Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

by

ROLF H. MÖHRING AND MARKUS W. SCHÄFFTER

NO. 506/1995

A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

Rolf H. Möhring* Markus W. Schäffter*

April 1996

Abstract

In the last few years, scheduling jobs due to communication delays has received a great deal of attention. We consider the problem of scheduling forests due to unit processing times and zero-one communication delays and focus on the approximation algorithm of Hanen and Munier and on the algorithm of Guinand, Rapine and Trystram. These algorithms and their analysis are quite complex. In contrast, we present a very simple list scheduling algorithm for the problem $P|prec{=}tree, p_j = 1, c_{ij} \in \{0,1\}|C_{\max}$ of scheduling trees subject to unit processing times and zero-one communication delays.

For sufficiently many machines, e.g. $m \geq |V|$, the resulting schedule is optimal. For a restricted number of machines, the presented algorithm has the same absolute worst case performance as the algorithm of Guinand, Rapine and Trystram: $\frac{m-1}{2}$. It's relative worst case performance ratio turns out to be bounded by $\left(2 - \frac{1}{m}\right)$ even for arbitrary processing times. This simplifies an argument by Hanen and Munier for the case that an optimal schedule on an infinite number of machines can be constructed.

1 Introduction

We consider the problem of scheduling jobs of unit processing times subject to precedence relations that are given by an out-forest and subject to zero-one communication delays. In the extended notion of Graham et al. [GLLRK79] this problem is denoted by $P|prec{=}forest, p_j = 1, c_{ij} \in \{0,1\}|C_{\max}$.

Since communication delays of length zero are allowed, we can transform every out-forest into an out-tree by introducing a "super-root" as a predecessor of each root in the forest. Let the processing time of that additional job be equal to 1 and set the communication delay between the "super-root" and all "original roots" to 0. Then each schedule that is feasible with respect to the resulting out-tree is also feasible with respect to the given out-forest, when the first time slot is removed. The minimum makespans of these two problems differ by exactly 1 time unit.

Hence, let us assume without loss of generality that the given precedence-relations are given by an out-tree. In the literature, the following related problems have been treated.

- In 1987, Rayward-Smith [RS87] proved that the problem of scheduling unit-time jobs with unit communication delays subject to arbitrary precedence relations is NP–hard. In the same paper it is shown that the relative worst case performance of every greedy schedule is at most $(3 - 2/m)$.
- In 1992, Picouleau developed in his thesis that the problem $P\infty|prec{=}tree, c_{ij} = c|C_{\max}$ is NP–hard when c is part of the input. He presented a simple linear time, $(h-1)c$ approximation algorithm for this problem [Pic92]. If the communication delays times are smaller than the processing times, the corresponding problem can be solved in polynomial time [Chr89].

*Technische Universität Berlin, Fachbereich Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136, 10623 Berlin, Germany, e-mail: {moehring,schaeffter}@math.tu-berlin.de.

- In 1993, Lenstra, Veldhorst and Veltman [LVV93] showed that the problem $P|prec, c_{ij} = 1|C_{\max}$ becomes NP–hard already for in-trees, thus in particular for the larger class of series-parallel orders. The given transformation can even be extended to *binary in-trees*. For the two machine case, the authors constructed a linear time algorithm that solves the problem optimally.

- In the same year, Lawler [Law93] developed a linear time approximation algorithm based on critical path scheduling for the problem $P|prec{=}tree, p_j = 1, c_{ij} = 1|C_{max}$ which constructs a schedule that misses the minimum makespan by at most $(m-2)$ time-units for any number m of machines.

- In 1994/95, Hanen and Munier [HM95] gave an approximation algorithm for the problem $P|prec, c_{ij}\text{ small}|C_{\max}$ with a worst case ratio of $\frac{4+3\rho}{2+\rho} - \frac{2+2\rho}{(2+\rho)m}$, where $\rho \leq 1$ denotes the ratio between the maximum communication delay and the minimum processing time. For the case of unit processing times and unit time communication delays, this approach yields a bound of $\frac{7}{3} - \frac{4}{3m}$ (see also [MH94]). This bound is obtained by solving an LP-relaxation and rounding the obtained fractional values. For the corresponding problem with an unlimited number of machines, the relative performance is $\frac{4}{3}$.

- In October 1995, Munier and Hanen [MH95] presented a list scheduling algorithm for the case that *task duplication* is allowed, whose worst case relative performance ratio is at most $\left(2 - \frac{1}{m}\right)$ for the problem $P|dup, prec, p_j = 1, c_{ij} = 1|C_{\max}$. Since for in-forests, duplication is of no use, this bound also states for the problem $P|prec{=}tree, p_j = 1, c_{ij} = 1|C_{max}$.

- In July 1995, Guinand, Rapine and Trystram [GRT95] presented an approximation algorithm that improves the bound for the absolute error stated by Lawler [Law93] for the problem $P|prec{=}tree, p_j = 1, c_{ij} = 1|C_{\max}$ by a factor of 2 to $\frac{m-1}{2}$. This is the same bound, we state in this paper.

- In December 1995, the authors [MS95] stated a 2-approximation algorithm for the case of series-parallel orders, $P|prec{=}series\text{-}parallel, c_{ij} = 1|C_{\max}$. The given bound can be strengthened to $\left(2 - \frac{1}{m}\right)$ by the same arguments used in this paper.

We give a simple linear-time approximation algorithm with an absolute worst case performance of $\frac{m-1}{2}$ for the problem $P|prec{=}tree, p_j = 1, c_{ij} \in \{0,1\}|C_{\max}$. It's relative worst case performance ratio can be bounded by $\left(2 - \frac{1}{m}\right)$ even for arbitrary processing times.

Even when we focus on this problem, the definitions and lemmas are given as general as possible. The paper is organized as follows. First we introduce the necessary notations in Section 2. Section 3 then presents the approximation algorithm and proves its correctness. In Section 4, we apply the extremal case of [GRT95] to our approximation algorithm, showing that the given bound of $\frac{m-1}{2}$ is tight.

2 Definitions and Notations

A comprehensive overview on the theory of scheduling can be found in [GLLRK79]. We only give a brief overview on the definitions necessary here. In order to be compatible with the commonly used notation, we give the definitions for arbitrary processing times and communication delays.

An instance (m, V, p, Θ, c) of the problem $P|prec{=}tree, c_{ij}|C_{\max}$ consists of the number m of available machines, a set V of jobs, a processing time $p(v)$ for each job $v \in V$, a precedence order Θ which is an out-tree, and an interprocessor communication delay $c(v,w)$ for each pair of jobs, v and w, where w is a direct successor of v. In this paper, we focus on the case of unit processing times and zero-one communication delays, i.e. $p(v) = 1$ and $c(u,v) \in \{0,1\}$ for all jobs $u, v \in V$.

We denote the set of *direct successors* of a job v by $Children(v)$ and the set of all successors (direct or indirect) of v, not including v, by $Succ(v)$. The single direct predecessor of a job v is denoted by $father(v)$, while the set of all predecessors (direct or indirect) of v is denoted by $Pred(v)$. A *schedule S on a job set V* is a function assigning a *starting time* $S(v)$ to each job $v \in V$. The *completion time* of a job v is defined by $C(v) = S(v) + p(v)$. Then the *length (or makespan) of a schedule S* can be computed as $C_{\max}(S) = \max\{C(v) \mid v \in V\}$. For a schedule S, we call the intervals $s_t = [t, t+1]$ (for $t = 0, \ldots, C_{\max}(S) - 1$) the *time slots of S*.

We call a schedule S *weakly feasible* if it respects the limited number of machines and the precedence relations:

(I) No more than m jobs are scheduled simultaneously in any time slot s_t, i.e.
$\sharp\{v \in V | S(v) \leq t < C(v)\} \leq m$ for all $t = 0, \ldots, C_{\max}(S) - 1$.

(II) No job is started before all its predecessors have been finished. Here, $S(v) \geq C(father(v))$ for all jobs $v \in V$ is sufficient.

A weakly feasible schedule does not obey the communication delays. When communication delays are introduced, we have to modify the second condition leading to the following definition of feasibility. A schedule is called *feasible* if all jobs $u \in V$ fulfill the following conditions.

(I) No more than m jobs are scheduled simultaneously in any time slot s_t, i.e.
$\sharp\{v \in V | S(v) \leq t < C(v)\} \leq m$ for all $t = 0, \ldots, C_{\max}(S) - 1$.

(IIa) $S(v) \geq C(u)$ for all $v \in Children(u)$.

(IIb) $S(v) < C(u) + c(u,v)$ for at most one job $v \in Children(u)$.

Condition (IIb) guarantees that, for each job u, at most direct successor v such that $c(u,v) > 0$ has to be scheduled on the same machine as u in order to prevent the corresponding communication delay. We call such a job $v \in Children(u)$, satisfying Condition (IIb) (if there is any) the *favored successor of u in S*.

For a given instance $I = (m, V, p, \Theta, c)$ of the problem $P|prec{=}forest, c_{ij}|C_{\max}$ let $C_{opt}(I)$ denote the minimum makespan among all feasible schedules. Similarly, $C^{\infty}_{opt}(I)$ denotes the minimum makespan among all schedules that are feasible for the instance if the number of machines is not restricted. Clearly, $C^{\infty}_{opt}(I) \leq C_{opt}(I)$.

For a feasible schedule S one can construct a *feasible machine assignment*, this is assigning every job u to a machine $M(u) \in \{M_1, M_2, \ldots, M_m\}$ such that the following conditions are fulfilled.

(I') At any time $t = 0, \ldots, C_{\max}(S) - 1$, no more than one job is assigned to each machine M, i.e.
$\sharp\{v \in V | S(v) \leq t < C(v) \text{ and } M(u) = M_i\} \leq 1$ for each $i \in \{1, 2, \ldots, m\}$.

(II'a) $S(v) \geq C(u)$ for all $v \in Children(u)$ with $M(u) = M(v)$.

(II'b) $S(v) \geq C(u) + c(u,v)$ for all $v \in Children(u)$ with $M(u) \neq M(v)$.

For a machine assignment consider a machine M, idle during a time slot $[t, t+1]$. We call the corresponding time unit from t to $t+1$ an *idle time on machine M*. Hence, at each time t when a machine M executes a job or there is an idle time on machine M at time t. Note that the number of idle times occurring in a time slot $[t, t+1]$ does not depend on the machine assignment. Hence, every feasible schedule S for m machines satisfies the following equation.

$$C_{\max}(S) = \frac{\sum\{p(v) \mid v \in V\} + \sharp\{\textit{idle times in } S\}}{m} \tag{1}$$

The term $\sum\{p(v) \mid v \in V\}/m$ is called the *load* of the corresponding problem instance.

A simple way to find a feasible machine assignment for a feasible schedule S is as follows. First assign all jobs with a starting time of zero, i.e. the jobs in time slot s_0, arbitrarily to the machines. Condition (I) guarantees that every jobs in time slot s_0 can be assigned to a machine.

Then iterate over the time slots s_i, $i = 1, 2, \ldots, C_{\max} - 1$. For each time slot s_i consider first of these jobs v that have at least one direct predecessor u such that $S(v) < C(u) + c(u, v)$. Condition (IIb) states that there is at most one such direct predecessor u of each job v. Hence, we can set $M(v) = M(u)$ for these jobs v. The remaining jobs v' in time slot s_i then fulfill $S(v') \geq C(u) + c(u, v')$ for each direct predecessor u of v'. Hence, these jobs v' can be scheduled on every one of the still empty machines. Then make an arbitrary such assignment. Again, Condition (I) guarantees that there are enough empty machines to assign every job v' in time slot s_i to one machine.

Note that this approach takes only linear time for arbitrary partial orders.

3 The Approximation Algorithm

We first determine for a given tree a priority function on the job set V. This function can be understood as a modified height in the tree that takes the communication delays into account. It is the same function as given in [VRKL94] The idea is to choose for each job a successor to be scheduled on the same machine, the so-called "favored successor". Hence, communication delays may only occur for the remaining, "non-favored" successors. The height function determines the choice of a favored successor for every job and computes the delay produced by that choice.

In a second step, we apply priority list scheduling to the list of jobs ordered by non-increasing height values. To respect the communication delays, we modify the priority list scheduling algorithm, restricting the "availability" of a job at a certain time.

3.1 Defining the Priorities

For a job $v \in V$, define the corresponding *height-value* $h(v)$ as follows. The definition works for arbitrary processing times and communication delays.

- Every leaf v obtains a height-value of $h(v) = p(v)$.
- The height-value of the remaining jobs is defined by $h(v) = p(v) + \max\{h(w_1), h(w_2) + c(v, w_2)\}$ where w_1 and w_2 are jobs with the two largest values of $h(w) + c(v, w)$ among all successors w of v. Without loss of generality, assume that $h(w_1) + c(v, w_1) \geq h(w_2) + c(v, w_2)$.

Obviously, the job r corresponding to the root of the tree fulfills $h(r) = \max\{h(v) \mid v \in V\}$.

Lemma 1 below states that the height values induce a feasible schedule on sufficiently many machines. Lemma 2 yields that this schedule is optimal (for sufficiently many machines) and that $C^{\infty}_{opt}(I) = h(r)$. The proof of Lemma 2 is done by the fact that in any feasible schedule the distance between the starting time of a job v and the starting time of the latest scheduled successor of v is bounded by the differences of their height values. This is stated by Proposition 1.

Lemma 1. *Consider an instance $I = (m, V, p, \Theta, c)$ of the problem $P|prec{=}tree, c_{ij} \in \{0,1\}|C_{\max}$. Let r denote the job corresponding to the root of the tree. Setting $S^{\infty}(v) = h(r) - h(v)$ for all jobs $v \in V$ determines a feasible schedule S^{∞} for the instance I on sufficiently many machines, and $C_{\max}(S^{\infty}) = h(r)$.*

Proof. Let us determine the length of S^{∞} first. Consider a job v that is completed last, $C_{\max}(S^{\infty}) = S^{\infty}(v) + p(v)$. Due to the definition of the schedule, $S^{\infty}(v) = h(r) - h(v)$. Since v is a job without successors, $h(v) = p(v)$. Both together yields $C_{\max}(S^{\infty}) = S^{\infty}(v) + h(v) = h(r)$.

To verify the feasibility, we show that S^{∞} fulfills the conditions (IIa) and (IIb) stated in Section 2. This is sufficient since we do not have to respect the machine restrictions here. Consider a job u. Due to the definition of the height function, every successor v of u satisfies $h(u) \geq p(u) + h(v)$. Thus $S^{\infty}(v) \geq S^{\infty}(u) + p(u)$, which proves

Condition (IIa). Moreover the definition of the height function guarantees that $h(u) \geq p(u) + h(v) + c(u,v)$ for each non–favored successor v of u. This proves Condition (IIb). $\square$

Proposition 1. *Consider a feasible schedule S for an instance $I = (m,V,\Theta,c)$ of the problem $P|prec{=}tree, c_{ij} \in \{0,1\}|C_{\max}$. Then for any job $v \in V$ there exists a job $w \in Succ(v) \cup \{v\}$ such that $h(w) = p(w)$ and $S(w) - S(v) \geq h(v) - h(w)$.*

Proof. Consider a feasible schedule S for an instance $I = (m,V,\Theta,c)$ of the problem $P|prec{=}tree, c_{ij} \in \{0,1\}|C_{\max}$ and a job $v \in V$. The proof is done by induction on the length of a longest chain of successors of job v. The inductive basis is done for jobs v who are leaves in the tree, i.e. $h(v) = p(v)$. For them, Proposition 1 follows directly with $w = v$.

For a job v with $h(v) > p(v)$ assume as the inductive hypothesis that $S(w) - S(v) \geq h(v) - h(w)$ has been proven for all successors of v. Due to the definition of the height function, $h(v)$ is determined by one of the following cases. Thereby, w_1 and w_2 are as in the definition of the height function.

Case 1: $h(v) = p(v) + h(w_1)$

Since w_1 is a successor of v, $S(w_1) \geq S(v) + p(v)$. This gives $S(w_1) - S(v) \geq h(v) - h(w_1)$.

If w_1 is a leaf, the choice $w = w_1$ proves the proposition. Otherwise, by the inductive hypothesis there exists a successor w of w_1 that satisfies $h(w) = p(w)$ and $S(w) - S(w_1) \geq h(w_1) - h(w)$. Together with $S(w_1) - S(v) \geq h(v) - h(w_1)$, this leads to $S(w) - S(v) \geq h(v) - h(w)$ which proves the proposition.

Case 2: $h(v) = p(v) + h(w_2) + c(v,w_2)$

If the height of v is determined by w_2, assume that $h(w_2) + c(v,w_2) > h(w_1)$. Otherwise apply Case 1. Then $h(w_2) = h(w_1)$ and $c(v,w_1) = c(v,w_2) = 1$ due to the definition of w_1, w_2 and the fact that all communication delays are 0 or 1.

Due to the communication delays, one of the jobs w_1, w_2 has to be scheduled after $C(v)+1$ in every feasible schedule. Since the height values and the communication delays are identical, assume without loss of generality that this is w_2.

Thus $S(w_2) \geq C(v) + 1 = S(v) + p(v) + c(v,w_2)$. This, together with the assumption of this case leads to $S(w_2) - S(v) \geq h(v) - h(w_2)$.

If w_2 is a leaf, the proof is obvious for $w = w_2$ since $h(w_2) = p(w_2)$. Otherwise there exists by the inductive hypothesis a successor w of w_2 that satisfies $h(w) = p(w)$ and $S(w) - S(w_2) \geq h(w_2) - h(w)$. Combining this with $S(w_2) - S(v) \geq h(v) - h(w_2)$ completes the proof of Proposition 1. $\square$

Lemma 2. *Given an instance $I = (m,V,\Theta,c)$ of the problem $P|prec{=}tree, c_{ij} \in \{0,1\}|C_{\max}$, then the height of job r corresponding to the root of the tree satisfies $h(r) \leq C^{\infty}_{opt}(I)$.*

Proof. Consider a schedule S_{opt} that is optimal on sufficiently many machines, i.e. $C_{\max}(S^{opt}) = C^{\infty}_{opt}(I)$. Clearly, $S_{opt}(r) = 0$ where r denotes the root of the tree. For $v = r$, Proposition 1 states that there exists a job w with $h(w) = p(w)$ such that $S_{opt}(w) \geq h(r) - h(w)$. Thus,

$$C^{\infty}_{opt}(I) = C_{\max}(S_{opt}) \geq S_{opt}(w) + p(w) = S_{opt}(w) + h(w) \geq h(r)\,.$$

$\square$

In the following we apply *priority list scheduling* (see e.g. [LLRKS93]) to the list of jobs ordered by non increasing height values. The priority list scheduling algorithm schedules jobs at certain decision times. These decision times are the time $t = 0$, the completion times of jobs, and, in order to incorporate the *unit-time communication delays*, the completion times of jobs plus one. These decision times depend on the previously taken decisions and are processed in ascending order. At every decision time t, the algorithm repeatedly chooses among the still unscheduled jobs a job with highest priority (the first "available" job in the given list) that can be started at time t. This is repeated until no more job can be started at t. Then the next decision time is considered.

Compared with the list scheduling rules for the corresponding problem without communication delays, we only need to incorporate Condition (IIb). This means that at most one direct successor of a job u can be scheduled directly after the completion of u. A detailed definition of the *availability of a job* is given in the next section. Note that the following algorithm does not determine a machine assignment, but that the resulting schedule fulfills the Conditions (I), (IIa), and (IIb) for feasibility stated in Section 2). Hence, a machine assignment can be obtained according to the procedure described there.

The next theorem states that priority list scheduling, applied to the list of jobs ordered by non-increasing height values, yields a feasible schedule S that is at most twice as long as the minimum. We will show that the number of idle time slots in S is at most the maximum height value $h(r)$, which, together with Lemma 2 proves the theorem. Theorem 1 can be generalized to series-parallel orders and unit communication delays, i.e. for the problem $P|prec{=}series\text{-}parallel,\, c_{ij}=1|C_{\max}$. In [MS95], the authors state a bound of 2 for this problem. The corresponding proof can be modified similar to the proof of Theorem 1, obtaining a bound of $\left(2-\frac{1}{m}\right)$.

Theorem 1. *Let $I=(m,V,p,\Theta,c)$ denote an instance of the problem $P|prec{=}forest,\, c_{ij}\in\{0,1\}|C_{\max}$ and let S be the schedule that is determined by the modified priority list scheduling approach, applied to any list of jobs ordered by non-increasing height values. Then S fulfills* $C_{\max}(S)\le\left(2-\frac{1}{m}\right)C_{opt}(I)$.

Proof. As described in the introduction, we can assume without loss of generality that the precedence relations are given by a tree. The claimed inequality is shown by considering in the schedule S for every time t the maximum *remaining height* $\hat{h}(v)$ among all jobs v that are busy at t or start after t. For a job v that is scheduled after time t, its *remaining height* $\hat{h}(v)$ equals its height $h(v)$, while for a job w that is being processed during time slot s_{t-1}, the *remaining height at time t* is the difference between its height and the number of time-units that w has already been processed, i.e. $\hat{h}(w)=h(w)-(t-S(w))=h(w)+S(w)-t$. The same idea was also used by Jaffe [Jaf80] for scheduling problems without communication delays. This leads to the following definition of $H(t)$ for each $t=0,1,\dots,C_{\max}(S)-1$.

$$\begin{aligned} H(t) &= \max\{\hat{h}(v)\mid C(v)>t\} \\ &= \max\Big(\{h(v)+S(v)-t\mid S(v)\le t<C(v)\}\cup\{h(v)\mid S(v)>t)\}\Big) \end{aligned}$$

Clearly, the function $H(t)$ is monotonically non-increasing in t. We will show that $H(t)$ decreases along time slots with idle times. More precisely,

$$H(t+1)\le H(t)-1 \quad \textbf{or} \quad H(t+1)\le H(t-1)-2 \tag{2}$$

for each time slot s_t that contains idle times. With inequality (2) we obtain that the number of time slots that contain idle times is at most the maximum height value $h(r)$. Since the partial order has tree structure the favored successor of a job v can be scheduled directly after v ends. Hence, at any time, at least one job is available to be processed. Thus, the number of idle times in the resulting schedule is at most $(m-1)h(r)$, which, by Lemma 2, is at most $(m-1)C^{\infty}_{opt}(I)\le(m-1)C_{opt}(I)$. On the other hand, $C_{opt}(I)$ is at least the load $\sum\{p(v)\mid v\in V\}/m$ of the problem instance and hence,

$$\begin{aligned} C_{\max}(S) &\overset{(1)}{=} \frac{\sum\{p(v)\mid v\in V\}}{m}+\frac{\sharp\{\textit{idle times}\}}{m} \\ &\le \frac{\sum\{p(v)\mid v\in V\}}{m}+\left(1-\frac{1}{m}\right)C_{opt}(I) \\ &\le \left(2-\frac{1}{m}\right)C_{opt}(I)\,. \end{aligned}$$

To prove inequality (2) consider a time slot s_t in which less than m machines are busy. If s_t is the last time slot, i.e. $t = C_{\max}(S) - 1$, then $H(t) = 1$ and $H(t+1) = 0 = H(t) - 1$ and inequality (2) is satisfied. Hence, assume without loss of generalization that $t < C_{\max}(S) - 1$. Due to the zero-one communication delays and to the tree structure, every time slot contains at least one job. Since s_t contains idle times, no job v with $S(v) \geq t+1$ could be scheduled at time t. This means that every such job v is a successor or a sibling of a job processed in time slot s_t. Let $\overline{v}$ denote a job that determines the maximum in the definition of $H(t+1)$ and consider the following case analysis.

The first case treats the situation when job $\overline{v}$ could not be scheduled since its direct predecessor $u = \mathit{father}(\overline{v})$ was processed during time slot s_t, i.e. $S(u) \leq t < C(u)$. In the second case, job $\overline{v}$ has been available at time t but one of its sibling jobs v' was processed instead. Since all communication delays are at most one, we have $C(u) = t = S(v')$ and $S(\overline{v}) = t+1$, where u denotes the common direct predecessor of $\overline{v}$ and v'.

Case 1: $S(u) \leq t < C(u)$

Since u is processed in time slot s_t, it is involved in determining $H(t)$, i.e. $H(t) \geq h(u) + S(u) - t$. This, together with the assumption $C(u) - t \geq 1$, leads to

$$H(t) \geq h(u) + S(u) - t \geq h(\overline{v}) + C(u) - t \geq h(\overline{v}) + 1 = H(t+1) + 1\,.$$

Case 2: $C(u) \leq t$

As stated above, in this case $C(u) = t = S(v')$ and $S(\overline{v}) = t+1$ and $c(u,\overline{v}) = c(u,v') = 1$. Since u is processed in time slot s_{t-1}, it is involved in determining $H(t-1)$, i.e. $H(t-1) \geq h(u) + S(u) - (t-1)$. Note that the priority list scheduling algorithm schedules v' instead of $\overline{v}$ only if $h(v') \geq h(\overline{v})$. Now, the definition of the height function leads to $h(u) \geq p(u) + h(\overline{v}) + 1$ since $c(u,\overline{v}) = c(u,v') = 1$.

This yields the following inequality which completes the proof of Theorem 1.

$$\begin{aligned} H(t-1) \geq h(u) + S(u) - t + 1 &= h(u) + C(u) - p(u) - t + 1 \\ &= h(u) - p(u) + 1 \\ &\geq h(\overline{v}) + 2 \\ &= H(t+1) + 2\,. \end{aligned}$$

□

3.2 Formulation of the Algorithm and Correctness Proof

From now on, we consider unit processing times, i.e. $p(v) = 1$ for all $v \in V$. The idea behind the algorithm is that all jobs fitting into a time slot compete with each other in order to be scheduled. In principle, the algorithm performs priority list scheduling with an implicit machine assignment. This means the algorithm determines a starting time for each job satisfying the conditions (I) and (II) for feasibility. Hence, a machine assignment can easily be constructed as stated in Section 2.

Compared with the list scheduling rules for the corresponding problem without communication delays, we only need to incorporate the fact that at most one direct successor of a job v can be scheduled directly after the completion of v. The algorithm works as follows: For the current time t, the set *avail* contains all jobs that are unscheduled and can be scheduled at time t without violating the precedence constraints and the communication delays. We call these jobs available to be scheduled at time t. In selecting the jobs to be scheduled at time t, we also determine the set *avail'* of jobs that will become *available* at the next decision time $t+1$.

Definition 1. *A job v is called* available at time *t if*

(i) $\mathrm{C^H}(u) \leq t$ *for all predecessors u of v,*

(ii) $t \geq \mathrm{C^H}(u) + c(u,v)$ *if v in not the favored successor of u.*

At time t, $t = 0, 1, 2, \ldots$, take the first job v in the list that has not been scheduled before and that is available at time t. Set its starting time to $\mathrm{S}^{\mathrm{H}}(v) = t$. If $\mathrm{S}^{\mathrm{H}}(v) = \mathrm{C}^{\mathrm{H}}(u)$ for $u = \mathit{father}(v)$ and if $c(u,v) = 1$, v is the favored successor of u in S^{H} and hence, no further successor v' of u with $c(u,v') = 1$ can be scheduled at time t. To capture this in the algorithm, we remove all such jobs v' from the set *avail* of currently available jobs. On the other hand, all these jobs become available on every machine at time $t+1$, and hence we store them in the set avail'.

Repeat this step until no further job is available to be scheduled at time t or the number of jobs scheduled at time t equals the number m of machines. Proceed with the next time slot until all jobs are scheduled. A detailed description of the algorithm is given in Figure 1.

Algorithm DETERMINE-SCHEDULE

Input: An instance $I = (m, V, \Theta, c)$ of the problem $P|\mathit{prec{=}forest}, p_j = 1, c_{ij} \in \{0,1\}|C_{\max}$.

Output: A feasible schedule S^{H} such that $C_{\max}(\mathrm{S}^{\mathrm{H}}) \leq C_{opt}(I) + \frac{1}{2}(m-1)$.

Preprocessing

1. Compute the height-value $h(v)$ for every job $v \in V$.
2. Set $h_{\max} = \max\{h(v) \mid v \in V\}$.
3. Set $\mathit{avail} = \{w \mid w \text{ is root in } \Theta\}$ and $\mathrm{S}^{\mathrm{H}}(v) = \infty$ for each job $v \in V$.

Main Loop

Begin with $t = 0$ and repeat while there exist jobs v with $\mathrm{S}^{\mathrm{H}}(v) = \infty$.

1. Set $\mathit{machines} = 0$ and $\mathit{avail}' = \emptyset$.
2. While $(\mathit{avail} \neq \emptyset$ and $\mathit{machines} < m)$ do
   (a) Select a job v of largest height-value in *avail*.
   (b) Set $\mathrm{S}^{\mathrm{H}}(v) = t$ and $\mathrm{C}^{\mathrm{H}}(v) = t+1$. Increase *machines* by 1, and remove v from *avail*.
   (c) Set $u = \mathit{father}(v)$. If $\mathrm{C}^{\mathrm{H}}(u) = t$ and $c(u,v) = 1$ then move all unscheduled successors $v' \in \mathit{Children}(u)$ with $c(u,v') = 1$ from *avail* to avail'.
   (d) Put all successors of v into avail'.
3. Set $t = t+1$ and $\mathit{avail} = \mathit{avail} \cup \mathit{avail}'$.

Figure 1: Detailed description of the algorithm DETERMINE-SCHEDULE

Lemma 3. *Algorithm* DETERMINE-SCHEDULE *produces a feasible schedule* S^{H} *for every instance* $I = (m, V, \Theta, c)$ *of the problem* $P|\mathit{prec{=}forest}, p_j = 1, c_{ij} \in \{0,1\}|C_{\max}$.

Proof. We only need to verify that the resulting schedule S^{H} satisfies Conditions (I) and (II) for feasibility.

Since the loop in Step (2) in Figure 1 stops when at most m jobs are scheduled in the current time slot s_t, Condition (I) holds. Condition (IIa) is fulfilled since jobs are only included into the set *avail* only after all their predecessors have been completed. Step (2c) guarantees that for a job v, no more that one successor w with $c(v,w) > 0$ is scheduled at time $\mathrm{C}^{\mathrm{H}}(v)$. Hence, Condition (IIb) is fulfilled. □

In the following we consider a subset $V' \subseteq V$ of jobs that are "critical" in the following sense. For a time slot $[t, t+1]$, we call m minus the number of jobs in V' processed during $[t, t+1]$ the number of *essential idle times* in

$[t,t+1]$. An optimal schedule is then distinguished by the fact that it minimizes the number of essential idle times. In Proposition 3 we show that in the schedule S^{H} that is determined by the algorithm DETERMINE-SCHEDULE, the jobs of V' are scheduled as early as possible subject to the considered choice of favored successors. This allows us to bound the number of essential idle times in S^{H} in the proof of Theorem2. We then compare the number of essential idle times in S^{H} and in any optimal schedule S_{opt} which yields that the makespan of S^{H} and of any optimal schedule S_{opt} differ by at most $\frac{1}{2}(m-1)$ time units.

Proposition 2. *For an instance $I=(m,V,\Theta,c)$ of the problem $P|prec{=}forest, p_j=1, c_{ij}\in\{0,1\}|C_{\max}$, let S denote an arbitrary feasible schedule. Then, for every job v, there exists a job $x\in Succ(v)\cup\{v\}$ such that $S(x)-S(v)\geq h(v)-1$.*

Proof. The proof is done by induction on the height value of the jobs. Obviously, Proposition 2 is satisfied by any leaf of the tree, i.e. for jobs v such that $h(v)=1$. Hence assume as inductive hypothesis that the proposition holds for every successor of a job v. Due to the definition of the height function, $h(v)$ is either determined by a direct successor w with $h(w)=h(v)-1$ or by two direct successors w_1,w_2 such that $c(v,w_1)=c(v,w_2)=1$ and $h(w_1)=h(w_2)=h(v)-2$.

In the first case, the inductive hypothesis states the existence of a job $x\in Succ(w)\subseteq Succ(v)$ such that $S(x)-S(w)\geq h(w)-1$ and hence, $S(x)-S(v)\geq S(x)-S(w)+1\geq h(w)=h(v)-1$.

In the second case, at least one of the jobs, say w_2 satisfies $S(w_2)\geq S(v)+p(v)+c(v,w_2)$. For w_2, the inductive hypothesis states the existence of a job $x\in Succ(w_2)\subseteq Succ(v)$ such that $S(x)-S(w_2)\geq h(w_2)-1$. Hence,

$$S(x)-S(v)\geq S(x)-S(w_2)+2\geq h(w)+1=h(v)-1\,.$$

This completes the proof of Proposition 2. □

Proposition 2 states that the height value of a job v is a lower bound of the length of the chain from v to a latest scheduled successor x of v. We use this insight to define a subtree Θ' of Θ induced by a set V' of jobs that have very long chains of predecessors. As we will see, this subtree is "critical" in the sense that the number of idle times in the sub–schedule S' of S^{H} induced by V' is not too far away from the number of idle times in an optimal schedule for the job set V'. Proposition 3 shows that S' has a special structure (see also Figure 2). Based on this structure, Proposition 4 bounds the number of idle times in S'.

If algorithm DETERMINE-SCHEDULE constructs a schedule that never schedules m jobs in parallel, Lemma 1 and Lemma 2 yield the optimality of this schedule. Hence assume for the following analysis that there exists a time when m jobs are processed.

Proposition 3. *For an instance $I=(m,V,\Theta,c)$ of the problem $P|prec{=}forest, p_j=1, c_{ij}\in\{0,1\}|C_{\max}$, let S^{H} denote the schedule determined by algorithm* DETERMINE-SCHEDULE. *Assume that there exists a time $t'\leq C_{\max}(\mathrm{S}^{\mathrm{H}})-2$, such that the number of jobs started at time t' is strictly less than m. Set*

$$V'=\{v\in V\mid \exists u\in Pred(v)\cup Succ(v)\cup\{v\}:(\mathrm{S}^{\mathrm{H}}(u)=t'-1,h(u)\geq 3)\ \textbf{\textit{or}}\ (\mathrm{S}^{\mathrm{H}}(u)=t',h(u)\geq 2)\}\,. \quad (3)$$

Then each job $v\in V'$ such that $\mathrm{S}^{\mathrm{H}}(v)\leq t'$ satisfies

(A) *$h(v)\geq t'-\mathrm{S}^{\mathrm{H}}(v)+2$,*

(B1) *every direct successor $w\in V'$ of v such that $\mathrm{S}^{\mathrm{H}}(w)\leq t'+1$ satisfies $\mathrm{S}^{\mathrm{H}}(w)\leq \mathrm{C}^{\mathrm{H}}(v)+c(v,w)$,*

(B2) *at least one direct successor $w\in V'$ of v satisfies $\mathrm{S}^{\mathrm{H}}(w)=\mathrm{C}^{\mathrm{H}}(v)$.*

Proof. The proof is done by induction on the completion times of the jobs. For the inductive basis consider jobs $v \in V'$ with $\mathrm{C^H}(v) = t'$. Due to the definition of V', each jobs $v \in V'$ satisfies $h(v) \geq 3$ since either $u = v$ satisfies the definition of V' or v has a successor of height at least 2 which implies that $h(v) \geq 3$. Hence, $h(v) \geq 3 = t' - \mathrm{S^H}(v) + 2$ since $t' - \mathrm{S^H}(v) = 1$. This proves inequality (A).

Statements (B1) and (B2) follow from the fact that time slot $[t', t'+1]$ contains idle times and hence, every direct successor w of v with $c(v,w) = 0$ and at least one direct successor w' of v with $c(v,w') = 1$ (if existent) are scheduled by the algorithm at time $t' = \mathrm{C^H}(v)$. Note that the remaining successors w'' of v such that $c(v,w'') = 1$ are not necessarily scheduled at time $t'+1$ though they are available at that time but inequality (B1) only considers successors scheduled before time t'.

For the inductive step consider a time $t < t'$ and a job $v \in V'$ with $\mathrm{C^H}(v) = t$. Assume as inductive hypothesis that every job $v' \in V'$ with $\mathrm{C^H}(v') > t$ and $\mathrm{S^H}(v') \leq t'$ satisfies inequality (A) and statements (B1) and (B2). Due to the definition of V', at least one direct successor w of v is contained in V' and $\mathrm{S^H}(w) \leq t'$. Obviously, $\mathrm{S^H}(w) \geq \mathrm{C^H}(v)$. If $\mathrm{S^H}(w) = \mathrm{C^H}(v)$ then v and w satisfy statements (B1) and (B2) directly and inequality (A) follows since

$$h(v) \geq h(w) + 1 \overset{hyp.}{\geq} t' - \mathrm{S^H}(w) + 3 = t' - \mathrm{S^H}(v) + 2\,.$$

Hence assume without loss of generality that $\mathrm{S^H}(w) > \mathrm{C^H}(v)$ for each direct successor w of v. Note that at least one direct successor w was available to be scheduled at time $\mathrm{S^H}(w) - 1$ but was not selected by the algorithm. Thus, there exist m jobs x that were scheduled instead of w and since the algorithm always selects jobs of highest priority, $h(x) \geq h(w)$.

We now show that each of the m jobs x has a successor z scheduled at time t'. Due to the tree structure, this implies that m different jobs are scheduled at time t' which contradicts the assumption that $[t', t'+1]$ contains idle times.

Since $w \in V'$, $\mathrm{S^H}(w) > \mathrm{C^H}(v) = t$ and $\mathrm{S^H}(w) \leq t'$, w satisfies inequality (A) by the inductive hypothesis and hence, $h(w) \geq t' - \mathrm{S^H}(w) + 2$. Since $h(x) \geq h(w)$ and $\mathrm{S^H}(x) = \mathrm{S^H}(w) - 1$, this yields $h(x) \geq t' - \mathrm{S^H}(x) + 1$. Now, Proposition 2 states the existence of a job $z \in \mathit{Succ}(x)$ such that $\mathrm{S^H}(z) \geq \mathrm{S^H}(x) + h(x) - 1 \geq t'$. Hence, each job x has a successor z scheduled after time t'. Since at every time, at least one job of the chain between x and z is available to be scheduled and since $[t', t'+1]$ contains idle times, each of the m jobs x has a successor that is scheduled at time t'. Due to the tree structure of Θ, these jobs are pairwise distinct which contradicts the assumption that $[t', t'+1]$ contains idle times. Hence, at least one direct successor w of v satisfies statement (B2).

The proof of statement (B1) is analogous. Consider a direct successor w' of v with a minimal starting time $\mathrm{S^H}(w)$ among all direct successors w of v. Assuming that $\mathrm{S^H}(w') > \mathrm{C^H}(v) + c(v,w')$ leads to the existence of m jobs x with $h(x) \geq h(w')$ and $\mathrm{S^H}(x) = \mathrm{S^H}(w') - 1$. The same arguments as used above then gives a contradiction to the assumption that $[t', t'+1]$ contains idle times. Hence, statement (B1) holds.

To complete the proof, we now show that statements (B1) and (B2) induce inequality (A). Following the definition of the height function, $h(v)$ is determined by one or two direct successors of largest height value. Either $h(v) = h(w) + 1$ or $h(v) = h(w_1) + 2 = h(w_2) + 2$ with $c(v,w_1) = c(v,w_2) = 1$. In the first case, $h(w)$ is maximum among all successors of v and, since the algorithm always selects available jobs of highest priority, statement (B2) yields that $\mathrm{S^H}(w) = \mathrm{C^H}(v) = \mathrm{S^H}(v) + 1$. By the inductive hypothesis, w satisfies inequality (A) and hence, $h(w) \geq t' - \mathrm{S^H}(w) + 2$. Both, together with $h(v) = h(w) + 1$ yields $h(v) \geq t' - \mathrm{S^H}(v) + 2$.

In the second case, statement (B1) guarantees that one of the jobs, say w_2, is scheduled at time $\mathrm{S^H}(w_2) = \mathrm{C^H}(v) + c(v,w_2) = \mathrm{S^H}(v) + 2$. By the inductive hypothesis, w_2 satisfies inequality (A) and hence, $h(w_2) \geq t' - \mathrm{S^H}(w_2) + 2$. Both, together with $h(v) = h(w_2) + 2$ yields $h(v) \geq t' - \mathrm{S^H}(v) + 2$.

This completes the proof of Proposition 3. $\square$

Let t' and V' be as defined in Proposition 3. Proposition 3 yields that the jobs of V' that are scheduled before time t' have a special structure: when a job v is completed, all direct successors w with $c(v,w) = 0$ and exactly on direct successor w' with $c(v,w')=1$ is scheduled directly at time $\mathrm{C^H}(v)$, while all the remaining direct successors are scheduled one time unit later. Hence, define $U=\{v\in V' \mid \mathrm{S^H}(v)\le t'+1\}$ and let S' denote the sub–schedule of $\mathrm{S^H}$ induced by U, i.e. $S'(u)=\mathrm{S^H}(u)$ for each $u\in U$. Note that all jobs processed at time $t'+1$ are contained in U.

With the following definitions, every feasible schedules for U decomposes into several sub–schedules. For a job $u\in U$, denote its *earliest starting time* without respecting the communication delays and the machine restrictions by $\mathrm{est}(u)$. For a tree, $\mathrm{est}(u)$ is the sum of the processing times over all predecessors of u, and, in the case of unit processing times, $\mathrm{est}(u)=\sharp Pred(u)$.

Let us now consider the following subsets of U.

$$\begin{aligned} \overline{U}_j &= \{v\in U \mid \mathrm{S^H}(v)-\mathrm{est}(v)=j\} \quad \text{and} \\ U_j &= \{v\in U \mid \mathrm{S^H}(v)-\mathrm{est}(v)\ge j\} \end{aligned}$$

for every $j=0,\dots,\ell$, where ℓ is a value that depends on $\mathrm{S^H}$ but fulfills $\ell<m$ as we will see later. Each job set $\overline{U}_j$ induces a subtree of Θ. Let $\overline{w}_j$ denote the width (the maximum size of an anti-chain) of that subtree. Then, $\overline{w}_j$ denotes the maximum number of jobs of $\overline{U}_j$ that can be scheduled in parallel by any feasible schedule for the job–set $\overline{U}_j$. Proposition 3 now states that $\sum_{i=0}^{\ell}\overline{w}_i=m$ since the first time when m jobs of V' are scheduled in parallel is $t'+1$. With $\overline{w}_i\ge 1$ for each $i=0,\dots,\ell$, this yields $\ell\le m-1$.

Let S'_j denote the sub–schedule of $\mathrm{S^H}$ induced by U_j and translated to a minimal starting time of 0, i.e.

$$S'_j(v)=\mathrm{S^H}(v)-t_j \quad \text{for each } v\in U_j \tag{4}$$

with $t_j=\min\{S'(v)\mid v\in U_j\}$. Obviously, the maximum number of jobs of U_j that can be scheduled in parallel by any feasible schedule is $m_j=\sum_{i=j}^{\ell}\overline{m}_i$.

Proposition 3 states that in S' every job v is scheduled directly after its direct predecessor $u=father(v)$ with the exception of all but one non–favored jobs v with $c(u,v)=1$. Notice that for these non–favored jobs v, Proposition 3 yields $\mathrm{S^H}(w)=\mathrm{C^H}(v)+1$. This and $\mathrm{est}(v)=\mathrm{est}(u)+p(u)$ for $u=father(v)$ leads to

$$\mathrm{S^H}(v)-\mathrm{est}(v)=\mathrm{S^H}(u)+p(u)+c(u,v)-(\mathrm{est}(u)+p(u))=\mathrm{S^H}(u)-\mathrm{est}(u)+1$$

Thus, for each job $v\in U_{j+1}$ that is minimal in U_{j+1}, the direct predecessor u is contained in U_j. Let X_j denote the set of all direct predecessors of minimal jobs $v\in U_{j+1}$, for $j=1,\dots,\ell$.

Resuming, all non–minimal jobs in $\overline{U}_j$ are scheduled directly after their direct predecessors while every minimal job $v\in\overline{U}_j$ is a direct successor of a job $u\in X_{j-1}$ with $c(u,v)=1$ (for $j=2,\dots,\ell$). Clearly, the minimal jobs v of $\overline{U}_j$ satisfy $\mathrm{S^H}(v)=t_j$ for every $j=1,\dots,\ell$.

Figure 2 depicts the structure of schedule S'. This leads to Proposition 4 which is the core of the proof of Theorem 2.

Proposition 4. *Consider the schedule* $\mathrm{S^H}$ *determined by algorithm* DETERMINE-SCHEDULE *for an instance I of the problem $P|prec{=}forest, p_j=1, c_{ij}\in\{0,1\}|C_{\max}$. Let S'_j denote the sub–schedules of* $\mathrm{S^H}$ *defined by (4) and let S_j^{opt} denote an optimal schedule for the job–set U_j on w_j machines. Then,*

$$\sharp\{\textit{idle times in } S'_j\}-\sharp\{\textit{idle times in } S_j^{opt}\} \quad \le \quad \sum_{i=j}^{\ell}(i-j)\overline{w}_i-\ell+j\,. \tag{5}$$

Proof. The proof is done by induction on the job set U_j for $j=\ell,\dots,0$. The inductive basis is done for U_ℓ. Notice that U_ℓ only consists of w_ℓ jobs scheduled at time $t'+1$ since $t'+1$ is the first time that m jobs of U are scheduled in parallel by schedule S'. Since the number of machines for schedule S'_ℓ is defined as m_ℓ, S'_ℓ contains no idles and so does every optimal schedule S_ℓ^{opt} for U_ℓ. This proves inequality 5.

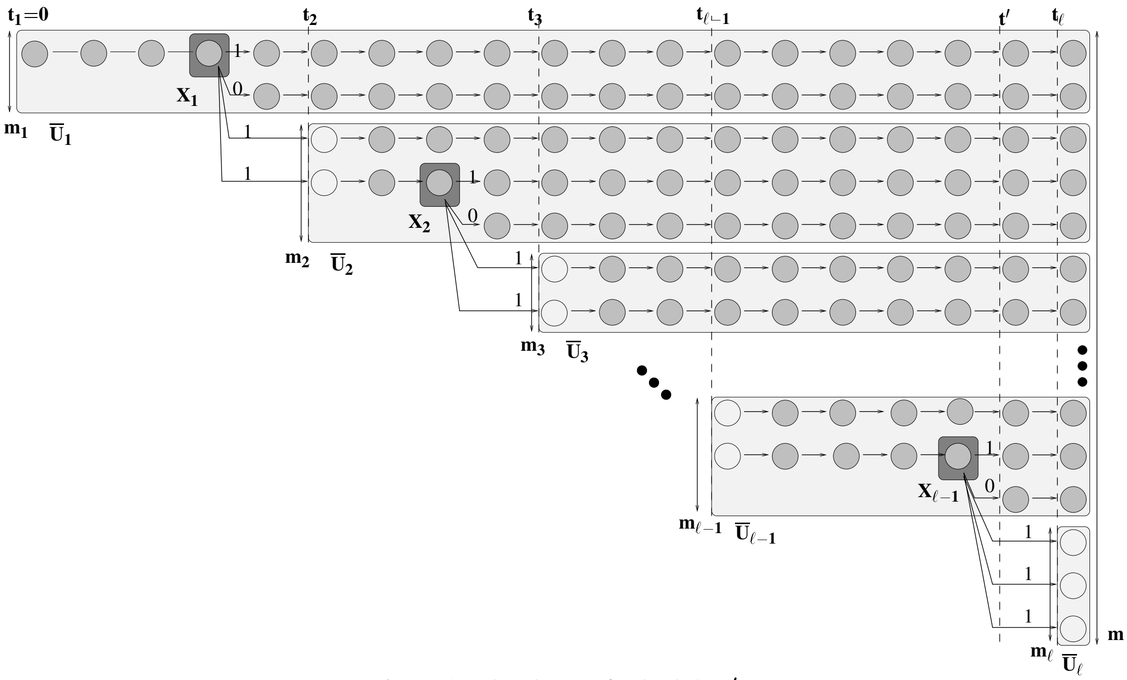

Figure 2: The shape of schedule S'

For the inductive step, assume as the inductive hypothesis that inequality (5) is satisfied by U_{j+1}. We now verify inequality (5) for U_j. Since at most $w_j \leq m$ jobs can be scheduled in parallel by any feasible schedule for U_j, every such schedule partitions into the sub-schedules induced by $\overline{U}_j$ and U_{j+1}.

Consider an optimal schedule S_j^{opt} for U_j. Since every job $x \in X_j$ has two or more direct successors y with $c(x,y) = 1$, S_j^{opt} contains an idle time at time $S_j^{opt}(x) + 1$ for every $x \in X_j$.

On the other hand, $S'_j(y) - C'_j(x) = 1$ and $S_j^{opt}(y) - C_j^{opt}(x) \geq 0$. Since all jobs of U_j are scheduled as early as possible in S'_j and the remaining jobs of U_{j+1} are scheduled after jobs of X_j,

$$\begin{aligned}
\sharp\{\textit{idle times in } S'_j\} - \sharp\{\textit{idle times in } S_j^{opt}\} &\leq \sharp\{\textit{idle times in } S'_{j+1}\} - \sharp\{\textit{idle times in } S_{j+1}^{opt}\} + w_{j+1} - \sharp X_j \\
&\overset{\text{hyp.}}{\leq} \sum_{i=j+1}^{\ell} (i-(j+1))\overline{w}_i - \ell + (j+1) + \sum_{i=j+1}^{\ell} \overline{w}_i - 1 \\
&= \sum_{i=j}^{\ell} (i-j)\overline{w}_i - \ell + j\,.
\end{aligned}$$

□

Theorem 2. *Algorithm* DETERMINE-SCHEDULE *produces a feasible schedule* $\mathrm{S^H}$ *for all instances* $I = (m, V, \Theta, c)$ *of the problem* $P|prec{=}forest,\ p_j = 1,\ c_{ij} \in \{0,1\}|C_{\max}$ *such that*

$$C_{\max}(\mathrm{S^H}) \quad \leq \quad C_{opt}(I) + \frac{1}{2}(m-1). \tag{6}$$

Proof. For an instance I, consider the schedule $\mathrm{S^H}$ produced by algorithm DETERMINE-SCHEDULE. Lemma 3 states that $\mathrm{S^H}$ is feasible. The remainder follows from Proposition 4. Note that if t' does not exists since $\mathrm{S^H}$ does not schedule m jobs in parallel before time $C_{\max}(\mathrm{S^H}) - 1$, Lemma 1 and Lemma 2 yield that $\mathrm{S^H}$ is optimal. Hence assume in the following that t' and V' are well–defined.

Note that $S'_0(v) = \mathrm{S^H}(v)$ for all $v \in V'$ such that $\mathrm{S^H}(v) \leq t'+1$ and that all jobs contained in $V \setminus V'$ are completed before time $t'+1$. Hence, the number of idle times in $\mathrm{S^H}$ is exactly the number of idle times in S'_0 minus $\sharp(V \setminus V')$

plus the number of idle times in the last time slot of S^{H}, which is at most $(m-1)$. Thus,

$$\sharp\{\textit{idle times in } \mathrm{S}^{\mathrm{H}}\} \leq \sharp\{\textit{idle times in } S_0'\} - \sharp(V \setminus V') + (m-1)\,.$$

Similar, the number of idle times in any optimal schedule S_{opt} for the instance I is at least the number of idle times of any optimal schedule S_0^{opt} for U_0 minus $\sharp(V \setminus V')$. This, together with inequality (5) yields

$$\sharp\{\textit{idle times in } \mathrm{S}^{\mathrm{H}}\} - \sharp\{\textit{idle times in } S_{opt}\} \leq \sharp\{\textit{idle times in } S_0'\} + (m-1) - \sharp\{\textit{idle times in } S_0^{opt}\}\,. \tag{7}$$

Now, apply Proposition 4. Inequality $(\star)$ follows since $\sum_{i=0}^{\ell}(i \cdot \overline{w}_i) - \ell$ becomes maximum if $\overline{w}_i = 1$ for each $i = 0, \dots, \ell$ and $\ell = m-1$. Note that $\ell \leq m-1$ and $\overline{w}_i \geq 1$, $i = 0, \dots, \ell$.

$$\begin{aligned}
C_{\max}(\mathrm{S}^{\mathrm{H}}) - C_{opt}(I) &\overset{(1)}{=} \frac{\sharp\{\textit{idle times in } \mathrm{S}^{\mathrm{H}}\} - \sharp\{\textit{idle times in } S_{opt}\}}{m} \\
&\overset{(7)}{\leq} \frac{\sharp\{\textit{idle times in } S_0'\} - \sharp\{\textit{idle times in } S_0^{opt}\} + (m-1)}{m} \\
&\overset{(5)}{\leq} \frac{\sum_{i=0}^{\ell}(i \cdot \overline{w}_i) - \ell + (m-1)}{m} \\
&\overset{(\star)}{\leq} \frac{\sum_{i=0}^{m-1} i}{m} = \frac{m-1}{2}
\end{aligned}$$

This completes the proof of Theorem 2. □

3.3 Implementation and Running Time

In this subsection, we give some suggestions on the data structures to use in an implementation of the above algorithm. More details on the used techniques like *depth-first search* and *bucket sort* can be found in the book of Cormen, Leiserson, and Rivest [CLR90], for example.

Preprocessing

The priority of all jobs can be computed in $\mathcal{O}(n)$ time by scanning every tree in the forest bottom-up by methods like *depth-first search*.

Main Loop

Using a data-structure that allows to find all $\delta(v) - 1$ successors of a job v in $\mathcal{O}(\delta(v))$ time makes it easy to realize Step (2b) up to Step (3) of the algorithm in an overall linear time. Note that every job is rejected at most once by Step (2c) since we consider zero-one communication delays.

The height-value increases by 2 if and only if a job u has at least two direct successors v and v' with $h(v) = h(v') = h(u) - 2$. Hence, the overall maximum height-value can be bounded by $h_{\max} \leq n$ where n denotes the number of jobs, $n = |V|$.

Thus, use as the mentioned data structure an array of n "buckets", one for each possible height-value occurring during the execution of algorithm DETERMINE-SCHEDULE. The k-th bucket contains a list of all jobs in *avail* having the height value k. Using this data structure, a job of maximum height value can be extracted in $\mathcal{O}(1)$ time in Step (2a) after looking for the first non-empty bucket in the array. The insertion of a newly available job in Step (3) can also be done in $\mathcal{O}(1)$ time. Since jobs are only inserted into the part of the array that has not been visited before, no component of the array will be visited more than once.

This leads to an overall running time of algorithm DETERMINE-SCHEDULE of $\mathcal{O}(n)$ time using $\mathcal{O}(n)$ space.

4 An Example

As an example we consider the extreme case presented by Guinand, Rapine and Trystram [GRT95, §6]. This example is also an extreme case for our algorithm.

The idea is to recursively construct a tree A_m. Begin with A_1 consisting of a root r_1 and 2 successors. To define $A_m, m \geq 2$ introduce a new node r_m as the root of A_m. As the successors of r_m take the root r_{m-1} of the subtree A_{m-1} and the first node of a chain of size $h(r_{m-1})$. Creating A_m, another $m-1$ successors are added to r_1 summing up to a total of $2+m(m-1)/2$. To obtain the corresponding problem instance, set all processing times and communication delays to 1. To obtain the extreme case, set the number of machines to be m.

Figure 3 depicts the tree A_3 for the cases of 3 and 4 machines. Thereby, the numbers inside the circles denote the height value $h(v)$ of the job v corresponding to the node in the tree. Clearly, $h(r_m) = h(r_{m-1}) + 2 = 2m+1$ due to the definition of the height function in Section 3.1.

Now apply algorithm DETERMINE-SCHEDULE to the instance A_3 . Since all successors of a job v have the same height value, Step (2a) of the algorithm may choose any successor when a job ends.

Figure 4 depicts different schedules determined by different choices of favored successors.

- For the 4-machine case, S_4° is the resulting schedule when the jobs denoted by non-filled circles in Figure 3 are chosen as favored successors. Similarly, $S_4^\bullet$ denotes the schedule determined when the jobs represented by filled circles are selected. Note that for $m = 4$, every choice of favored successors yields an optimal schedule.

- For the case $m = 3$, the notation s similar. Here, S_3° is an optimal schedule while $S_3^\bullet$ is a schedule of maximum length, algorithm DETERMINE-SCHEDULE can produce for the instance A_3. Hence, every feasible schedule S for A_3 fulfills the stated bound $C_{\max}(S) \leq C_{opt}(A_3) + \frac{m-1}{2}$. For $S_3^\bullet$, this bound is tight: $C_{\max}(S_3^\bullet) = C_{opt}(A_3) + \frac{m-1}{2} = 8$.

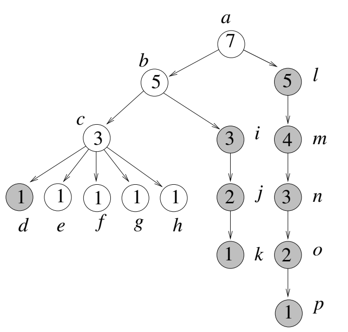

Figure 3: Example instance A_3

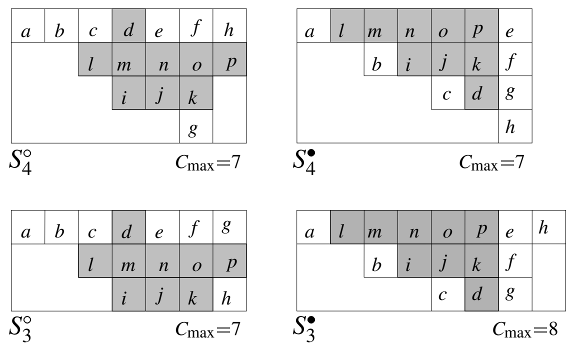

Figure 4: Possible schedules for instance A_3

5 Concluding Remarks

The bound of $\frac{m-1}{2}$ is tight if no further condition is made on the choice of the favored successor of a job that has more than one successor of of maximum height. Consider the example given in [GRT95, §6] as an extremal case. For $m = 3$, the corresponding instance is discussed in Section 4.

Theorem 1 and Theorem 2 also hold when *release dates* are introduced, i.e. when a job u may not be started before a given time $r(u)$. To incorporate release dates into the algorithm, include the following condition into Definition 1:
(iii) $t \geq r(u)$.

Theorem 1 can be extended to series-parallel orders and unit communication delays. In [MS95] we show in detail how to define the height function and formulate the modified priority list scheduling algorithm. The obtained scheduling algorithm has an overall computation time of $\mathcal{O}(|V|+|E|)$ if the processing times are bounded by a constant or $\mathcal{O}(|E|+|V|\log|V|)$ for unbounded processing times. Here, V denotes the set of jobs while E denotes the set of precedence relations in the series-parallel order.

In [MH95], Munier and Hanen bounded the number of idle times in a schedule produced by a modified list scheduling algorithm by $(m-1)C^{\infty}_{opt}(I)$ which leads to an upper bound of $\left(2-\frac{1}{m}\right)$ for the problem when job–duplication is allowed, i.e. $P|dup, prec, p_j=1, c_{ij}=1|C_{\max}$. Let $p_{\max}=\max\{p(v) \mid v \in V\}$. Following the proof of Munier and Hanen, the stated bound of $\left(2-\frac{1}{m}\right)$ can be generalized to the problem $P|dup, prec, c_{ij} \in \{0,1\}|C_{\max}$, obtaining

$$C_{\max}(S) \leq \left(\frac{p_{\max}+3}{2}-\frac{1}{m}\right) C_{opt}(I)+\frac{p_{\max}-1}{2}.$$

Due to the symmetry of the communication delays, it does not matter whether to schedule in-forests or out-forests when job–duplication is not allowed.S_{opt} for the job set V. Since job–duplication is of no use for in-forests, the mentioned bounds also hold for the problem $P|prec{=}forest, c_{ij} \in \{0,1\}|C_{\max}$.

While the problem $P|dup, prec{=}in\text{-}forest, p_j=1, c_{ij} \in \{0,1\}\}|C_{\max}$ is NP–complete [Vel93], the corresponding problem for out-trees can be solved optimally in linear time. One way to see this, is to look for the first time t with $\sharp\{v \in V \mid h(v)=t\} \geq m$. Select arbitrarily m of these jobs and for for each of them, introduce one duplicate of each of their predecessors (direct and indirect). Applying algorithm DETERMINE-SCHEDULE to the resulting job set V' determines an optimal schedule since the duplication of the predecessors allows each of the selected jobs v to be scheduled at time $h(r)-h(v)$, i.e. as early as possible, while from time t on, all machines are kept busy (the proof for this is similar to the proof of Proposition 3).

References

[Chr89] Philippe Chrétienne, *A polynomial algorithm to optimally schedule tasks on virtual distributed systems under tree-like precedence constraints*, Europ. Journal Oper. Res. **43** (1989), 225–236.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[GLLRK79] Ronald L. Graham, Eugene L. Lawler, Jan K. Lenstra, and Alexander H.G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: a survey*, Annals of Discrete Math. **5** (1979), 287–326.

[GRT95] Frédéric Guinand, Christophe Rapine, and Denis Trystram, *Worst case analysis of Lawler's algorithm for scheduling trees with communication delays*, Tech. report, LMC-IMAG, 46 avenue Fèlix Viallet, 38031 Grenoble, France, {Guinand,Rapine,Trystram}@imag.fr, 1995.

[HM95] Claire Hanen and Alix Munier, *An approximation algorithm for scheduling dependent tasks on m processors with small communication delays*, Tech. report, Laboratoire Informatique Théorique et Programmation, Institut Blaise Pascal, Université Pierre et Marie Curie, 4, place jussieu, 75252 Paris cedex 05, 1995.

[Jaf80] Jeffrey M. Jaffe, *Efficient scheduling of tasks without full use of processor resources*, Theoretical Computer Science **12** (1980), 1–17.

[Law93] Eugene L. Lawler, *Scheduling trees on multiprocessors with unit communication delays*, Presented at the First Workshop on Models and Algorithms for Planning and Scheduling Problems, unpublished script, June 1993.

[LLRKS93] Eugene L. Lawler, Jan K. Lenstra, Alexander H.G. Rinnooy Kan, and David B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, Logistics of Production and Inventory (S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, eds.), Handbooks in Operations Research and Management Science, vol. 4, North Holland, 1993, pp. 445–522.

[LVV93] Jan K. Lenstra, Marinus Veldhorst, and Bart Veltman, *The complexity of scheduling trees with communication delays*, Proceedings of the First Annual European Symposium on Algorithms (Th. Lengauer, ed.), Springer-Verlag, Sept. 1993.

[MH94] Alix Munier and Claire Hanen, *An approximation algorithm for scheduling unitary tasks on m processors with communication delays*, Tech. report, Laboratoire Informatique Théorique et Programmation, Institut Blaise Pascal, Université Pierre et Marie Curie, 4, place jussieu, 75252 Paris cedex 05, 1994.

[MH95] Alix Munier and Claire Hanen, *Using duplication for scheduling unitary tasks on m processors with communication delays*, Tech. report, Laboratoire Informatique Théorique et Programmation, Institut Blaise Pascal, Université Pierre et Marie Curie, 4, place jussieu, 75252 Paris cedex 05, 1995, LITP 95/47.

[MS95] Rolf H. Möhring and Markus W. Schäffter, *Approximation algorithms for scheduling series-parallel orders subject to unit time communication delays*, Tech. Report No. 483/1995, University of Technology, Berlin, Straße des 17.Juni 136, 10623 Berlin, Germany, 1995, ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-483-1995.ps.Z.

[Pic92] Christophe Picouleau, *Étude de problèmes d'optimisation dans les systèmes distribués*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, 1992.

[RS87] Victor J. Rayward-Smith, *UET scheduling with unit interprocessor communication delays*, Discrete Applied Math. **18** (1987), 55–71.

[Vel93] Bart Veltman, *Multiprocessor scheduling with communication delays*, Ph.D. thesis, University of Technology Eindhoven, Department of Mathematics and Computer Science, Eindhoven, Netherlands, 1993.

[VRKL94] Theodora A. Varvarigou, Vwani P. Roychowdhury, Thomas Kailath, and Eugene L. Lawler, *Scheduling in and out forests in the presence of communication delays*, IEEE Trans. Parallel Distributed Systems (1994), (to appear).

Reports from the group

"Algorithmic Discrete Mathematics"

of the Department of Mathematics, TU Berlin

506/1996 *Rolf H. Möhring and Markus W. Schäffter:* A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

504/1996 *Uta Wille:* The Role of Synthetic Geometry in Representational Measurement Theory

502/1996 *Nina Amenta and Günter M.Ziegler:* Deformed Products and Maximal Shadows of Polytopes

498/1996 *Ewa Malesinska, Alessandro Panconesi:* On the Hardness of Allocating Frequencies for Hybrid Networks

496/1996 *Jörg Rambau:* Triangulations of Cyclic Polytopes and higher Bruhat Orders

489/1995 *Eva Maria Feichtner and Dmitry N. Kozlov:* On Subspace Arrangements of Type $\mathcal{D}$

483/1995 *Rolf H. Möhring and Markus W. Schäffter:* Approximation Algorithms for Scheduling Series-Parallel Orders Subject to Unit Time Communication Delays

478/1995 *Sven G. Bartels:* The complexity of Yamnitsky and Levin's simplices algorithm

477/1995 *Jens Gustedt, Michel Morvan and Laurent Viennot:* A compact data structure and parallel algorithms for permutation graphs, to appear in : Nagl et al., editors, *Graph-Theoretic Concepts in Computer Science*, Proccedings of the 20th International Workshop WG '95.

476/1995 *Jens Gustedt:* Efficient Union-Find for Planar Graphs and other Sparse Graph Classes

475/1995 *Ross McConnell and Jeremy Spinrad:* Modular decomposition and transitive orientation

474/1995 *Andreas S. Schulz:* Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP–Based Heuristics and Lower Bounds

473/1995 *Günter M. Ziegler:* Shelling Polyhedral 3-Balls and 4-Polytopes

472/1995 *Martin Henk, Jürgen Richter-Gebert and Günter M. Ziegler:* Basic Properties of Convex Polytopes

471/1995 *Jürgen Richter-Gebert and Günter M. Ziegler:* Oriented Matroids

465/1995 *Ulrich Betke and Martin Henk:* Finite Packings of Spheres

463/1995 *Ulrich Hund:* Every simplicial polytope with at most $d+4$ vertices is a quotient of a neighborly polytope, to appear in Discrete & Computational Geometry

462/1995 *Markus W. Schäffter:* Scheduling with Forbidden Sets

461/1995 *Markus W. Schäffter:* Drawing Graphs on Rectangular Grids with at most 2 Bends per Edge

458/1995 *Ewa Malesińska:* List Coloring and Optimization Criteria for a Channel Assignment Problem

447/1995 *Martin Henk:* Minkowski's second theorem on successive minima

441/1995 *Andreas S. Schulz, Robert Weismantel, Günter M. Ziegler:* 0/1–Integer Programming: Optimization and Augmentation are Equivalent, appeared in Paul Spirakis (ed.): Algorithms – ESA '95, Lecture Notes in Computer Science 979, Springer: Berlin, 1995, pp. 473-483

440/1995 *Maurice Queyranne, Andreas S. Schulz:* Scheduling Unit Jobs with Compatible Release Dates on Parallel Machines with Nonstationary Speeds, appeared in Egon Balas and Jens Clausen (eds.): Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 920, Springer: Berlin, 1995, pp. 307-320

439/1995 *Rolf H. Möhring, Matthias Müller-Hannemann:* Cardinality Matching: Heuristic Search for Augmenting Paths

436/1995 *Andreas Parra, Petra Scheffler:* Treewidth Equals Bandwidth for AT-Free Claw-Free Graphs

432/1995 *Volkmar Welker, Günter M. Ziegler, Rade T. Živaljević:* Comparison Lemmas and Applications for Diagrams of Spaces

431/1995 *Jürgen Richter-Gebert, Günter M. Ziegler:* Realization Spaces of 4-Polytopes are Universal, to appear in *Bulletin of the American Mathematical Society*, October 1995.

430/1995 *Martin Henk:* Note on Shortest and Nearest Lattice Vectors

429/1995 *Jörg Rambau, Günter M. Ziegler:* Projections of Polytopes and the Generalized Baues Conjecture

428/1995 *David B. Massey, Rodica Simion, Richard P. Stanley, Dirk Vertigan, Dominic J. A. Welsh, Günter M. Ziegler:* Lê Numbers of Arrangements and Matroid Identities

408/1994 *Maurice Queyranne, Andreas S. Schulz:* Polyhedral Approaches to Machine Scheduling

407/1994 *Andreas Parra, Petra Scheffler:* How to Use the Minimal Separators of a Graph for Its Chordal Triangulation

401/1994 *Rudolf Müller, Andreas S. Schulz:* The Interval Order Polytope of a Digraph, appeared in Egon Balas and Jens Clausen (eds.): Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 920, Springer: Berlin, 1995, pp. 50-64

396/1994 *Petra Scheffler:* A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-width

394/1994 *Jens Gustedt:* The General Two-Path Problem in time $O(m \log n)$, extended abstract

393/1994 *Maurice Queyranne:* A Combinatorial Algorithm for Minimizing Symmetric Submodular Functions

392/1994 *Andreas Parra:* Triangulating Multitolerance Graphs

390/1994 *Karsten Weihe:* Maximum (s,t)–Flows in Planar Networks in $\mathcal{O}(|V|\log|V|)$ Time

386/1994 *Annelie von Arnim, Andreas S. Schulz:* Facets of the Generalized Permutahedron of a Poset, to appear in *Discrete Applied Mathematics*

383/1994 *Karsten Weihe:* Kurzeinführung in C++

377/1994 *Rolf H. Möhring, Matthias Müller-Hannemann, Karsten Weihe:* Using Network Flows for Surface Modeling

376/1994 *Valeska Naumann:* Measuring the Distance to Series-Parallelity by Path Expressions

375/1994 *Christophe Fiorio, Jens Gustedt:* Two Linear Time Union-Find Strategies for Image Processing

374/1994 *Karsten Weihe:* Edge–Disjoint (s,t)–Paths in Undirected Planar graphs in Linear Time

373/1994 *Andreas S. Schulz:* A Note on the Permutahedron of Series–Parallel Posets, appeared in *Discrete Applied Mathematics* 57 (1995), pp. 85-90

371/1994 *Heike Ripphausen–Lipa, Dorothea Wagner, Karsten Weihe:* Efficient Algorithms for Disjoint Paths in Planar Graphs

Reports may be requested from:

S. Marcus
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: Marcus@math.TU-Berlin.DE

Reports are available via anonymous ftp from:

ftp.math.tu-berlin.de
cd pub/Preprints/combi
file Report–<number>–<year>.ps.Z