1992

J.A. Hoogeveen, J.K. Lenstra, B. Veltman

Three, four, five, six, or the complexity of scheduling with communication delays

Department of Operations Research, Statistics, and System Theory Report BS-R9229 October

CWI is het Centrum voor Wiskunde en Informatica van de Stichting Mathematisch Centrum
CWI is the Centre for Mathematics and Computer Science of the Mathematical Centre Foundation

CWI is the research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a non-profit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands organization for scientific research (NWO).

Copyright © Stichting Mathematisch Centrum, Amsterdam

Three, four, five, six, or the Complexity of Scheduling with Communication Delays

J.A. Hoogeveen
Department of Civil Engineering and Operations Research
Princeton University
Eng. Quad E-220, Princeton, NJ 08544, USA

J.K. Lenstra
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

B. Veltman
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract: A set of unit-time jobs has to be processed on identical parallel machines subject to precedence constraints and unit-time communication delays; does there exist a schedule of length at most L? The problem has two variants, depending on whether the number of machines is restrictively small or not. For the first variant, the question can be answered in polynomial time for $L = 3$ and is *NP*-complete for $L = 4$. The second variant is solvable in polynomial time for $L = 5$ and *NP*-complete for $L = 6$. As a consequence, neither of these problems has a polynomial approximation scheme, unless $P = NP$.

1991 Mathematics Subject Classification: 90B35.
Key Words & Phrases: Identical parallel machines, precedence constraints, communication delays, makespan, complexity, approximation.

1. Introduction

The introduction of parallel computing has led to a new obstacle that has to be dealt with when we wish to process a set of precedence-constrained jobs: the execution of a job cannot start before all necessary information produced by its predecessors has been transmitted to the machine that has to process the job. The resulting delays in the processing of the job are referred to as *communication delays*.

We consider the following type of scheduling problem with communication delays. There are n jobs J_j $(j = 1, \ldots, n)$ that have to be executed by m identical parallel machines M_i $(i = 1, \ldots, m)$. Each machine is continuously available from time zero onward and can handle at most one job at a time.

Report BS-R9229
ISSN 0924-0659
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Each job has a *processing time* of one time unit. A *Precedence relation* defines predecessors and successors for each job. It is represented by an acyclic directed graph G with vertex set $\{1,\ldots,n\}$. If G contains a directed path from j to k, then we write $J_j \rightarrow J_k$, and require that J_k cannot be started before J_j has been completed and the information produced by the execution of J_j has been transferred from the machine that executed J_j to the machine that is about to process J_k; the transmission of information does not interfere with the availability of the machines. The *communication delay* amounts to one unit of time if J_j and J_k are processed on different machines; there is no delay if J_j and J_k are processed on the same machine. A *schedule* defines for each job the time interval during which and the machine by which it is processed such that the machine availability constraints, the precedence constraints, and the communication delays are satisfied. Given a schedule σ, $S_j(\sigma)$ and $C_j(\sigma)$ denote the start and completion time of job J_j, respectively; we will use just S_j and C_j if there is no confusion possible as to the schedule we are referring to. We wish to minimize the *length* or *makespan*, that is, the maximum job completion time, defined as $C_{\max}=\max_{1\le j\le n} C_j$.

There are two variants of the problem, depending on whether the number of machines is restrictively small or not. Using the three-field notation scheme that was proposed by Veltman, Lageweg, and Lenstra [1991] as an extension of the terminology introduced by Graham, Lawler, Lenstra, and Rinnooy Kan [1979], the first variant is denoted by $P\,|prec,c=1,p_j=1\,|C_{\max}$, whereas the second variant is denoted by $\overline{P}\,|prec,c=1,p_j=1\,|C_{\max}$.

The $P\,|prec,c=1,p_j=1\,|C_{\max}$ problem was first addressed by Rayward-Smith [1987], who, after establishing NP-hardness, showed that the length of an *active* schedule, that is, a schedule in which no job can start earlier without increasing the start time of another job, is at most equal to $3-2/m$ times the makespan of the optimal schedule. Picouleau [1991B] considered the same problem, and showed that the problem of deciding whether an instance has a schedule of length at most 3 is decidable in polynomial time. For the case of an unrestrictively large number of machines, he established NP-completeness for the problem of deciding whether an instance has a schedule of length at most 8 [Picouleau, 1991A].

In this paper, we analyze the complexity of the types of decision problems introduced by Picouleau [1991A, 1991B]. We wish to determine the borderline the threshold value y has to exceed to change the problem from polynomially decidable into NP-complete. In Section 2, we include our own version of the proof that shows that the decision problem originating from the restricted variant of the problem is polynomially solvable if y is at most 3, and we show NP-completeness for the case $y \ge 4$. In Section 3, we show that the decision problem originating from the unrestricted variant of the problem is polynomially solvable if $y \le 5$ and NP-complete if $y \ge 6$.

As a consequence, there exists no polynomial-time algorithm with performance bound smaller than 5/4 for $P\,|prec,c=1,p_j=1\,|C_{\max}$, and there exists no polynomial-time algorithm with performance bound smaller than 7/6 for $\overline{P}\,|prec,c=1,p_j=1\,|C_{\max}$, unless P = NP. Neither of these problems has a polynomial approximation scheme, unless P = NP.

2. The restricted variant

In this section, we start by showing that the problem of deciding whether an instance has a schedule of length at most 3 is decidable in polynomial time. This problem has already been solved by Picouleau [1991B], but we include our own version of the proof in the paper for sake of completeness. Next, we prove NP-completeness for a special case of the problem of deciding whether an instance has a schedule of length at most 4, in which the precedence relation has the form of a

bipartite graph.

Theorem 1. *The problem of deciding whether an instance of* $P\,|prec,c=1,p_j=1|\,C_{\max}$ *has a schedule of length at most* 3 *is solvable in polynomial time.*

Proof. Given an instance of $P\,|prec,c=1,p_j=1|\,C_{\max}$, we first check whether the trivial necessary constraints for the existence of a feasible schedule with length no more than 3 are satisfied; these are the constraints that there are no paths in the graph with length more than 3, that the number of jobs amounts to no more than $3m$, and that no two paths with length 3 are dependent. Subsequently, we strip the set of independent jobs off the instance; these jobs are dealt with later.

Our approach to check the existence of a feasible schedule with length no more than 3 consists of two steps. We first assign the jobs to time slots. Subsequently, the jobs are assigned to the machines by which they have to be executed. The first step proceeds in such a way that the number of machines needed in the second step is minimized.

We first deal with the paths of length 3; the jobs in a path of length 3 are entirely assigned to a single machine. As no two paths of length 3 interfere, we have that the second job in a chain has only one predecessor and only one successor. The first job in a path of length 3 may be succeeded by several jobs without successors; these jobs are assigned to the third time period. The third job in a path of length 3 may be preceded by several jobs without predecessors; these jobs are assigned to the first time period. Furthermore, we assign the jobs with two or more successors to the first time period, and the jobs with two or more predecessors to the third time period.

The jobs that still have to be assigned belong to either a chain with length two, or a rooted intree or outtree with length at most equal to 2. In case of a chain with length 2, the jobs can be assigned to either the time periods one and two, to the time periods two and three, and to the time periods one and three; if a job is assigned to the second time period, then the other job in the chain has to be executed by the same machine. In case of a rooted intree, at most one of the jobs can be assigned to time period 2, and all other jobs must be assigned to time period 1, whereas, in case of a rooted outtree, at most one job can be assigned to time period 2, and all other jobs must be assigned to time period 3. A straightforward approach finds an assignment of these jobs, belonging to chains and trees, to time slots such that the maximum number of jobs assigned to a single time slot is minimized. If this number is greater than the number of available machines, then clearly there exists no schedule with length at most 3 for this instance.

Given an assignment of jobs to time periods, a feasible schedule is constructed in the following way. First assign each job that is to be processed in the second time period to a machine and assign its predecessor or successor to the same machine. The remaining jobs can be scheduled on arbitrary machines according to the time period assignment. Finally, the independent jobs can be used to fill the empty spots. □

Theorem 2. *The problem of deciding whether an instance of* $P\,|bipartite,c=1,p_j=1|\,C_{\max}$ *has a schedule of length at most* 4 *is* NP*-complete.*

Proof. Our proof is based on a reduction from the NP-complete problem Clique and extends the proof found in Lenstra and Rinnooy Kan [1978] for the variant without communication delays $P\,|prec,p_j=1|\,C_{\max}$.

Clique
Given a graph $G=(V,E)$ and an integer k, does G have a complete subgraph on k vertices?

Given an instance of Clique, define the number of edges in a clique of size k by $l=k(k-1)/2$ and define $m_1=\max\{|V|+l-k, |E|-l\}$. We construct the following instance of $P|prec,c=1,p_j=1|C_{\max}$. There are $m=2(m_1+1)$ machines that have to process $4m$ jobs. Each vertex $v\in V$ corresponds to a pair of vertex-jobs J_v and K_v, and each edge $e\in E$ corresponds to an edge-job L_e; we introduce precedence constraints $J_v\rightarrow K_v$ and $J_v\rightarrow L_e$ if v is incident to the edge e. In addition, we define $4m-2|V|-|E|$ dummy jobs: there are $m-k$ jobs of type U, $m-|V|$ of type W, $m-|V|+k-l$ of type X, and $m-|E|+l$ of type Y. The precedence constraints between these dummy jobs are such that all U jobs should precede all X and Y jobs, and all W jobs should precede all Y jobs.

Suppose that G contains a clique of size k. Then a schedule with length no more than 4 is obtained by scheduling the jobs according to the pattern depicted in Figure 1.

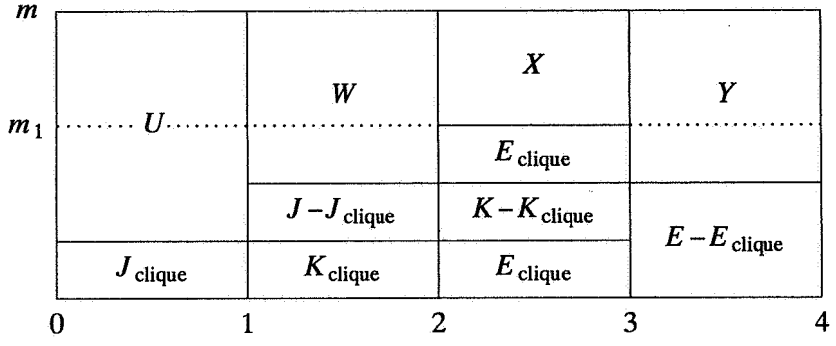

Figure 1. Schedule with makespan $C_{\max}\leq 4$.

Conversely, suppose that there exists a feasible schedule σ with length no more than 4. We will show that in any schedule with length 4 the non dummy jobs processed in time slot [0,1] correspond to the vertices of a clique of size k. The U jobs must be processed in time period [0,1] in σ, since they must precede all of the at least $m+2$ jobs of type X and Y. A similar argument shows that the Y jobs must be processed in time period [3,4] in σ. It follows immediately from these observations that the jobs of type W and X must be processed in σ in the time periods [0,2] and [2,4], respectively.

As the number of jobs is exactly equal to $4m$, we know that σ does not contain any idle time; hence, next to the jobs of type U and W, exactly $k+|V|$ vertex jobs must be processed in time period [0,2]. As the vertex jobs of type J have to precede the corresponding vertex jobs of type K, we know that at most $|V|$ vertex jobs are processed in time period [1,2] in σ; this observation, combined with the previous one, implies that in σ all W jobs are processed in time period [1,2], that k vertex jobs of type J are processed in time period [0,1], and that the corresponding vertex jobs of type K and the remaining vertex jobs of type J are processed in time period [1,2]. The set of jobs that are processed in time period [2,3] consists of X jobs, edge jobs L that have both predecessors processed in time period [0,1], vertex jobs of type K, and L jobs with one predecessor in time period [0,1] and one predecessor in time period [1,2]; the total number of these jobs amounts to m, as σ contains no idle time. Note that both the K jobs and the L jobs with one predecessor in time period [1,2] must be scheduled immediately after their preceding job of type J, implying that the number of these jobs amounts to at most

$|V|-k$. Hence, there are at least l edge jobs with both predecessors processed in time period [0,1], implying that the k vertices corresponding to the k vertex jobs that are processed in time period [0,1] induce a complete subgraph of G. □

Corollary 1. *There exists no polynomial-time algorithm with performance bound smaller than* 5/4 *for* $P\,|prec,c=1,p_j=1\,|\,C_{\max}$, *unless* P=NP. □

3. The unrestricted variant

This section concerns the variant for which the number of machines is not restrictively small. We first show that the problem of deciding whether an instance has a schedule of length at most 5 is solvable in polynomial time. Subsequently, we show that the problem of deciding whether an instance has a schedule of length at most 6 is NP-complete.

Theorem 3. *The problem of deciding whether an instance of* $\bar{P}\,|prec,c=1,p_j=1\,|\,C_{\max}$ *has a schedule of length at most* 5 *is solvable in polynomial time.*

Proof. Given an arbitrary instance of the problem $\bar{P}\,|prec,c=1,p_j=1\,|\,C_{\max}$, we first check whether the obviously necessary constraints hold; these are that the graph contains no path with length more than 5 and that there are no two interfering paths with length 5. Suppose that these constraints are satisfied. Then it is easy to see that all jobs that do not belong to a path of length 4 can be assigned to a machine and time period without violating any constraint. We now present a polynomial-time algorithm that checks whether a given set of paths of length 4 fits into a feasible schedule with length no more than 5.

Let $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$ denote a path of length 4. Without loss of generality, J_1 and J_4 can be processed in the first and last time period, respectively. We develop an algorithm to check in polynomial time whether there exists a feasible assignment of the middle jobs J_2 and J_3 to time periods, while taking care of the constraint that two dependent jobs that are assigned to two consecutive time periods must be processed on the same machine. We distinguish a number of cases. Suppose that J_2 has at least two predecessors, implying that J_2 cannot start before time 2; then we have to assign J_2, J_3, and J_4 to the last three time periods and they have to be processed by the same machine. Similarly, if J_3 has at least two successors, then it has to be executed in time slot [2,3], besides, J_1 and J_2 have to be processed by the same machine in the time periods [0,1] and [1,2], respectively.

The other cases require a more intricate procedure. For each unscheduled job J_j, we define the *depth* d_j as the length of the path to the root; d_j thus amounts to either 1 or 2. As J_j starts at time d_j or d_j+1 in any feasible schedule with length no more than 5, we have that $S_j = d_j + x_j$, with $x_j \in \{0,1\}$. The problem of assigning feasible start times to the jobs J_j has thus been transformed into the problem of feasibly assigning binary values to the variables x_j.

Consider the case depicted in Figure 2a. Let V denote the set of successors of J_1 that belong to a path of length 4; in this case we have $V = \{J_2, J_{2a}, J_{2b}\}$. As at most one of the jobs in V can be executed in time period [1,2], we have that the x variables corresponding to the jobs in V must satisfy the constraint $\Sigma_{j \in V} x_j \geq |V| - 1$. The other three cases we distinguish are depicted in Figures 2b to 2d. Analogous observations lead to similar constraints for each case; these constraints are shown at the right hand side of the graph. Note that in case 2b we have already assigned J_2 to time period [1,2], and that J_3 has already been assigned to time period [3,4] in case 2c. If two dependent jobs J_2 and J_3

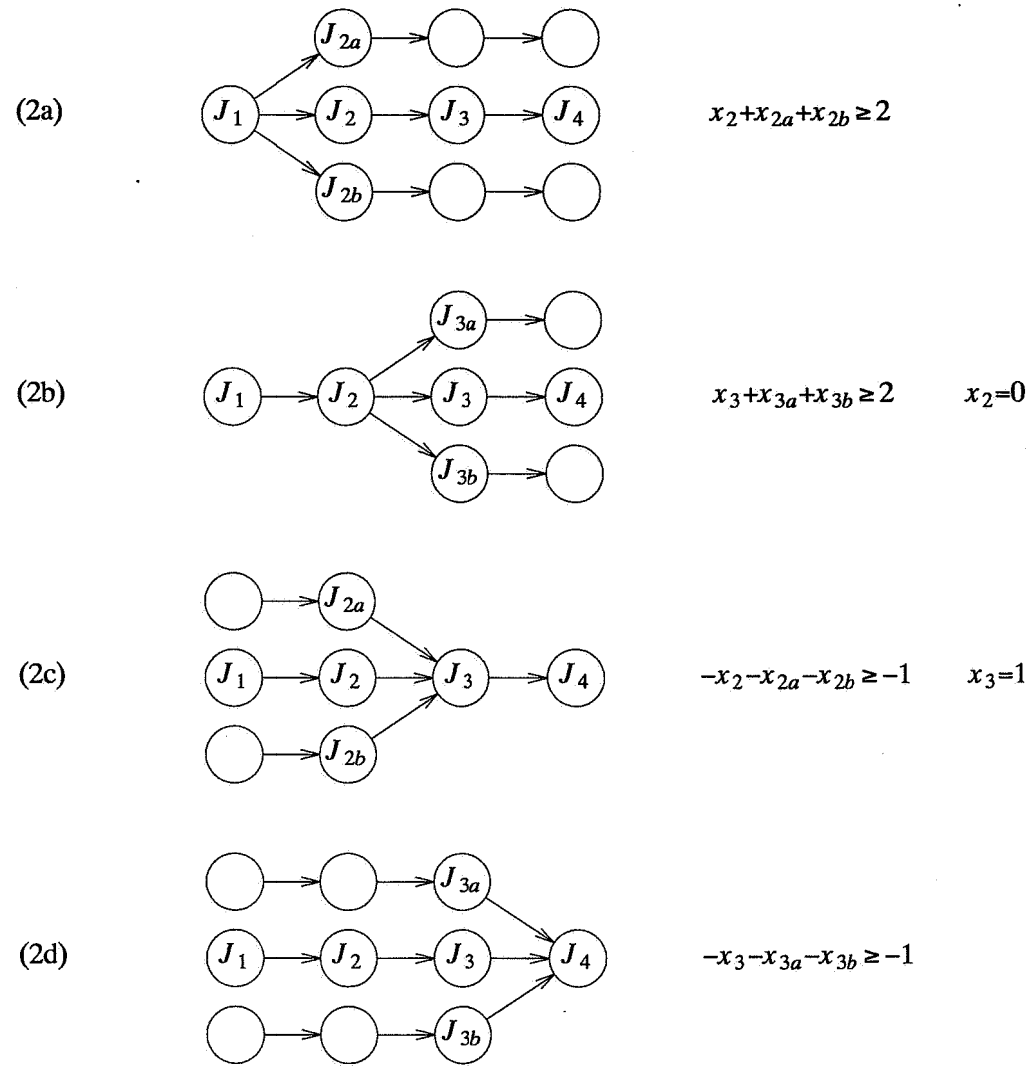

Figure 2. The four cases.

have not been assigned to time periods yet, then we have to add the constraint $x_3 \geq x_2$ to ensure consistency. Note that each binary solution that satisfies all constraints derived for the cases 2a through 2d induces a feasible schedule with length no more than 5; let $Ax \geq b$ denote the set of constraints. As every column of A contains at most one +1 and at most one −1 entry, A is a *network* matrix [Schrijver, 1986]. Hence, if we add the inequalities $0 \leq x_j \leq 1$, then the constraint matrix remains *totally unimodular*, implying that the polyhedron $\{x \mid 0 \leq x \leq 1; Ax \geq b\}$ is *integral*. As we can decide in polynomial time whether the polyhedron is empty, the problem whether for a given instance of $\bar{P} \mid prec, c=1, p_j=1 \mid C_{\max}$ there exists a schedule with length no more than 5 is decidable in polynomial time. □

Theorem 4. *The problem of deciding whether an instance of* $\overline{P}\,|prec,c=1,p_j=1|\,C_{\max}$ *has a schedule of length at most* 6 *is* NP*-complete.*

Proof. Our proof is based on a reduction from the NP-complete problem 3-Satisfiability.

3-Satisfiability
Given a set U of variables and a collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$, does there exist a truth assignment for C?.

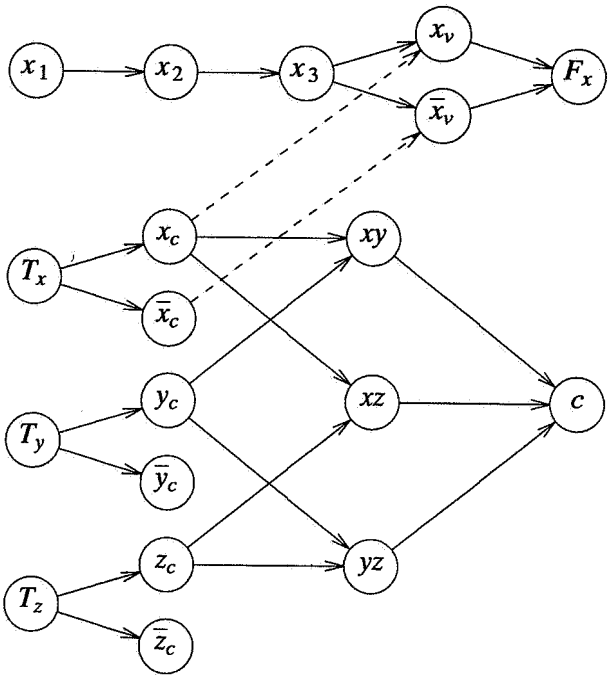

Figure 3. Variable-jobs and clause-jobs.

Given an instance (U,C) of 3-Satisfiability, we construct the following instance of $\overline{P}\,|prec,c=1,p_j=1|\,C_{\max}$. For each variable x we introduce 6 jobs: $x_1, x_2, x_3, x_v, \bar{x}_v$, and F_x; the precedence constraints between these jobs are depicted in Figure 3. For each clause $c=(x,y,z)$, where the literals x, y, and z can denote both negated and unnegated variables, we introduce 13 jobs: T_x, T_y, T_z, x_c, $\bar{x}_c$, y_c, $\bar{y}_c$, z_c, $\bar{z}_c$, xy, xz, yz, and c; the precedence constraints between these jobs are depicted in Figure 3. We further introduce precedence constraints between the variable jobs and the clause jobs. If the literal x occurs in clause c, then x_c has to precede x_v and $\bar{x}_c$ has to precede $\bar{x}_v$; these precedence constraints are depicted in Figure 3 by the dashed lines.

We start by making two essential observations. To begin with, note that in a schedule with length no more than 6 there are exactly two ways to schedule the jobs corresponding to a variable x, depending upon whether x_v is scheduled in time slot [3,4] and $\bar{x}_v$ in [4,5] or the other way around. In both cases, the jobs x_1, x_2, and x_3 have to be processed on the same machine as the variable-job that is scheduled in time slot [3,4], just like F_x and the variable-job scheduled in [4,5]. Secondly, note that

in order to schedule the clause-jobs corresponding to clause $c=(x,y,z)$ within 6 time units at least one of the jobs x_c, y_c, and z_c must be scheduled in time slot [1,2].

Suppose that a truth assignment for C exists. Then a schedule with length no more than 6 is obtained by scheduling the variable-job x_v in time slot [3,4], if x is true, and in time slot [4,5], otherwise. If the literal x occurring in clause c is true, then x_c is scheduled in time slot [1,2] on the same machine as T_x, and $\bar{x}_c$ is scheduled in time slot [2,3]; if the literal x is false, then the jobs x_c and $\bar{x}_c$ are switched. The other jobs are scheduled in a greedy manner. As every clause c contains at least one true literal, each clause-job c will be completed by time 6.

Conversely, suppose that there exists a schedule with length no more than 6. We will show that there exists a truth assignment for the instance of 3-Satisfiability. Define a variable x as true, if the corresponding variable-job x_v is processed in time slot [3,4], and false, otherwise. Without loss of generality, suppose that x_v is executed in time slot [3,4], implying that x_3 immediately precedes x_v on the same machine. Hence, each literal x_c must be scheduled in time slot [1,2], and each literal $\bar{x}_c$ in [2,3], implying that all literals are assigned values consistently. As each clause job c has been completed at time 6, we know that each clause contains at least one true literal. □

Corollary 2. *There exists no polynomial-time algorithm with performance bound smaller than* 7/6 *for* $\bar{P}\,|\,prec,c=1,p_j=1\,|\,C_{\max}$, *unless* P=NP. □

References

R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math. 5*, 287-326.

J.K. Lenstra, A.H.G. Rinnooy Kan (1978). Complexity of scheduling under precedence constraints. *Oper. Res. 26*, 22-35.

C. Picouleau (1991A). *Two new* NP*-complete scheduling problems with communication delays and unlimited number of processors,* RP91/24, MASI, Institut Blaise Pascal, Paris.

C. Picouleau (1991B). *Ordonnancement de tâches de durée unitaire avec des délais de communication unitaires sur m processeurs,* RP91/64, MASI, Institut Blaise Pascal, Paris.

V.J. Rayward-Smith (1987). UET scheduling with unit interprocessor communication delays. *Discrete Appl. Math. 18*, 55-71.

A. Schrijver (1986). *Theory of linear and integer programming,* Wiley, Chichester.

B. Veltman, B.J. Lageweg, J.K. Lenstra (1990). Multiprocessor scheduling with communication delays. *Parallel Comput. 16*, 173-182.