2. Suppose that a set of consistent due dates $\{D_i : i \in J\}$ has been determined in accord with step 2 using a set of integers $\{d_i : i \in J\}$. The schedule σ, constructed as a result of the subsequent execution of step 3, is a list schedule corresponding to a list where tasks are arranged in a nondecreasing order of consistent due dates. Then by Theorem 3, $C_i(\sigma) \leq D_i$ for all $i \in J$, and hence at the next execution of step 2, $C_i(\sigma) \leq d'_i$ for all $i \in J'$, where J' is the set of tasks and $\{d'_i : i \in J'\}$ is the set of due dates considered at this execution of step 2. □

2.2 Proof of Optimality

The execution of any step of the main algorithm, apart from the very first, starts with some value of the parameter τ, some set of tasks J, and some set of due dates $\{d_i : i \in J\}$ assigned either during the execution of step 1 or step 7. For the purpose of the following discussion, it is convenient to introduce the notion of a regular call. In conditions (c1) and (c2), stated below, τ is the value of this parameter at the start of execution of the considered step, J is the set of tasks which have not been included in the blocks constructed prior to this call, and $\{d_i : i \in J\}$ are due dates associated with tasks from J at the moment of this call. A call of a step is regular if

(c1) there exists a schedule η for the set N, which is optimal for the criterion C_{max}, coincides with δ on the time interval $[\min_{1 \leq i \leq n} r_i, \tau]$, and satisfies inequalities $C_i(\eta) \leq d_i$ for all $i \in J$;

(c2) there exists a schedule β for the set N, which is optimal for the criterion C_Σ, coincides with δ on the time interval $[\min_{1\le i\le n} r_i, \tau]$, and satisfies inequalities $C_i(\beta) \le d_i$ for all $i \in J$.

Lemma 2 *If the Main Algorithm terminates at a regular call of step 4, then δ is an ideal schedule.*

Proof: According to (c1) and (c2), schedule δ coincides with β and η in the time interval $[\min_{1\le i\le n} r_i, \tau]$, and these two schedules are optimal for the criteria C_Σ and C_{max}, respectively. The tasks, which are not processed in this time interval, form the set J and are processed in the current schedule σ in the most efficient manner from the viewpoint of both criteria, since in this schedule both machines are busy in the time interval $[\tau, C_{max}(\sigma) - 1]$. Therefore, the resulting schedule δ is optimal for both criteria C_Σ and C_{max}. □

The following lemmas show that any regular call results in another regular call.

Lemma 3 *If a call of step 5 is regular, then the next call is also regular.*

Proof: If the next call is a call of step 6, then this new call is also regular, because in this case neither value of τ, nor sets J and $\{d_i : i \in J\}$ are changed.

Suppose that the call of step 5 results in a call of step 4, and let τ, J and $\{d_i : i \in J\}$ be the value of the parameter τ, the set of tasks and the set of due dates corresponding to this call of step 5. Let J' be the set of tasks which remain not included in the already constructed blocks after the completion of step 5. Since the call of step 5 is regular, there exists a schedule β, specified in (c2). Construct a new schedule β' by setting

$$C_i(\beta') = \begin{cases} C_i(\sigma), & \text{for all } \ i \in R(\sigma, t(\sigma,\tau)) \\ C_i(\beta), & \text{for all } \ i \notin R(\sigma, t(\sigma,\tau)) \end{cases}$$

where $t(\sigma,\tau)$ and σ are the point in time and the list schedule considered during the execution of step 5. Since $r_i \ge t(\sigma,\tau)$ for all $i \in J - R(\sigma, t(\sigma,\tau))$,

$$R(\beta, t(\sigma,\tau)) \subseteq R(\delta,\tau) \cup R(\sigma, t(\sigma,\tau)) = R(\beta', t(\sigma,\tau))$$

This relation together with the observation that in β' both machines are occupied in the time interval $[\tau, t(\sigma,\tau) - 1]$ leads to the conclusion that schedule β' is optimal for the criterion C_Σ.

On the other hand, step 5 does not change the set of due dates $\{d_i : i \in J\}$, and the inequalities $C_i(\beta) \le d_i$ and $C_i(\sigma) \le d_i$, which hold for all $i \in J$, imply that $C_i(\beta') \le d_i$ for all $i \in J'$. Hence, (c2) holds after the completion of step 5.

Let η be a schedule specified in (c1). Then the result that (c1) holds after the completion of step 5 can be proven analogously to the one above by constructing a new schedule η', where

$$C_i(\eta') = \begin{cases} C_i(\sigma), & \text{for all } \ i \in R(\sigma, t(\sigma,\tau)) \\ C_i(\eta), & \text{for all } \ i \notin R(\sigma, t(\sigma,\tau)) \end{cases}$$

So, the call of step 4 is a regular one. □

Suppose that a regular call of step 7 is followed by either a call of step 2 or a call of step 4. Suppose that this call of step 7 corresponds to some τ, J, $\{d_i : i \in J\}$, $\{D_i : i \in J\}$, σ, and $i(\tau)$.

Lemma 4 *Let α be an arbitrary schedule for J such that $C_v(\alpha) \leq d_v$, for all $v \in J$, and let α satisfy at least one of the following two conditions: either $C_{i(\tau)}(\alpha) \geq C_{i(\tau)}(\sigma)$, or there exists $x \in J$ such that $\mu_x > \mu_{i(\tau)}$ and $C_x(\alpha) = D_{i(\tau)}$. Then*

$$R(\alpha, t(\sigma,\tau)) \subseteq R(\sigma, t(\sigma,\tau)) \tag{6}$$

Proof: Suppose that the considered call of step 7 is the kth call of this step since the last call of step 6. Consider a sequence of tasks $g_1, \ldots, g_k$ and a sequence of sets of tasks $J^0, J^1, \ldots, J^k$ determined as follows. Task g_1 satisfies the equality $C_{g_1}(\sigma) = t(\sigma,\tau)$. Each task g_i, $1 < i \leq k$, is the task selected as a result of the $(i-1)$st call of step 8. Set J^0 is the set of all tasks i satisfying the inequality $C_i(\sigma) > C_{g_1}(\sigma)$. Each J^i, $1 \leq i < k$, is comprised of task g_i and all tasks v such that $C_{g_{i+1}}(\sigma) < C_v(\sigma) < C_{g_i}(\sigma)$. Set J^k is defined as follows. If $C_{g_k}(\sigma) < D_{g_k}$, then $J^k = \{g_k\}$. If $C_{g_k}(\sigma) = D_{g_k}$, then J^k depends on whether or not t' has been calculated as a result of the kth call of step 7. If t' has been calculated, then J^k is comprised of task g_k and all tasks v such that $t' < C_v(\sigma) < C_{g_k}(\sigma)$. If t' has not been calculated, then J^k is comprised of task g_k and all tasks v such that $\tau < C_v(\sigma) < C_{g_k}(\sigma)$.

Consider a sequence of tasks $y_1, \ldots, y_k$, where for all $1 \leq i \leq k$

$$C_{y_i}(\alpha) = \max_{v \in J^i} C_v(\alpha)$$

For any task y_i, D_{y_i} and the corresponding D_{g_i} satisfy the inequality $D_{y_i} \leq D_{g_i}$, because $\mu_{y_i} \geq \mu_{g_i}$. On the other hand, $C_v(\alpha) \leq d_v$ for all $v \in J$, and therefore by Theorem 2, $C_{y_i}(\alpha) \leq D_{y_i}$. Hence, $C_{y_i}(\alpha) \leq D_{g_i}$ for all $1 \leq i \leq k$.

Observe that $g_1 \rightarrow v$ for all $v \in J^0$ such that $r_v < C_{g_1}(\sigma)$, because σ is a list schedule and one of the machines is idle during the time interval $[C_{g_1}(\sigma) - 1, C_{g_1}(\sigma)]$. Moreover, for all $1 < i \leq k$, $g_i \rightarrow v$ for all $v \in J^{i-1}$ such that $r_v < C_{g_i}(\sigma)$, because the list corresponding to σ arranges tasks in the decreasing

order of their priorities and during the time interval $[C_{g_i}(\sigma) - 1, C_{g_i}(\sigma)]$ one of the machines processes a task with a priority less then priorities of tasks in J^{i-1}. Therefore, if $C_{g_i}(\alpha) \geq C_{g_i}(\sigma)$, then

$$\min_{v \in J^{i-1}} C_v(\alpha) \geq \min_{v \in J^{i-1}} C_v(\sigma) \tag{7}$$

Suppose that there exists a task q such that $\mu_q > \mu_{g_i}$ and $C_q(\alpha) = D_{g_i}$. By Theorem 2, $C_q(\alpha) \leq D_q$. The inequality $\mu_q > \mu_{g_i}$ implies that $D_q \leq D_{g_i}$, which together with $D_{g_i} = C_q(\alpha) \leq D_q$ gives $D_q = D_{g_i}$. Since $D_q = D_{g_i}$ and $\mu_q > \mu_{g_i}$, $\omega(q, D_q)$ is greater than or equal to $\omega(g_i, D_{g_i})$ in the lexicographical order. This is equivalent to the statement that $q \rightarrow v$ for all $v \in J^{i-1}$ such that $r_v < C_{g_i}(\sigma)$, which again gives (7).

The above reasonings lead to the conclusion that (6) holds if $k = 1$, because in this case $g_1 = i(\tau)$ and consequently either $C_{g_1}(\alpha) \geq C_{g_1}(\sigma)$, or $C_x(\alpha) = D_{g_1}$. Suppose that $k > 1$, then $C_{g_i}(\alpha) = D_{g_i}$ for all $1 \leq i < k$ and $g_k = i(\tau)$. Since either $C_{g_k}(\alpha) \geq C_{g_k}(\sigma)$, or there exists $x \in J$ such that $\mu_x > \mu_{g_k}$ and $C_x(\alpha) = D_{g_k}$,

$$\min_{v \in J^{k-1}} C_v(\alpha) \geq \min_{v \in J^{k-1}} C_v(\sigma)$$

Therefore,

$$D_{g_{k-1}} = C_{g_{k-1}}(\sigma) \leq C_{y_{k-1}}(\alpha) \leq D_{g_{k-1}},$$

which gives $C_{y_{k-1}}(\alpha) = D_{g_{k-1}}$. On the other hand, for any $1 < i < k$, either $y_i = g_i$, or $y_i \neq g_i$ and $\mu_{y_i} > \mu_{g_i}$. Therefore the equality $C_{y_i}(\alpha) = D_{g_i}$ implies (7), which analogously to the above gives $C_{y_{i-1}}(\alpha) = D_{g_{i-1}}$. Consequently, $C_{y_{k-1}}(\alpha) = D_{g_{k-1}}$ implies $C_{y_1}(\alpha) = D_{g_1}$, which in turn gives

$$\min_{v \in J^0} C_v(\alpha) \geq \min_{v \in J^0} C_v(\sigma)$$

and therefore (6). □

Lemma 5 *If a call of step 7 is regular, then the next call is also regular.*

Proof: Let a regular call of step 7 correspond to some τ, J, $\{d_i : i \in J\}$, $\{D_i : i \in J\}$, $i(\tau)$ and σ. Since this call is regular, there exist schedules η and β specified in (c1) and (c2). If this call is followed by a call of step 8, then the latter call is also regular, because in this case neither the parameter τ, nor the sets J and $\{d_i : i \in J\}$ change during the execution of step 7.

Suppose that the call of step 7 is followed by a call of step 2. Then step 7 results in the replacement of $\{d_i : i \in J\}$ by due dates $\{d'_i : i \in J\}$, where $d_{i(\tau)} = C_{i(\tau)}(\sigma)$ and $d_v = D_v$ for all other $v \in J$. If $C_{i(\tau)}(\eta) \leq C_{i(\tau)}(\sigma)$ and $C_{i(\tau)}(\beta) \leq C_{i(\tau)}(\sigma)$, then $C_{i(\tau)}(\eta) \leq d'_{i(\tau)}$ and $C_{i(\tau)}(\beta) \leq d'_{i(\tau)}$. On the other hand, by Theorem 2, $C_v(\eta) \leq D_v = d'_v$ and $C_v(\beta) \leq D_v = d'_v$ for all other $v \in J$. Hence, in this case, the call of step 2 is also regular.

If $C_{i(\tau)}(\beta) > C_{i(\tau)}(\sigma)$, then consider schedule β', where

$$C_i(\beta') = \begin{cases} C_i(\sigma), & \text{for all } \ i \in R(\sigma, t(\sigma,\tau)) \\ C_i(\beta), & \text{for all } \ i \notin R(\sigma, t(\sigma,\tau)) \end{cases} \tag{8}$$

By Lemma 4

$$J \cap R(\beta, t(\sigma,\tau)) \subseteq R(\sigma, t(\sigma,\tau)) \subseteq R(\beta', t(\sigma,\tau))$$

Since in σ both machines are busy in the interval $[\tau, t(\sigma,\tau) - 1]$, $C_\Sigma(\beta') \leq C_\Sigma(\beta)$. Therefore, schedule β' is optimal for the criterion C_Σ. On the other hand, $C_{i(\tau)}(\sigma) = d'_{i(\tau)}$, and by Theorem 3, $C_i(\sigma) \leq D_i = d'_i$ for all other $i \in R(\sigma, t(\sigma,\tau))$. Furthermore, by Theorem 2, $C_i(\beta) \leq D_i = d'_i$ for all $i \in J$ such that $i \notin R(\sigma, t(\sigma,\tau))$. Therefore, $C_v(\beta') \leq d'_v$ for all $v \in J$, and β' satisfies (c2) for $\{d'_v : v \in J\}$.

Similarly, if $C_{i(\tau)}(\eta) > C_{i(\tau)}(\sigma)$, then, at the moment of the call of step 2, (c1) holds for schedule η', where

$$C_i(\eta') = \begin{cases} C_i(\sigma), & \text{for all } \ i \in R(\sigma, t(\sigma,\tau)) \\ C_i(\eta), & \text{for all } \ i \notin R(\sigma, t(\sigma,\tau)) \end{cases} \tag{9}$$

Hence, the call of step 2 is regular.

Suppose that the call of step 7 is followed by a call of step 4. Consider schedule β' defined by (8). Let J' be the set of all tasks v satisfying the inequality $C_v(\beta') > t(\sigma,\tau)$. Since β satisfies (c2), $C_v(\beta') \leq d_v$ for all $v \in J'$, and in order to prove that β' satisfies (c2) at the moment of the call of step 4, it remains to show that β' is optimal for the criterion C_Σ. Let

$$T = \begin{cases} t', & \text{if } t' \text{ has been found during the execution of step 7} \\ \tau, & \text{otherwise} \end{cases}$$

Consider the set of tasks $\tilde{J}$ which is comprised of $i(\tau)$ and all tasks v such that $T < C_v(\sigma) < C_{i(\tau)}(\sigma)$. It is easy to see that $C_v(\beta) > T$ for all $v \in \tilde{J}$. On the other hand, for any $v \in \tilde{J}$, $\mu_v \geq \mu_{i(\tau)}$ and therefore $D_v \leq D_{i(\tau)}$. Since, by Theorem 2, $C_v(\beta) \leq D_v$, all tasks constituting $\tilde{J}$ are processed in β in the time interval $[T, D_{i(\tau)}]$. Hence $C_v(\beta) = D_{i(\tau)}$ for some $v \in \tilde{J}$. Then again by Lemma 4

$$J \cap R(\beta, t(\sigma,\tau)) \subseteq R(\sigma, t(\sigma,\tau)) \subseteq R(\beta', t(\sigma,\tau))$$

and because both machines are busy in the time interval $[\tau, t(\sigma,\tau) - 1]$, so $C_\Sigma(\beta') \leq C_\Sigma(\beta)$, which implies that β' is optimal for the criterion C_Σ. Thus, at the moment of the call of step 4, β' satisfies (c2). The proof that η', defined by (9), satisfies (c1) is similar. Hence, the call of step 4 is regular. □

Theorem 4 *The procedure constructs an ideal schedule and has complexity* $O(n^6)$.

Proof: Each block of schedule δ corresponds to a particular value of the parameter τ and this parameter takes on at most n different values. Each d_i satisfies inequalities $r_i < d_i \leq r_i + n$ and if its value changes, then a new value is at least one unit less than the previous one. Therefore, the number of calls of step 2 is $O(n^2)$. Since the calculation of consistent due dates according to step 2 requires $O(n^3)$ operations, the total complexity is $O(n^6)$.

It is easy to see that the first call of steps 2 is regular. On the other hand, Lemmas 3 and 5 guarantee that all subsequent calls are regular too. Therefore, by Lemma 2, the resulting schedule is ideal. □

3. COMPUTATIONAL COMPLEXITY OF THE $P2|in\text{-}tree, p_j = 1, size_j|C_\Sigma$ PROBLEM

This section shows that the problem $P2|in\text{-}tree, p_j = 1, size_j|C_\Sigma$ is NP-hard in the strong sense by a reduction of the 3-partition problem which is NP-complete in the strong sense (Garey and Johnson, 1979). The 3-partition problem can be stated as follows:

> *Instance:* A set of positive integers $\mathcal{A} = \{a_1, \ldots, a_{3z}\}$ together with a positive integer b such that $\sum_{i=1}^{3z} a_i = zb$ and $\frac{b}{4} < a_i < \frac{b}{2}$, for all $i \in \{1, \ldots, 3z\}$.
>
> *Question:* Does there exist a partition of the set $\mathcal{A}$ into subsets $\mathcal{A}_1, \ldots, \mathcal{A}_z$, such that $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for all $i \neq j$, and the sum of the elements of each subset is equal to b?

For any instance of the 3-partition problem, the corresponding scheduling decision problem will contain $3z$ chains (one for each integer in $\mathcal{A}$), an in-tree, and a set of 2-tasks. Following Brucker *et al.* (2000), a task j requiring $size_j$ machines is referred to as a $size_j$-task. Analogously to Zinder *et al.* (submitted), for each integer $a_j \in \mathcal{A}$, the corresponding scheduling problem contains a chain of $2a_j$ tasks, where the first a_j tasks in the chain are 2-tasks, and the remaining tasks are 1-tasks. The chain for some $a_i \in \mathcal{A}$ and the in-tree are depicted in Figure 9. The set of a_j 2-tasks associated with a_j will be denoted by M_1^j. Similarly, the set of all remaining tasks associated with a_j will be denoted by M_2^j. Let $M_1 = \cup_{j=1}^{3z} M_1^j$ and $M_2 = \cup_{j=1}^{3z} M_2^j$. Thus, all tasks in M_1 are 2-tasks, whereas all tasks in M_2 are 1-tasks.

In contrast to Zinder *et al.* (submitted), where the out-tree is comprised of 1-tasks only, the in-tree contains both 1-tasks and 2-tasks. It is convenient to group these tasks into several sets denoted K_v^i, where K_1^i for $1 \leq i \leq z$, K_2^i for $1 \leq i \leq z-1$, and K_3^i for $1 \leq i \leq z-1$ consist of 1-tasks, and K_4^i for

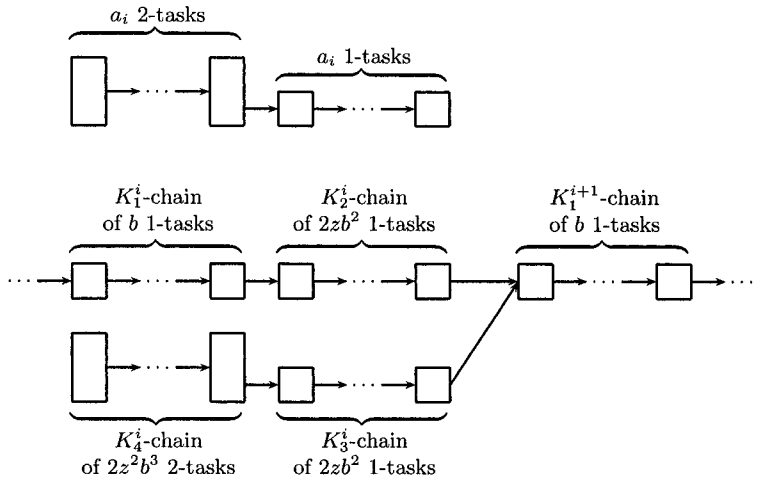

Figure 9.

$1 \leq i \leq z-1$ consist of 2-tasks. These sets have the following cardinalities: $|K_1^i| = b$, $|K_2^i| = |K_3^i| = 2zb^2$, $|K_4^i| = 2z^2b^3$. The tasks constituting each set form a chain. These chains will be referred to as K_v^i-chains and are linked as follows. For each $1 \leq i \leq z-1$

- the last task in the K_1^i-chain precedes the first task in the K_2^i-chain;
- the last task in the K_4^i-chain precedes the first task in the K_3^i-chain;
- the first task in the K_1^{i+1}-chain has two immediate predecessors: the last task in the K_2^i-chain and the last task in the K_3^i-chain.

Let $K_1 = \cup_{i=1}^{z} K_1^i$, $K_2 = \cup_{i=1}^{z-1} K_2^i$, $K_3 = \cup_{i=1}^{z-1} K_3^i$, and $K_4 = \cup_{i=1}^{z-1} K_4^i$, and let K_5 be the set of 2-tasks, where

$$|K_5| = 2 \sum_{j=1}^{|M_1|+|K_1|+|K_2|+|K_4|} j$$

and each task of the set K_5 does not precede nor succeed any other task. Then the set of all tasks is $N = M_1 \cup M_2 \cup K_1 \cup K_2 \cup K_3 \cup K_4 \cup K_5$.

Let $\widehat{\sigma}$ be a schedule, probably infeasible, satisfying the following conditions:

(t1) for each $1 \leq i \leq z$, b tasks from M_1 are processed in the time interval $[(2b + 2zb^2 + 2z^2b^3)(i-1), (2b + 2zb^2 + 2z^2b^3)(i-1) + b]$;

(t2) for each $1 \leq i \leq z$, b tasks from K_1^i and b tasks from M_2 are processed in parallel in the time interval $[(2b + 2zb^2 + 2z^2b^3)(i-1) + b, (2b + 2zb^2 + 2z^2b^3)(i-1) + 2b]$;

(t3) for each $1 \leq i \leq z-1$, $2z^2b^3$ tasks from K_4^i are processed in the time interval $[(2b+2zb^2+2z^2b^3)(i-1)+2b, (2b+2zb^2+2z^2b^3)(i-1)+2b+2z^2b^3]$.

(t4) for each $1 \leq i \leq z-1$, $2zb^2$ tasks from K_2^i and $2zb^2$ tasks from K_3^i are processed in parallel in the time interval $[(2b + 2zb^2 + 2z^2b^3)(i-1) + 2b + 2z^2b^3, (2b + 2zb^2 + 2z^2b^3)i]$.

(t5) all tasks from K_5 are processed in the time interval $[(2b + 2zb^2 + 2z^2b^3)(z-1) + 2b, (2b + 2zb^2 + 2z^2b^3)(z-1) + 2b + |K_5|]$.

Let $C = \sum_{j \in N} C_j(\widehat{\sigma})$. Then the scheduling decision problem is a problem requiring to answer the question: does there exist a feasible schedule σ satisfying the inequality $C_\Sigma(\sigma) \leq C$?

In what follows, for any positive integer t the time interval $[t-1, t]$ will be called a time slot t.

Lemma 6 *For any schedule σ, optimal for the criterion C_Σ,*

$$\max_{j \in N} C_j(\sigma) = |M_1| + |K_1| + |K_2| + |K_4| + |K_5| \tag{10}$$

and

$$\max_{j \in N - K_5} C_j(\sigma) + 1 = \min_{j \in K_5} C_j(\sigma) \tag{11}$$

Proof: Let σ be an arbitrary optimal schedule. It follows from the optimality of σ that the time interval $[0, \max_{j \in N} C_j(\sigma)]$ does not contain any time slot in which both machines are idle. Suppose that σ does not satisfy (10). Since each task in K_5 is a 2-task and does not have any successor, σ can be transformed into a schedule η in which all tasks from $N - K_5$ are processed in the same order as in σ, all tasks from K_5 are processed after all tasks from $N - K_5$, and

$$\max_{j \in N} C_j(\eta) = \max_{j \in N} C_j(\sigma)$$

Because the set K_5 is comprised of 2-tasks, the optimality of σ implies the optimality of η, and since σ does not satisfy (10)

$$\max_{v \in N - K_5} C_v(\eta) \geq |M_1| + |K_1| + |K_2| + |K_4| + 1$$

It is easy to see that there exists a feasible schedule μ such that all 2-tasks comprising M_1 are processed in the time interval $[0, |M_1|]$; all 2-tasks constituting K_4 are processed in the time interval $[|M_1|, |M_1| + |K_4|]$; all 1-tasks

from $K_1 \cup K_2$ are processed in the time interval $[|M_1| + |K_4|, |M_1| + |K_4| + |K_1| + |K_2|]$ each in parallel with some task from $M_2 \cup K_3$; and all 2-tasks from K_5 are processed in the time interval $[|M_1| + |K_4| + |K_1| + |K_2|, |M_1| + |K_4| + |K_1| + |K_2| + |K_5|]$.

Because sets M_1 and K_4 are comprised of 2-tasks,

$$\sum_{v \in N - K_5} C_v(\mu) < \sum_{i=1}^{|M_1|+|K_1|+|K_2|+|K_4|} 2i = |K_5|$$

Since

$$\sum_{v \in K_5} C_v(\eta) = |K_5| \max_{v \in N - K_5} C_v(\eta) + 1 + 2 + \cdots + |K_5|$$

and

$$\sum_{v \in K_5} C_v(\mu) = |K_5| \max_{v \in N - K_5} C_v(\mu) + 1 + 2 + \cdots + |K_5|$$

we have

$$\begin{aligned} & C_\Sigma(\eta) - C_\Sigma(\mu) \\ & \quad = \sum_{v \in N - K_5} C_v(\eta) - \sum_{v \in N - K_5} C_v(\mu) + \sum_{v \in K_5} C_v(\eta) - \sum_{v \in K_5} C_v(\mu) \\ & \quad > -|K_5| + |K_5| \left(\max_{v \in N - K_5} C_v(\eta) - \max_{v \in N - K_5} C_v(\mu) \right) \geq 0 \end{aligned}$$

which contradicts the optimality of η, and therefore the optimality of σ.

Let σ be an optimal schedule, then (10) implies that every task from $K_1 \cup K_2$ is processed in σ in parallel with a task from $K_3 \cup M_2$. Hence, the time slot $\max_{j \in N - K_5} C_j(\sigma)$ contains two 1-tasks: the last task in the K_1^z-chain and a task from M_2. Since the set K_5 is comprised of 2-tasks, the optimality of σ implies that all these tasks are processed in σ after the time slot $\max_{j \in N - K_5} C_j(\sigma)$, which is equivalent to (11). □

In what follows, only schedules σ, satisfying (10) and (11), will be considered. For any two tasks j and g, let

$$\delta_{jg}(\sigma) = \begin{cases} 1 & \text{if} \quad C_j(\sigma) < C_g(\sigma) \text{ and } j \text{ is a 2-task} \\ \frac{1}{2} & \text{if} \quad C_j(\sigma) < C_g(\sigma) \text{ and } j \text{ is a 1-task} \\ 1 & \text{if} \quad j = g \\ 0 & \text{otherwise} \end{cases}$$

Let $\Delta_{22}(\sigma)$ be the sum of all $\delta_{jg}(\sigma)$, where both j and g are 2-tasks; $\Delta_{11}(\sigma)$ be the sum of all $\delta_{jg}(\sigma)$, where both j and g are 1-tasks; $\Delta_{12}(\sigma)$ be the sum of

all $\delta_{jg}(\sigma)$, where j is a 1-task and g is a 2-task; and $\Delta_{21}(\sigma)$ be the sum of all $\delta_{jg}(\sigma)$, where j is a 2-task and g is a 1-task.

Since all 1-tasks are processed in σ in pairs, for any $g \in N$,

$$C_g(\sigma) = \sum_{j \in N} \delta_{jg}(\sigma)$$

and consequently,

$$C_\Sigma(\sigma) = \sum_{g \in N} C_g(\sigma) = \sum_{g \in N} \sum_{j \in N} \delta_{jg}(\sigma) = \Delta_{22}(\sigma) + \Delta_{11}(\sigma) + \Delta_{12}(\sigma) + \Delta_{21}(\sigma)$$

It is easy to see that $\Delta_{22}(\sigma) + \Delta_{11}(\sigma)$ is a constant and does not depend on σ. Therefore, for any two schedules α and η, satisfying (10) and (11),

$$C_\Sigma(\alpha) - C_\Sigma(\eta) = \Delta_{12}(\alpha) - \Delta_{12}(\eta) + \Delta_{21}(\alpha) - \Delta_{21}(\eta)$$

Let j be a 1-task and g be a 2-task. Then

$$\delta_{jg}(\alpha) - \delta_{jg}(\eta) = -\frac{1}{2}\{\delta_{gj}(\alpha) - \delta_{gj}(\eta)\}$$

and therefore

$$C_\Sigma(\alpha) - C_\Sigma(\eta) = \frac{1}{2}\{\Delta_{21}(\alpha) - \Delta_{21}(\eta)\} \tag{12}$$

Let σ be an arbitrary schedule. Consider a new schedule η such that

- $C_j(\eta) = C_j(\sigma)$ for all $j \notin K_4 \cup K_3$;
- $\{C_j(\eta) : j \in K_4\} = \{C_j(\sigma) : j \in K_4\}$;
- $C_{j_1}(\eta) < C_{j_2}(\eta)$ for any two tasks $j_1 \in K_4^{i_1}$ and $j_2 \in K_4^{i_2}$ such that $i_1 < i_2$;
- $C_{j_1}(\eta) < C_{j_2}(\eta)$ for any two tasks j_1 and j_2 from the same set K_4^i such that $C_{j_1}(\sigma) < C_{j_2}(\sigma)$;
- $\{C_j(\eta) : j \in K_3\} = \{C_j(\sigma) : j \in K_3\}$;
- $C_{j_1}(\eta) < C_{j_2}(\eta)$ for any two tasks $j_1 \in K_3^{i_1}$ and $j_2 \in K_3^{i_2}$ such that $i_1 < i_2$;
- $C_{j_1}(\eta) < C_{j_2}(\eta)$ for any two tasks j_1 and j_2 from the same set K_3^i such that $C_{j_1}(\sigma) < C_{j_2}(\sigma)$.

It is easy to see that η is feasible and $C_\Sigma(\eta) = C_\Sigma(\sigma)$. In what follows, only schedules η satisfying the conditions

$$\max_{j \in K_4^i} C_j(\eta) < \min_{j \in K_4^{i+1}} C_j(\eta), \quad \text{for all } 1 \le i < z - 1 \tag{13}$$

and

$$\max_{j \in K_3^i} C_j(\eta) < \min_{j \in K_3^{i+1}} C_j(\eta), \quad \text{for all } 1 \leq i < z-1 \tag{14}$$

will be considered.

Let σ be an arbitrary schedule, and suppose that for some i

$$\max_{j \in K_4^i} C_j(\sigma) + 1 < \min_{j \in K_3^i} C_j(\sigma)$$

Let

$$C_{j_1}(\sigma) = \max_{j \in K_4^i} C_j(\sigma) \quad \text{and} \quad C_{j_2}(\sigma) = \min_{j \in K_3^i} C_j(\sigma)$$

Then consider a new schedule η, where

$$C_x(\eta) = \begin{cases} C_x(\sigma) & \text{if } C_x(\sigma) < C_{j_1}(\sigma) \text{ or } C_x(\sigma) \geq C_{j_2}(\sigma) \\ C_x(\sigma) - 1 & \text{if } C_{j_1}(\sigma) < C_x(\sigma) < C_{j_2}(\sigma) \\ C_{j_2}(\sigma) - 1 & \text{if } x = j_1 \end{cases}$$

Since j_1 is a 2-task which has only one immediate successor, task j_2, this schedule is feasible and $C_\Sigma(\eta) \leq C_\Sigma(\sigma)$. In what follows, only schedules η satisfying the condition

$$\max_{j \in K_4^i} C_j(\eta) + 1 = \min_{j \in K_3^i} C_j(\eta), \quad \text{for all } 1 \leq i < z-1 \tag{15}$$

will be considered.

Furthermore, let σ be an arbitrary schedule, and suppose that for some i there exist two tasks $j_1 \in K_4^i$ and $j_2 \in K_4^i$ such that $C_{j_1}(\sigma) + 1 < C_{j_2}(\sigma)$ and $j \notin K_4^i$ for any j satisfying the inequalities $C_{j_1}(\sigma) < C_j(\sigma) < C_{j_2}(\sigma)$. Then consider a new schedule η where

$$C_x(\eta) = \begin{cases} C_x(\sigma) & \text{if } C_x(\sigma) < C_{j_1}(\sigma) \text{ or } C_x(\sigma) \geq C_{j_2}(\sigma) \\ C_x(\sigma) - 1 & \text{if } C_{j_1}(\sigma) < C_x(\sigma) < C_{j_2}(\sigma) \\ C_{j_2}(\sigma) - 1 & \text{if } x = j_1 \end{cases}$$

Since j_1 is a 2-task and j_2 is its only immediate successor, it is obvious that η is a feasible schedule and $C_\Sigma(\eta) \leq C_\Sigma(\sigma)$. Similar reasoning can be used if tasks j_1 and j_2 belong instead of K_4^i to some M_1^i. Therefore, in what follows, only schedules η satisfying the conditions

$$\max_{j \in K_4^i} C_j(\sigma) - \min_{j \in K_4^i} C_j(\sigma) = |K_4^i| - 1, \quad \text{for all } 1 \leq i \leq z-1 \tag{16}$$

and

$$\max_{j \in M_1^i} C_j(\sigma) - \min_{j \in M_1^i} C_j(\sigma) = |M_1^i| - 1, \quad \text{for all } 1 \leq i \leq 3z \tag{17}$$

will be considered.

Lemma 7 *For any schedule η, optimal for the criterion C_Σ, each task from K_1 is processed in parallel with a task from M_2.*

Proof: Suppose that there exists an optimal schedule η which does not satisfy this lemma. Since η is an optimal schedule, by Lemma 6, each task from $K_1 \cup K_2$ is processed in this schedule in parallel with some task from the set $M_2 \cup K_3$. Consider the first time slot containing a task from K_1, say task $u \in K_1^h$, together with a task from K_3, say task v. Suppose that $h > 1$. Since $j \rightarrow g$ for each $j \in \cup_{1 \leq i < h} K_3^i$ and each $g \in K_1^h$, the processing of all tasks constituting $\cup_{1 \leq i < h} K_3^i$ is completed in η before the processing of tasks from K_1^h begins. Each $j \in \cup_{1 \leq i < h} K_3^i$ is processed in η in parallel with some task from $\cup_{1 \leq i < h} K_2^i$, because all tasks constituting $\cup_{1 \leq i < h} K_1^i$ are processed in parallel with tasks from M_2. Since $|\cup_{1 \leq i < h} K_2^i| = |\cup_{1 \leq i < h} K_3^i|$, no other tasks from K_3 are processed in parallel with tasks from $\cup_{1 \leq i < h} K_2^i$. Hence v is the first task in the K_3^h-chain. Suppose that $h = 1$, then any task from K_3 preceding v is to be processed in parallel with some task from K_1^1, which contradicts the selection of v. Hence v is the first task of the K_3^1-chain.

Among all tasks $j \in M_2$, satisfying the condition $C_j(\eta) > C_u(\eta)$, select the task with the smallest completion time. Let it be task w and suppose that this task corresponds to a_q. Let $\widetilde{M}_1^q$ be the set of all tasks $j \in M_1^q$ such that

$$C_u(\eta) < C_j(\eta) < C_w(\eta)$$

By Lemma 6 and by the selection of w, each time slot in the time interval $[C_u(\eta), C_w(\eta) - 1]$, containing a 1-task, contains one task from $K_1 \cup K_2$ and one task from K_3. Let $\widetilde{K}_1$ be the set of all tasks $j \in K_1$ which are processed in η in the time interval $[C_u(\eta), C_w(\eta)]$, $\widetilde{K}_2$ be the set of all tasks $j \in K_2$ which are processed in η in the time interval $[C_u(\eta), C_w(\eta)]$, and $\widetilde{K}_3$ be the set of all tasks $j \in K_3$ which are processed in η in the time interval $[C_u(\eta), C_w(\eta) - 1]$. Let $\widetilde{N}$ be the set of all tasks j such that

$$C_u(\eta) < C_j(\eta) < C_w(\eta) \quad \text{and} \quad j \notin \widetilde{M}_1^q \cup \widetilde{K}_1 \cup \widetilde{K}_2 \cup \widetilde{K}_3$$

Observe that all tasks in $\widetilde{N}$ are 2-tasks.

Among all $j \in \widetilde{K}_3$ select a task with the largest completion time $C_j(\eta)$. Let it be task y and let $y \in K_3^r$ for some $r \geq h$. Then by (14) and the fact that, for any i, $|K_2^i| = |K_3^i|$,

$$C_w(\eta) < \min_{j \in K_1^{r+1}} C_j(\eta) \tag{18}$$

and for any $h \leq i < r$

$$\max_{j \in K_3^i} C_j(\eta) + 1 < \min_{j \in K_1^{i+1}} C_j(\eta) \tag{19}$$

Consider a new schedule α where

- $C_x(\alpha) = C_x(\eta)$ if either $C_x(\eta) < \min\limits_{j \in K_4^h} C_j(\eta)$, or $C_x(\eta) > C_w(\eta)$;
- tasks comprising $\widetilde{M}_1^q$ are processed in the time interval

$$\left[\min_{j \in K_4^h} C_j(\eta) - 1, \min_{j \in K_4^h} C_j(\eta) - 1 + |\widetilde{M}_1^q|\right]$$

  in the same order as in η;
- $C_u(\alpha) = C_w(\alpha) = \min\limits_{j \in K_4^h} C_j(\eta) + |\widetilde{M}_1^q|$;
- tasks comprising K_4^h are processed in the time interval

$$\left[\min_{j \in K_4^h} C_j(\eta) + |\widetilde{M}_1^q|, \min_{j \in K_4^h} C_j(\eta) + |\widetilde{M}_1^q| + |K_4^h|\right]$$

  in the same order as in η;
- tasks comprising $\widetilde{K}_1 \cup \widetilde{K}_2 \cup \widetilde{N}$ are processed in the time interval

$$\left[\min_{j \in K_4^h} C_j(\eta) + |\widetilde{M}_1^q| + |K_4^h|, C_w(\eta)\right]$$

  in the same order as in η;
- tasks comprising $\widetilde{K}_3$ are processed in parallel with tasks from $\widetilde{K}_1 \cup \widetilde{K}_2$ in the same order as in η.

The feasibility of α follows from (16), the selection of w, (18) and (19). On the other hand, by (12),

$$\begin{aligned} C_\Sigma(\alpha) - C_\Sigma(\eta) &= \frac{1}{2}\{\Delta_{21}(\alpha) - \Delta_{21}(\eta)\} \\ &< \frac{1}{2}\left\{-2 \cdot |K_4^h| + (2|\widetilde{K}_3| + 2) \cdot |\widetilde{M}_1^q|\right\} \\ &< \frac{1}{2}\left\{-4z^2b^3 + (4z^2b^2 + 2)\frac{b}{2}\right\} < 0 \end{aligned}$$

which contradicts the optimality of η. □

Theorem 5 *For any instance of the 3-partition problem, a 3-partition exists if and only if there is a feasible schedule η for the corresponding scheduling problem such that $C_\Sigma(\eta) \leq C$.*

Suppose that a 3-partition $\mathcal{A}_1, \ldots, \mathcal{A}_z$ exists, and that for each i, $\mathcal{A}_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$. Consider a schedule η in which

- for each $1 \leq i \leq z$, tasks from $M_1^{i_1} \cup M_1^{i_2} \cup M_1^{i_3}$ are processed in the time interval $[(2b+2zb^2+2z^2b^3)(i-1), (2b+2zb^2+2z^2b^3)(i-1)+b]$, and tasks from $M_2^{i_1} \cup M_2^{i_2} \cup M_2^{i_3}$ are processed in the time interval $[(2b + 2zb^2 + 2z^2b^3)(i-1) + b, (2b + 2zb^2 + 2z^2b^3)(i-1) + 2b]$ in parallel with tasks from K_1^i;
- tasks from K_5 and from all K_4^i, K_2^i, K_3^i are processed in accord with (t3), (t4) and (t5).

This schedule is feasible, and since η satisfies all the conditions (t1)–(t5), $C_\Sigma(\eta) \leq C$.

Suppose that a 3-partition does not exist. Consider any schedule σ, probably infeasible, satisfying conditions (t1)–(t5), and an optimal schedule η. Denote by $M_i(\sigma)$ the set of all tasks $j \in M_2$ which are processed in σ in parallel with one of the tasks from K_1^i, and denote by $M_i(\eta)$ the set of all tasks $j \in M_2$ which are processed in η in parallel with one of the tasks from K_1^i. It is easy to see that for any $1 \leq i \leq z$ and any $g \in K_1^i \cup M_i(\sigma)$

$$\sum_{j \in M_1 \cup K_4} \delta_{jg}(\sigma) = bi + (i-1)2z^2b^3 \tag{20}$$

and for any $1 \leq i \leq z-1$ and any $g \in K_2^i \cup K_3^i$

$$\sum_{j \in M_1 \cup K_4} \delta_{jg}(\sigma) = bi + 2z^2b^3i \tag{21}$$

Similarly, taking into account (13), (14), (15), (17) and Lemmas 6 and 7, for any $1 \leq i \leq z$ and any $g \in K_1^i \cup M_i(\eta)$

$$\sum_{j \in M_1 \cup K_4} \delta_{jg}(\eta) > (i-1)b + (i-1)2z^2b^3 \tag{22}$$

for any $1 \leq i \leq z-1$ and any $g \in K_2^i \cup K_3^i$

$$\sum_{j \in M_1 \cup K_4} \delta_{jg}(\eta) \geq bi + 2z^2b^3i \tag{23}$$

and for at least one i, say i^*,

$$\sum_{j \in M_1 \cup K_4} \delta_{jg}(\eta) \geq bi^* + 2z^2b^3i^* + 1 \tag{24}$$

Hence by (20)–(24) and (12),

$$C_\Sigma(\eta) - C = C_\Sigma(\eta) - C_\Sigma(\sigma) = \frac{1}{2}\{\Delta_{21}(\eta) - \Delta_{21}(\sigma)\}$$

$$> \frac{1}{2}\{-b|K_1 \cup M_2| + |K_2^{i^*} \cup K_3^{i^*}|\} = \frac{1}{2}\{-2zb^2 + 4zb^2\} > 0$$

Since η is an optimal schedule, there is no feasible schedule with the criterion value less than or equal to C. □

Since the 3-partition problem is NP-complete in the strong sense, Theorem 5 implies that $P2|in\text{-}tree, p_j = 1, size_j|C_\Sigma$ is NP-hard in the strong sense.

4. CONCLUSIONS

The classical Coffman–Graham–Geray result establishing the existence of an ideal schedule for the problem with two identical parallel machines, arbitrary precedence constraints and unit execution time tasks can be extended in two different ways:

- as has been shown above, an ideal schedule exists and can be found in polynomial time if each task has a release time;
- as has been proven in Coffman *et al.* (2003), an ideal schedule exists and can be found in polynomial time if preemptions are allowed.

The question of the existence of an ideal schedule and a polynomial-time algorithm for the model, obtained from the original one by introducing both release times and preemptions, still remains open and can be viewed as a direction of further research.

As has been proven above, the $P2|prec, p_j = 1, size_j|C_\Sigma$ problem is NP-hard in the strong sense even if the precedence constraints are restricted to in-trees. Although this proof together with Zinder *et al.* (submitted), where NP-hardness has been proven for out-trees, strengthens the original result of Brucker *et al.* (2000), these two results do not provide the full characterization of the $P2|prec, p_j = 1, size_j|C_\Sigma$ problem and its complexity status also requires further research.

References

Brucker, P., Knust, S., Roper, D. and Zinder, Y. (2000) Scheduling UET task systems with concurrency on two parallel identical machines. *Mathematical Methods of Operations Research*, **52/3**:369–387.

Coffman Jr, E. G. and Graham, R. L. (1972) Optimal scheduling for two-processor systems. *Acta Informatica*, **1**:200–213.

Coffman, E. G., Sethuraman, J. and Timkovsky, V. G. (2003) Ideal preemptive schedules on two processors. *Acta Informatica*, **39**:597–612.

Garey, M. R. and Johnson, D. S. (1977) Two-processor scheduling with start-time and deadlines. *SIAM Journal of Computing*, **6**:416–426.

Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (1993) Sequencing and scheduling: algorithms and complexity. In *Logistics of Production and Inventory*, S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin (Eds.), Elsevier, Amsterdam.

Zinder, Y. (1986) An efficient algorithm for deterministic scheduling problem with parallel machines. *Cybernetics*, **N2** (in Russian).

Zinder, Y., Do, V. H. and Oguz, C. (2002) Computational complexity of some scheduling problems with multiprocessor tasks. *Discrete Applied Mathematics*. Submitted.

Personnel Scheduling