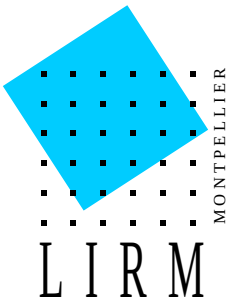
Complexity and Approximation for the Precedence Constrained Scheduling Problem with Large Communications Delays

Rodolphe Giroudeau, Jean-Claude König, Feryal Windal, Jérôme Palaysi



Laboratoire d'Informatique, de
Robotique et de Microélectronique de
Montpellier

Unité Mixte CNRS - Université Montpellier II 5506

RAPPORT DE RECHERCHE

Complexity and approximation for the precedence constrained scheduling problem with large communications delays

Rodolphe Giroudeau **Jean-Claude König**
Feryal-Kamila Moulai **Jérôme Palaysi**

mars 2005 R.R.LIRMM 05025

161, Rue Ada - 34392 Montpellier Cedex 5 - France
Tél: +33 (0)4 67 41 85 85 - Fax: +33 (0)4 67 41 85 00

Complexity and approximation for the precedence constrained scheduling problem with large communications delays

R. Giroudeau, J.C. König, F.K. Moulaï and J. Palaysi

LIRMM, 161 rue Ada,34392 Montpellier Cedex 5, France, UMR 5056

Abstract

We investigate the problem of minimizing the makespan (resp. the sum of the completion time) for the multiprocessor scheduling problem. We show that there is no hope of finding a ρ-approximation with $\rho < 1 + \frac{1}{c+4}$ (resp. $1 + \frac{1}{2c+5}$) (unless $\mathcal{P} = \mathcal{NP}$) for the case where all the tasks of the precedence graph have unit execution times, where the multiprocessor is composed of an unrestricted number of machines, and where c denotes the communication delay between two tasks i and j submitted to a precedence constraint and to be processed by two different machines. The problem becomes polynomial whenever the makespan is at the most $(c+1)$. The case where it is $(c+2)$ is still partially opened.

Résumé

Nous étudions le problème qui consiste à minimiser la durée totale d'un ordonnancement (resp. la somme des dates de fin d'exécution). Dans le cas où les tâches ont une durée d'exécution unitaire et où le nombre de processeurs est non borné, nous montrons qu'il n'existe pas (à moins que $\mathcal{P} = \mathcal{NP}$) de ρ-approximation telle que $\rho < 1 + \frac{1}{c+4}$ (resp. $1 + \frac{1}{2c+5}$) où c est le délais de communication entre 2 tâches i et j soumises à une contrainte de précédence et devant être exécutées sur 2 processeurs différents. Le problème devient polynomial lorsque la durée d'ordonnancement est au plus $c+1$ mais reste en partie ouvert lorsqu'il est égal à $c+2$.

1 Introduction

Scheduling theory is concerned with the *optimal allocation of scarce resources to activities over time*. The *theory* of the design of algorithms for scheduling is younger, but still has a significiant history.

In this article we adopt the *classical scheduling delay model* or *homogeneous model* in which an instance of a scheduling problem is specified by a set $J = \{j_1, \ldots, j_n\}$ of n nonpreemptive tasks, a set of U of q precedence constraints (j_i, j_k) such that $G = (J, U)$ is an directed acyclic graph(dag), the processing times $p_i, \forall j_i \in J$, and the communication times $c_{ik},\ \forall (j_i, j_k) \in U$.

If the task j_i starts its execution at time t on processor π and task j_k is a successor of j_i in the dag; then either j_k starts its execution after the time $t + p_{j_i}$ on processor π or after time $t + p_{j_k} + c_{j_i j_k}$ on some other processor. In what follows we consider the case of $\forall j_k \in J,\ p_{j_k} = 1$ and $\forall (j_i, j_k) \in E,\ c_{j_i j_k} = c \geq 2$.

This model was first introduced, by Rayward-Smith [15]. In this model we have a set of identical processors that are able to communicate in a uniform way. We want to use these processors in order to process a set of tasks that are subject to precedence constraints. The problem is to find a trade-off between the two extreme solutions, namely, execute all the tasks sequentially without communications, or try to use all the potential parallelism but at the cost of an increased communication overhead. This model has been extensively studied these last years both from the complexity and the (non)-approximability points of view [2].

Using the three fields notation scheme proposed by Graham et al. [6], the problem is denoted as

$$\bar{P}|prec, c_{ij} = c \geq 2; p_i = 1|C_{max}.$$

i.e. we have an unbounded number of identical processors in order to schedule a dag such that each task has the same execution time and each pair of tasks have the same communication time. The aim is to minimize the length of the schedule.

1.1 Complexity results

The problems with unitary communication delay In the case where we consider the problem is to schedule on an unbounded number of processors a precedence graph with unitary communications delays and unit execution time (UET-UCT), Hoogeveen et al. [8] proved that the decision problem associated to $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{\max}$ becomes $\mathcal{NP}$-complete even for $C_{\max} = 6$, and that it is polynomial for $C_{\max} = 5$. Their proof is based on a reduction from the $\mathcal{NP}$-complete problem $3SAT$ [4]. The $\mathcal{NP}$-completeness result for $C_{\max} = 6$ implies that there is no polynomial time approximation algorithm with ratio guarantee better than $7/6$, unless $\mathcal{P} = \mathcal{NP}$.

Moreover, in presence of a bounded number of processors, Hoogeveen et al. [8] establish that whether an instance of $P|prec; c_{ij} = 1; p_i = 1|C_{max}$ has a schedule of length of at the most 4 is $\mathcal{NP}$-complete (they use a reduction from the $\mathcal{NP}$-complete problem $Clique$), whereas Picouleau [13] develops a polynomial time algorithm for the $C_{\max} = 3$. In the same way, the $\mathcal{NP}$-completeness result for $C_{\max} = 4$ implies that there is no polynomial time approximation algorithm with ratio guarantee better than $5/4$, unless $\mathcal{P} = \mathcal{NP}$.

The problems with large communication delay In the case where we consider the problem to schedule on bounded number of processors a precedence graph with large communications delays and unit execution time (UET-LCT), Bampis et al. in [1] proved that the decision problem denoted by $P|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ for $C_{\max} = c + 3$ is $\mathcal{NP}$-complete problem, and for $C_{\max} = c + 2$ (for the special case $c = 2$), they develop a polynomial time algorithm. Their proof is based on a reduction from the $\mathcal{NP}$-complete problem *Balanced Bipartite Complete Graph*, $BBCG$ [4, 16]. Thus, Bampis et al. [1] proved that the $P|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ problem does not possess a polynomial time approximation algorithm with ratio guarantee better than $(1 + \frac{1}{c+3})$, unless $\mathcal{P} = \mathcal{NP}$.

Remark: Notice that in the case where the number of processors is unbounded ($\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$), the complexity to an associated decision problem is unknown.

Type of machines	c_{ij}	C_{max}	Complexity	Lower bound	References
$\bar{P}$	$c = 1$	5	Polynomial		[17]
$\bar{P}$	$c = 1$	6	$\mathcal{NP}$-complete [1]	$7/6 \leq \rho$	[17]
P	$c = 1$	3	Polynomial		[13]
P	$c = 1$	4	$\mathcal{NP}$-complete [2]	$5/4 \leq \rho$	[17]
$\bar{P}$	$c \geq 2$	$> c$	?	?	?
P	$c \geq 2$	$c + 1$	Polynomial		[1]
P	$c \geq 2$	$c + 3$	$\mathcal{NP}$-complete [3]	$1 + 1/(c + 3) \leq \rho$	[1]

Table 1. Complexity results for scheduling problems

The complexity results are given by the summary table 1.

[1] $3SAT$ see [4]
[2] $Clique$ see [4]
[3] $BBCG$ see [4, 16]

1.2 Approximation results

The problems with unitary communcation delay The best known approximation algorithm for $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ is due to Munier and König [12]. They presented a $(4/3)$-approximation algorithm for this problem, which is based on an integer linear programming formulation. They relax the integrity constraints and they produce a feasible schedule by rounding.

Munier and Hanen [11] proposed a $(\frac{7}{3} - \frac{4}{3m})$-approximation algorithm for the problem $P|prec; c_{ij} = 1; p_i = 1|C_{max}$. They define and study a new list scheduling approximation algorithm based on the solution given on an unrestricted number of processors. They introduce the notion of *favorite successor* in order to define priorities between conflicting successors of a task. Notice that, in the case where we consider the large communications delays, there is no polynomial time, as known, approximation algorithm except the trivial bound $(c + 1)$ whose first step consists in executing the tasks and second step in intiating communications phasis and so on ...

Concerning the case of the number where processors is restricted, an only (as known) constant 2-approximation algorithm is given by Munier [9], for the special case where the precedence graph is tree in presence of large communications delays.

The problems with large communcation delay Contrary to the complexity results, as we know, an unique appproximation algorithm is given by Rapine [14]. The autor gives the lower bound $O(c)$ for the list scheduling in presence of large communications delays. Approximation results are given by the table 2.

Type of machines	c_{ij}	Approximation results	References
$\bar{P}$	$c = 1$	$\rho \leq 4/3$	[11]
P	$c = 1$	$\rho \leq 7/3$	[10]
$\bar{P}$	$c \geq 2$	2 for tree	[9]
$\bar{P}$	$c \geq 2$	?	?
P	$c \geq 2$	$O(c)$	[14]

Table 2. Approximation results for scheduling problems

1.3 Presentation of an article

The challenge is to determinate a threshold for approximation algorithm for the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$, to develop a non trivial approximation algorithm, and to improve, in the presence of a restricted number of processors, the bound given by Rapine [14].

This article is organized as follows: in the second section, we give a preliminary result. In the third section, we give the two non-approximabilities results for the scheduling problem with the objective function of minimizing the length of the schedule (resp. minimization of the completion time). In the fourth section, we propose a polynomial time algorithm for $C_{max} = c + 2$ with $c \in \{2, 3\}$. In the last section, we develop a $\frac{2(c+1)}{3}$-approximation algorithm based on the notion of expansion, of the makespan, of a good feasible schedule.

2 Preliminary result

In this part, we will define a variant of SAT problem [4], denoted in what follows by Π_1. The $\mathcal{NP}$-completeness of the scheduling problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ (see section 3), is based on a reduction from this problem.

The problem Π_1 is a variant of the well known SAT problem [4]. We will call this variant the *One-in-*$(2,3)SAT(2,\bar{1})$ problem.We denote by $\mathcal{V}$, the set of variables. Let n be a multiple of 3 and let $\mathcal{C}$ be a set of clauses of cardinality 2 or 3. There are n clauses of cardinality 2 and $n/3$ clauses of cardinality 3 so that:

- each clause of cardinality 2 is equal to $(x \vee \bar{y})$ for some $x,\ y \in \mathcal{V}$ with $x \neq y$.
- each of the n literals x (resp. of the literals $\bar{x}$) for $x \in \mathcal{V}$ belongs to one of the n clauses of cardinality 2, thus to only one of them.
- each of the n literals x belongs to one of the $n/3$ clauses of cardinality 3, thus to only one of them.
- whenever $(x \vee \bar{y})$ is a clause of cardinality 2 for some $x,\ y \in \mathcal{V}$, then x and y belong to different clauses of cardinality 3.

Question:
Is there a truth assignment $I : \mathcal{V} \to \{0,1\}$ such that every clause in $\mathcal{C}$ has exactly a true literal?
In order to illustrate Π_1, we consider the following example.
Example The following logic formula is a valid instance of Π_1:
$(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_0 \vee x_3) \wedge (\bar{x}_3 \vee x_0) \wedge (\bar{x}_4 \vee x_2) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_5 \vee x_1) \wedge (\bar{x}_2 \vee x_5)$.
The answer to Π_1 is *yes*. It suffices to choose $x_0 = 1$, $x_3 = 1$ and $x_i = 0$ for $i = \{1,2,4,5\}$. This yields a truth assignment satisfying the formula, and there is exactly one true literal in every clause. For the proof of the $\mathcal{NP}$-completeness see [5].

3 Non-approximability results

In this section, we show in the first part, that the problem denoted by $\bar{P}|prec; c_{ij} = c \geq 3; p_i = 1|C_{max}$ does not possess a polynomial time approximation algorithm with ratio guarantee better than $1 + \frac{1}{c+4}$ for the minimization of the length of the schedule (resp. $1 + \frac{1}{2c+5}$, for the minimization of the sum of the completion time). We also give the $\mathcal{NP}$-completeness for the special case $c = 2$ and $C_{max} = 6$ in the section 3.2.

3.1 The minimization of length of the schedule

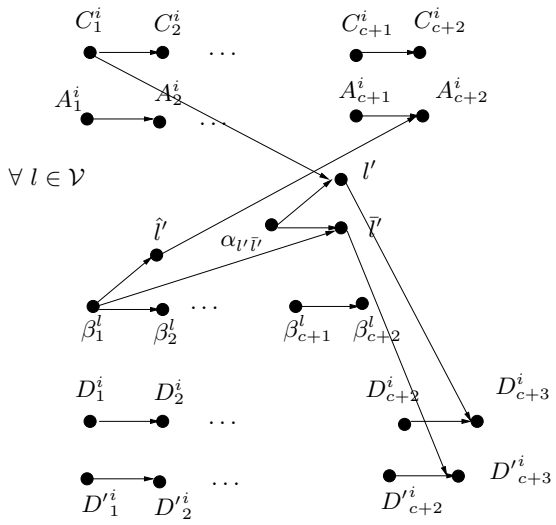


Fig. 1. A partial precedence graph for the $\mathcal{NP}$-completess of the scheduling problem $\bar{P}|prec; c_{ij} = c \geq 3; p_i = 1|C_{max}$.
Remark: $\bar{l}'$ is in the clause of lenght two associated to $D'^i_1 \to D'^i_2 \to \ldots D'^i_{c+2} \to D'^i_{c+3}$

Theorem 1. *The problem of deciding whether an instance of* $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max}$ *has a schedule of length at most* $(c+4)$ *is* $\mathcal{NP}$*-complete with* $c \geq 3$.

Proof. It is easy to see that $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max} = c+4 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_1.

Given an instance π^* of Π_1, we construct an instance π of the problem $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max} = c+4$, in the following way:

1. For all $x \in \mathcal{V}$, we introduce $(c+6)$ variables-tasks: $\alpha_{x'\bar{x}'}$, x', $\bar{x}'$, $\hat{x}'$, β_j^x with $j \in \{1, 2, \ldots, c+2\}$. We add the precedence constraints: $\alpha_{x'\bar{x}'} \rightarrow x'$, $\alpha_{x'\bar{x}'} \rightarrow \bar{x}'$, $\beta_1^x \rightarrow \hat{x}'$, $\beta_1^x \rightarrow \bar{x}'$, $\beta_j^x \rightarrow \beta_{j+1}^x$ with $j \in \{1, 2, \ldots, c+1\}$.
2. For all clauses of length three denoted by $C_i = (y \vee z \vee t)$, we introduce $2 \times (2+c)$ clauses-tasks C_j^i and A_j^i, $j \in \{1, 2, \ldots c+2\}$, with precedence constraints: $C_j^i \rightarrow C_{j+1}^i$ and $A_j^i \rightarrow A_{j+1}^i$, $j \in \{1, 2, \ldots, c+1\}$. We add the constraints $C_1^i \rightarrow l$ with $l \in \{y', z', t'\}$ and $l \rightarrow A_{c+2}^i$ with $l \in \{\hat{y}', \hat{z}', \hat{t}'\}$.
3. For all clauses of length two denoted by $C_i = (x \vee \bar{y})$, we introduce $(c+3)$ clauses-tasks D_j^i, $j \in \{1, 2, \ldots, c+3\}$ with precedence constraints: $D_j^i \rightarrow D_{j+1}^i$ with $j \in \{1, 2, \ldots, c+2\}$ and $l \rightarrow D_{c+3}^i$ with $l \in \{x', \bar{y}'\}$.

The above construction is illustrated in Figure 1. This transformation can be clearly computed in polynomial time.

- Let us first assume that there is a schedule of length at most $(c+4)$. In the following, we will prove that there is a truth assignment $I : \mathcal{V} \rightarrow \{0, 1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal.
  First we can remark that if $c \geq 3$ then $2c+2 > c+4$ and so, each path A_j^i, β_j^x, C_j^i or $D_{j'}^i$ with $j \in \{1, 2, \ldots, c+2\}$ and $j' \in \{1, 2, \ldots, c+3\}$ must be executed on the same processor. What's more, two of these paths cannot be executed on the same processor.
  Notation: In the following we denote by P_A (resp. P_C) the set of the $\frac{n}{3}$ processors which execute a path A_j^i (resp. a path C_j^i). Notice that we know by the definition of the problem Π_1, that in an instance admits $\frac{n}{3}$ clauses of length three where n denotes the number of variables. In the same way, we denote by P_β (resp. P_D) the set of the n processors which execute a path β_j^x (resp. a path D_j^i).

  Lemma 1. *If* $C_{max} = c+4$ *then assigning the value true to the variable* x *if and only if the variable-task* x' *is executed on a processor of the path* P_C *we obtain a correct solution.*

  Proof. In order to respect the feasible schedule of length $(c+4)$, in the first time, we can stem from the polynomial time transformation, that the starting time of the variables-tasks l', $\bar{l}'$ and $\hat{l}'$, and that the processors on which these tasks must be executed, are given by the following remarks:
  $\forall l \in \mathcal{V}$:
  - Each variable-task l' is executed on a processor of P_C at slot 3 or on a processor of P_D at slot $(c+2)$ or $(c+3)$,
  - Each variable-task $\bar{l}'$ is executed on a processor of P_β at slot 3 or on a processor of P_D at slot $(c+2)$ or $(c+3)$,
  - Each variable-task $\hat{l}'$ is executed on a processor of P_β at slot 3 or on a processor of P_A at slot $(c+2)$ or $(c+3)$,
  - The variables-tasks $\bar{l}'$ and $\hat{l}'$ cannot be executed together on a processor of P_β (they have a common predecessor).

  Notation and property: For each $l \in \mathcal{V}$, we can associate the three tasks l', $\bar{l}'$, $\hat{l}'$. We denote by $X = \{l' | l \in \mathcal{V}\}$, $\bar{X} = \{\bar{l}' | l \in \mathcal{V}\}$ and $\hat{X} = \{\hat{l}' | l \in \mathcal{V}\}$ three sets of tasks. For each subset A of $\bar{X}$ (resp. $\hat{X}$), we can associate a subset B of X in the following way: $l' \in B$ *if and only if* $\bar{l}' \in A$ (resp. $\hat{l}' \in A$).
  Let be the following sets:
  - $X_1 = \{l' \backslash \pi(l') = \pi(P_C)\}$ where $\pi(l')$ (resp. $\pi(P_C)$) designs the processor on which the task l' is scheduled.
  - $X_2 = \{l' \backslash \pi(l') = \pi(P_D)\}$.

- $X_3 = \{l' \backslash \pi(\bar{l}') = \pi(P_\beta)\}$.
- $X_4 = \{l' \backslash \pi(\bar{l}') = \pi(P_D)\}$
- $X_5 = \{l' \backslash \pi(\hat{l}') = \pi(P_\beta)\}$
- $X_6 = \{l' \backslash \pi(\hat{l}') = \pi(P_A)\}$

Let be $x_i = |X_i|$ for $i \in \{1, \ldots, 6\}$.
We can stem from the construction of an instance of the scheduling problem the following table,

	P_C	P_β	P_A	P_D
x'	X_1			X_2
$\bar{x}'$		X_3		X_4
$\hat{x}'$		X_5	X_6	

From the previous table, using the variable x_i, we obtain the following inequations system:

$$x_1 + x_2 = n \tag{1}$$
$$x_3 + x_4 = n \tag{2}$$
$$x_5 + x_6 = n \tag{3}$$
$$x_1 \leq \frac{n}{3} \tag{4}$$
$$x_6 \leq \frac{2n}{3} \tag{5}$$
$$x_3 + x_5 \leq n \tag{6}$$
$$x_2 + x_4 \leq n \tag{7}$$

We will give some details about the previous system:

- For the equations(1), (2) and (3): We must execute all the tasks of the sets X, $\bar{X}$ and $\hat{X}$.
- For the equation (4), on the processor which executes the path C_j^i of the clause $C_i = (y \vee z \vee t)$, we can execute at most one of the three variables-tasks y', z', t'. Indeed, all variables-tasks l' as a successor which is executed on a processor of P_D. If it is executed on the processor which scheduled the tasks from the path P_C it cannot be executed before the slot 3 and so, the variable-task $\alpha_{l'\bar{l}'}$ must be executed on the same processor which becomes saturated. So, we have $|X_3| < |P_C|$.
- For the equation (5), each processor of the paths P_A has two free slots and $|P_A| = \frac{n}{3}$.
- For the equation (6), all the variables-tasks $\bar{l}'$ or $\hat{l}'$ which are executed on a processor of the path P_β must be finished before slot 3 (it has a successor executed on another processor). So the variable-task $\alpha_{l'\bar{l}'}$ must be executed on the same processor which becomes saturated. Therefore, at the most one task between the variables-tasks $\bar{l}'$ and $\hat{l}'$ can be executed on a processor of the path P_β and so, $|X_3| + |X_5| \leq |P_\beta|$.
- For the equation (7), it is clear that, $|P_D| = n$ and there is at the most one free slot on each processor of P_D.

In the one hand, we have $x_3 + x_5 = n$ and in the second hand, $\forall l'$ only one variable-task between the variables-tasks $\bar{l}'$ and $\hat{l}'$ can be executed on a processor of P_β, thus we obtain $X_3 \cap X_5 = \emptyset$. Consequently, we have $X_3 \cup X_5 = X$. As the set X_4 (resp. X_6) is the complementary of the set X_3 (resp. X_5) we have $X_4 \cup X_6 = X$. In more if the variable-task l' is executed on a processor of P_C then the variable-task $\alpha_{l'\bar{l}'}$ is executed on the same processor. The variable-task $\bar{x}'$ cannot be executed before the slot $(c+2)$ therefore on a processor of P_D. We can deduce that $X_1 = X_4$ (the two sets are the same cardinality). Finally, we have $X_1 \cup X_2 = X$, $X_3 \cup X_4 = X$, $X_5 \cup X_6 = X$, $X_4 \cup X_6 = X$, $X_3 \cup X_5 = X$, $X_1 = X_4$ and therefore $X_1 = X_4 = X_5$ and $X_2 = X_3 = X_5$.
We can deduce from the previous equations that $x_1 = x_4 = x_5 = \frac{n}{3}$ and $x_2 = x_3 = x_6 = \frac{2n}{3}$.
This concludes the proof of Lemma 1.

So, if we affect the value "true" to the variable l if and only the variable-task l' is executed on a processor of P_C it is trivial to see that in the clause of length 3 we have one and only one literal equal to "true".
Let be $c = (x \vee \bar{y})$, a clause of length 2.

- If $x' \in X_1 \implies y' \in X_4 \implies y' \in X_1$. The first implication (resp. the second) is due to the fact that each processor of the path P_D must be saturated ($x_2 + x_4 = n$) (resp. $X_1 = X_4$). Only the literal x is "true" between the variables x and $\bar{y}$.
- If $x' \in X_2 \implies y' \in X_3 \implies y' \in X_2$. The first (resp. the second) implication is due to the fact that there is only one free slot on each processor executing the path P_D (resp. $X_3 = X_2$). Only the literal $\bar{y}$ is "true" between the variables x and $\bar{y}$.

In conclusion, there is only one true literal per clause.

- Conversely, we suppose that there is a truth assignment $I : \mathcal{V} \to \{0, 1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal.
Suppose that the true litteral in the clause $C_i = (y \vee z \vee t)$ is t. Therefore, the variable-task t' (resp. y' and z') is processed at the slot 3 (resp. at the slot $(c+2)$) on the same processor as the path P_{C_i} (resp. as the path P_D and $P_{D'}$, where D and D' indicates a clause of length two where the variables y and z occurred). The $\frac{2n}{3}$ other variables-tasks y' not yet scheduled are executed at slot 3 on processor P_β as the variable-task $\alpha_{y'\bar{y}'}$. The variable-task $\hat{t}'$ (resp. $\hat{y}'$ and $\hat{z}'$) is executed at the slot 2 (resp. $c+2$ and $c+3$) on a processor of the path P_β (resp. P_A).
This concludes the proof of Theorem 1.

3.2 The special case $c = 2$

In this part, we will consider the special case; $\forall (i,j) \in E$, if $\pi^i = \pi^j$ then $t_j \geq t_i + p_i$ else $t_j \geq t_i + p_i + 2$. We study the complexity of this problem, and we will prove the $\mathcal{NP}$-completeness of the $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max}$ for C_{max}. Recall that (see [13],[17]), for the classical UET-UCT problem ($\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$) that the problem becomes $\mathcal{NP}$-complete (resp. polynomial) for $C_{max} = 6$ (resp. $C_{max} = 5$).

The idea of the construction of an instance of the scheduling problem $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max}$ is similar as previously, and it is based on the problem Π_1.

Theorem 2. *The problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max}$ has a schedule of length at most six is $\mathcal{NP}$-complete.*

Proof. It is easy to see that $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max} = 6 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_1.

Given an instance π^* of Π_1, we construct an instance π of the problem $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max} = 6$, in the following way:

1. For all $x \in \mathcal{V}$, we introduce 5 variables-tasks: $\alpha_{x'}$, x', $\bar{x}'$, $\hat{x}'$, $\beta_{x'}$. We add the precedence constraints: $\alpha_{x'} \to x'$, $\alpha_{x'} \to \bar{x}'$, $\beta_{x'} \to \hat{x}'$, $\beta_{x'} \to \bar{x}'$.
2. For all clauses of length three denoted by $C_i = (y \vee z \vee t)$, we introduce two clauses-tasks C^i and A^i. We add the following precedence constraints: $C^i \to l$ with $l \in \{y', z', t'\}$ and $l \to A^i$ with $l \in \{\hat{y}', \hat{z}', \hat{t}'\}$.
3. For all clauses of length two denoted by $C_i = (x \vee \bar{y})$, we introduce five clauses-tasks D^i_j, $j \in \{1, 2, \ldots, 5\}$ with precedence constraints: $D^i_j \to D^i_{j+1}$ with $j \in \{1, 2, 3, 4\}$ and $l \to D^i_5$ with $l \in \{x', \bar{y}'\}$.

- Let us first assume that there is a schedule of length at most 6. In the following, we will prove that there is a truth assignment $I : \mathcal{V} \to \{0, 1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal.
First we can remark that if a task is executed at slot 3 (resp. at slot 4) or before (resp. after) all these predecessors (resp successors) are executed on the same processor. Since the communication is allowed on a path of length 5 so that $\forall i$, all the clauses-tasks D^i_j are executed on the same processor.

  Lemma 2. *If $C_{max} = 6$ then if we assign the value true to the variable x if and only if the variable-task x' is executed at slot 3 we obtain a correct solution.*

  Proof. $\forall l \in \mathcal{V}$, by construction, it is clear that:

- The variable-task l' is executed at slot 3 on the processor which executed $\alpha_{x'}$ and C^i where $x \in C_i$ or after slot 4 on the processor which executed D_5^i such that $x \in C_i$.
- The variable-task $\bar{l}'$ is executed at slot 3 on the processor which executed $\alpha_{x'}$ and $\beta_{x'}$ or after slot 4 on the processor which executed D_5^i such that $\bar{x} \in C_i$.
- The variable-task $\hat{l}'$ is executed at slot 2 or 3 on the processor which executed $\beta_{x'}$ or after slot 4 on the processor which executed A^i such that $x \in C_i$.

Using the same notation as previously, we consider

- $X_1 = \{l' \backslash t_{l'} = 3\}$ where t_j designs the starting of the task j.
- $X_2 = \{l' \backslash t_{l'} \geq 4\}$.
- $X_3 = \{l' \backslash t_{\bar{l}'} = 3\}$.
- $X_4 = \{l' \backslash t_{\bar{l}'} \geq 4\}$.
- $X_5 = \{l' \backslash t_{\hat{l}'} = 2 \text{ or } t_{\hat{l}'} = 3\}$.
- $X_6 = \{l' \backslash t_{\hat{l}'} \geq 4\}$.

Let be $x_i = |X_i|$ for $i \in \{1, \ldots, 6\}$.
We can stem from the construction of an instance of the scheduling problem,

- $x_1 \leq \frac{n}{3}$ (there are only $\frac{n}{3}$ clauses-tasks C_i),
- $x_2 + x_4 \leq n$ (there are only n processors which executed a clause-task D_5^i),
- $x_3 + x_5 \leq n$ ($\forall\ x'$, $\hat{x}'$ and $\bar{x}'$ can not be executed together at slot 3 in the same processor (they have a common predecessor)),
- $x_6 \leq \frac{2n}{3}$ (there are at most $\frac{n}{3}$ processors which executed a clause-task A^i and each processor have only the slot 4 and 5 to execute variables-tasks $\hat{x}'$).

So we can deduce from $\sum_{i=1}^{6} x_i \leq 3n$. As $\sum_{i=1}^{6} x_i = 3n$ (all the variables-tasks must be executed), the inequalities are all equalities.
And finally, we have $x_1 = x_4 = x_5 = \frac{n}{3}$ and $x_2 = x_3 = x_6 = \frac{2n}{3}$.
If the variable-task x' (resp. the variable-task $\bar{x}'$) is executed at slot 3, the variable-task $\bar{x}'$ (resp. x') must be executed at or after slot 4. Therefore, $X_1 \subseteq X_4$ and $X_3 \subseteq X_2$ and finally as the cardinalities are equal $X_1 = X_4$ and $X_3 = X_2$.
So if we affect the value "true" to the variable l if and only if the variable-task l' is executed at slot 3, it is trivial to see that in the clause of length 3 we have one and only one literal equal to "true".
Let be $c = (x \vee \bar{y})$, a clause of length 2. As in the proof of the theorem 1, we obtain:

- If $x' \in X_1 \implies y' \in X_4 \implies y' \in X_1$. Only the literal x is "true" between x and $\bar{y}$,
- If $x' \in X_2 \implies y' \in X_3 \implies y' \in X_2$. Only the literal $\bar{y}$ is "true" between x and $\bar{y}$

In conclusion, there is only one true literal per clause.

- Conversely, we suppose that there is a truth assignment $I : \mathcal{V} \to \{0, 1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal.
  Suppose that the true litteral in the clause $C_i = (y \vee z \vee t)$ is t. Therefore, the variable-task t' is processed at the slot 3 on the same processor which executed the clause-task C^i and the task α'_t. The task $\bar{t}'$ (resp. y', z') are executed on the processor which executed the clause-task D_5^i corresponding to the clauses of length 2 where $\bar{t}$ appears (resp. y, z). The clause-task D_j^i, $j = 1, 2, 3, 4$, is executed at slot j on the same processor. The $\frac{2n}{3}$ other variables-tasks $\bar{y}'$ not yet scheduled are executed at slot 3 in the processor which executed $\alpha_{y'}$ and $\beta_{y'}$. The variables-tasks $\hat{t}'$ (resp. $\hat{y}'$ and $\hat{z}'$) are executed at slot 2 (resp. 4 and 5) on the processor executing β'_t (resp. A^i). We can observe that this scheduling is valid.
  This concludes the proof of theorem 2.

We can deduice from the two previous theorems, the classical following Corollary:

Corollary 1. *There is no polynomial-time algorithm for the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ with performance bound smaller than $1 + \frac{1}{c+4}$ unless $\mathcal{P} \neq \mathcal{NP}$.*

Proof. The proof of Corollary 1 is an immediate consequence of the Impossibility Theorem, (see [3], [4]).

3.3 The minimization of the sum of the completion time

In this section, we will show that there is no polynomial-time algorithm for the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|\sum_j C_j$ with performance bound smaller than $1 + \frac{1}{2c+5}$ unless $\mathcal{P} \neq \mathcal{NP}$. This result is obtained by the polynomial transformation used for the proof of the Theorem 1 and the gap technic (see [7]).

Theorem 3. *There is no polynomial-time algorithm for the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|\sum_j C_j$ with performance bound smaller than $1 + \frac{1}{2c+5}$ unless $\mathcal{P} \neq \mathcal{NP}$.*

Proof. We suppose that there is a polynomial time approximation algorithm denoted by A with performance guarantee bound smaller than $1 + \frac{1}{2c+5}$,

Let be I the instance of the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ obtained by a reduction (see Theorem 1). Let be I' the instance of the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|\sum_j C_j$ by adding x new tasks from an initial instance I. In the precedence constraints, each x (with $x > \frac{(2c+6)c+(c+4)\rho n}{2c+6)-(2c+5)\rho}$) new tasks is a successor of the old tasks (old tasks are from the polynomial transformation used for the proof of Theorem 1). We obtain a complete directed graph from old tasks to new tasks.

If there exists such an algorithm A, then it can be used to decide the existence of an independent set of size n.

Let $A(I')$ (resp. $A^*(I')$) the result given by A (resp. an optimal result) on an instance I'.

1. If $A(I') < (2c+5)\rho x + (c+4)\rho n$ then $A^*(I') < (2c+5)\rho x + (c+4)\rho n$. So we can decide that there exists a scheduling of an instance I with $C_{max} \leq c+4$. Indeed, if there no exists a such schedule. So, for every schedule a task of an instance I is executed after the time $c+5$. But, perhaps c tasks are executed after $2c+6$ and so $A^*(I') > (2c+6)(x-c)$. It is impossible, indeed if $(2c+6)(x-c) \geq (2c+5)\rho x + (c+4)\rho n$ then $x \leq \frac{(2c+6)c+(4+c)\rho n}{2c+6)-(2c+5)\rho}$. It is a contradiction with the hypothesis.
   A contradiction with $x > \frac{(2c+6)c+(4+c)\rho n}{2c+6)-(2c+5)\rho}$
   Thus, it exists a schedule of length $c+4$ on an old task.
2. We suppose that $A(I') \geq (2c+5)\rho x + (c+4)\rho n$. So, $A^*(I') \geq (2c+5)x + (c+4)n$ because an algorithm A is a polynomial time approximation algorithm with performance guarantee bound smaller than $\rho < \frac{2c+6}{2c+5}$. There is no algorithm to decide whether the tasks from an instance I admit a schedule of length of at the most $c+4$.
   Indeed, if there exists such an algorithm, by executing the x tasks at time $t = 4+2c$, we obtain a schedule with a completion time stricly less than $(5+2c)x + (4+c)n$ (there is at least one task which is executed before the time $t = c+4$). It is a contradiction since $A^*(I') \geq (2c+5)x + (c+4)n$.

4 A polynomial time for $C_{max} = c+2$ with $c \in \{2,3\}$

Theorem 4. *The problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max}$ with $c \in \{2,3\}$ has a schedule of length at most $(c+2)$ is solvable in polynomial time.*

Proof. For $c = 2$. It easy to see that the source (resp. the sink) is executed at the slot 0 (resp. the slot $c+2$). If a source i (resp. a sink i) is scheduled before or at the slot 2 (resp. before or at the slot $c+1$) then the task i admits only one successor (resp. only one predecessor). The other tasks must be executed as soon as possible.

5 Approximation by expansion

5.1 Introduction and notation

In this section, we develop a new method based on a notion of expansion of a schedule. Let us first give some notation before an explanation of the method.

Notation: We denote by σ^{∞}, the UET-UCT schedule, and by σ_c^{∞} the UET-LCT schedule. Moreover, we denote by t_i (resp. t_i^c) the starting time of the task i in the schedule σ^{∞} (resp. in the schedule σ_c^{∞}).

Principle:

An idea consisting in keeping an assignment for the tasks given by a "good" feasible schedule on an unbounded number of processors σ^{∞}, and proceed to an expansion of the makespan, in order to preserve temporal distance between two tasks, i and j with $(i,j) \in E$, processing on two different processors respect the communications delays *i.e.* $t_j^c \geq t_i^c + 1 + c$.

5.2 Description of the method

Let be a precedence graph $G = (V, E)$, we determinate a feasible schedule σ^{∞}, for the model UET-UCT, using an $(4/3)-$approximation algorithm proposed by Munier and König [12]. This algorithm gives a couple $\forall i \in V,\ (t_i, \pi)$ on the schedule σ^{∞} corresponding to:

- t_i the starting time of the task i for the schedule σ^{∞} and
- π the processor on which the task i is processed at t_i.

Now, we determinate a couple $\forall i \in V,\ (t_i^c, \pi')$ on the schedule σ_c^{∞} in the following ways:

- The starting time $t_i^c = d \times t - i = \frac{(c+1)}{2} t_i$ and,
- $\pi = \pi'$.

The justification of the expansion coefficient is given below. An illustration of the expansion is given by Figure 2.

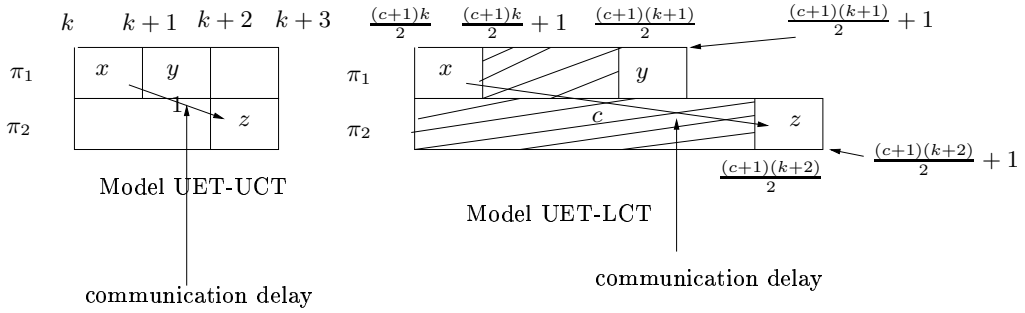


Fig. 2. Illustration of notion of an expansion

5.3 Analysis of the method

Lemma 3. *The coefficient of an expansion is* $d = \frac{(c+1)}{2}$.

Proof. Let be two tasks i and j such that $(i,j) \in E$, which are processed on two differents processors in the feasible schedule σ^{∞}. We are interested in having a coefficient d such that $t_i^c = d \times t_i$ and $t_j^c = d \times t_j$. After an expansion, in order to respect the precedence constraints and the communication delays we must have $t_j^c \geq t_i^c + 1 + c$, and so

$$d \times t_i - d \times t_j \geq c + 1$$
$$d \geq \frac{c+1}{t_i - t_j}$$
$$d \geq \frac{c+1}{2}$$

It is sufficient to choose $d = \frac{(c+1)}{2}$.

Lemma 4. *An expansion algorithm gives a feasible schedule for the problem denoted by* $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{\max}$.

Proof. It sufficient to check that the solution given by an expansion algorithm produces a feasible schedule for the model UET-LCT. Let be two tasks i and j such that $(i,j) \in E$. We denote by π_i (resp. π_j) the processor on which the task i (resp. the task j) is executed in the schedule σ^∞. Moreover, we denote by π'_i (resp. π'_j) the processor on which the task i (resp. the task j) is executed in the schedule σ_c^∞. Thus

- If $\pi_i = \pi_j$ then $\pi'_i = \pi'_j$. Since the solution given by Munier and König [12] gives a feasible schedule on the model UET-UCT, then we have

$$\begin{aligned} t_i + 1 &\leq t_j \\ \frac{2}{c+1}t_i^c + 1 &\leq \frac{2}{c+1}t_j^c \\ t_i^c + 1 \leq t_i^c + \frac{c+1}{2} &\leq t_j^c \end{aligned}$$

- If $\pi_i \neq \pi_j$ then $\pi'_i \neq \pi'_j$. We have

$$\begin{aligned} t_i + 1 + 1 &\leq t_j \\ \frac{2}{c+1}t_i^c + 2 &\leq \frac{2}{c+1}t_j^c \\ t_i^c + (c+1) &\leq t_j^c \end{aligned}$$

Theorem 5. *An expansion algorithm gives a* $\frac{2(c+1)}{3}-$*approximation algorithm for the problem* $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{\max}$.

Proof. We denote by C_{max}^h (resp. C_{max}^{opt}) the makespan of the schedule computed by the Munier and König (resp. the optimal value of a schedule σ^∞).

In the same way we denote by $C_{max}^{h^*}$ (resp. $C_{max}^{opt,c}$) the makespan of the schedule computed by our algorithm (resp. the optimal value of a schedule σ_c^∞).

We know that $C_{max}^h \leq \frac{4}{3}C_{max}^{opt}$. Thus, we obtain

$$\frac{C_{max}^{h^*}}{C_{max}^{opt,c}} = \frac{\frac{(c+1)}{2}C_{max}^h}{C_{max}^{opt,c}} \leq \frac{\frac{(c+1)}{2}C_{max}^h}{C_{max}^{opt}} \leq \frac{\frac{(c+1)}{2}\frac{4}{3}C_{max}^{opt}}{C_{max}^{opt}} \leq \frac{2(c+1)}{3}$$

Remark: The method of an expansion can be used for another problems.

6 Analysis of our results

With the result proposed in this article, we complete the table 2. We proved that in the first case, as we know, the lower bound of approximation is the same for the problems with a bounded and an unbounbed number of processors in presence of large communications delays.

The new complexity results are given by Table 3.

[1] $3SAT$ see[4]
[2] $Clique$ see[4]
[3] Π_1 see Therorem 1
[4] $BBCG$ see [4, 16]

Type of machines	c_{ij}	C_{max}	Complexity	Lower bound	References
$\bar{P}$	$c=1$	5	Polynomial		[17]
$\bar{P}$	$c=1$	6	$\mathcal{NP}$-complete [1]	$7/6 \leq \rho$	[17]
P	$c=1$	3	Polynomial		[13]
P	$c=1$	4	$\mathcal{NP}$-complete[2]	$5/4 \leq \rho$	[17]
$\bar{P}$	$c \in \{2,3\}$	$c+2$	Polynomial		Theorem 4
$\bar{P}$	$c \geq 2$	$c+4$	$\mathcal{NP}$-complete[3]	$1+1/(c+4) \leq \rho$	Theorem 1
P	$c \geq 2$	$c+1$	Polynomial		[1]
P	$c \geq 2$	$c+3$	$\mathcal{NP}$-complete[4]	$1+1/(c+3) \leq \rho$	[1]

Table 3. Complexity results

Type of machines	c_{ij}	Approximation results	References
$\bar{P}$	$c=1$	$\rho \leq 4/3$	[11]
P	$c=1$	$\rho \leq 7/3$	[10]
$\bar{P}$	$c \geq 2$	2 for tree	[9]
$\bar{P}$	$c \geq 1$	$\frac{2(c+1)}{3}$	Theorem 5
P	$c \geq 2$	$O(c)$	[14]

Table 4. New approximation results

7 Conclusion

In this paper, we first proved that the problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ has a schedule of length at most $(c+4)$ is $\mathcal{NP}$-complete.

This result is to be compared with the result of [8] and [1], which states that $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max} = 6$ (resp. $P|prec; c_{ij} = c \geq 3; p_i = 1|C_{max} = c+3$) is $\mathcal{NP}$-complete. Our result implies that there is no $\rho-$approximation algorithm with $\rho < 1 + \frac{1}{c+4}$, unless $\mathcal{P} = \mathcal{NP}$. In addition, we show that there is no hope of finding a ρ-approximation algorithm with ρ strictly less than $\rho < 1 + \frac{1}{2c+5}$ for the problem of the minimization of the sum of the completion time.

Secondly, we established that the problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max}$ with $c \in \{2,3\}$ has a schedule of length at most $(c+2)$ is solvable in polynomial time.

We also propose a $\frac{2(c+1)}{3}-$approximation algorithm based on the notion of dilatation. **Remark:** We conjecture that the problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max}$ with $c \geq 2$ has a schedule of length at most $(c+3)$ is solvable in polynomial time.

References

1. E. Bampis, A. Giannakos, and J.C. König. On the complexity of scheduling with large communication delays. *European Journal of Operation Research*, 94:252–260, 1996.
2. B. Chen, C.N. Potts, and G.J. Woeginger. A review of machine scheduling: complexity, algorithms and approximability. Technical Report Woe-29, TU Graz, 1998.
3. P. Chrétienne and C. Picouleau. *Scheduling Theory and its Applications*. John Wiley & Sons, 1995. Scheduling with Communication Delays: A Survey.
4. M.R. Garey and D.S. Johnson. *Computers and Intractability, a Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, 1979.
5. R. Giroudeau. *L'impact des délais de communications hiérarchiques sur la complexité et l'approximation des problèmes d'ordonnancement*. PhD thesis, Université d'Évry Val d'Essonne, 2000.

6. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministics sequencing and scheduling theory: a survey. *Ann. Discrete Math.*, 5:287–326, 1979.
7. H. Hoogeveen, P. Schuurman, and G.J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In R.E. Bixby, E.A. Boyd, and R.Z. Ríos-Mercado, editors, *IPCO VI*, Lecture Notes in Computer Science, No. 1412, pages 353–366. Springer-Verlag, 1998.
8. J.A. Hoogeveen, J.K. Lenstra, and B. Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *O. R. Lett.*, 16(3):129–137, 1994.
9. A. Munier. Approximation algorithms for scheduling trees with general communication delays. *Parallel Computing*, 25(1):41–48, January 1999.
10. A. Munier and C. Hanen. An approximation algorithm for scheduling dependent tasks on m processors with small communication delays. In *IEEE Symposium on Emerging Technologies and Factory Automation*, Paris, 1995.
11. A. Munier and C. Hanen. An approximation algorithm for scheduling unitary tasks on m processors with communication delays. Non publié, 1996.
12. A. Munier and J.C. König. A heuristic for a scheduling problem with communication delays. *Operations Research*, 45(1):145–148, 1997.
13. C. Picouleau. New complexity results on scheduling with small communication delays. *Discrete Applied Mathematics*, 60:331–342, 1995.
14. C. Rapine. *Algorithmes d'approximation garantie pour l'ordonnancement de tâches, Application au domaine du calcul parallèle*. PhD thesis, Institut National Polytechnique de Grenoble, 1999.
15. V.J. Rayward-Smith. UET scheduling with unit interprocessor communication delays. *Discr. App. Math.*, 18:55–71, 1987.
16. R. Saad. Scheduling with communication delays. *JCMCC*, 18:214–224, 1995.
17. B. Veltman. *Multiprocessor scheduling with communications delays*. PhD thesis, CWI-Amsterdam, Holland, 1993.