General Non-Approximability Results in Presence of Hierarchical Communications

Rodolphe Giroudeau, Jean-Claude König

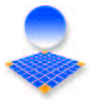
Rapport de Recherche

No : 03-019

Mars 2004

General non-approximability results in presence of hierarchical communications

R. Giroudeau and J.C. Konig, konig,rgirou@lirmm.fr



Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
161 rue Ada, 34392 Montpellier cedex 5
tel : (+33)4 67 41 85 85 – fax : (+33)4 67 41 85 00

General non-approximability results in presence of hierarchical communications

Mars 2004

Abstract

We investigate the problem of minimizing the makespan (resp. the sum of the completion time) for the multiprocessor scheduling problem in presence of hierarchical communications. We show that there is no hope to find a ρ-approximation with $\rho < 1 + \frac{1}{c+3}$ (resp. $1 + \frac{1}{2c+4}$) (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 with $l \geq 4$ identical processors each, and where c denotes the communication delay between two tasks i and j submitted to a precedence constraints and to be processed by two differents machines. We also prove that the problem becomes polynomial whenever the makespan is at most $(c+1)$.es

Keywords: scheduling, hierarchical communications, non-approximability

Résumé

Mots-clés : ordonnancement, communications hiérarchiques, non-approximabilité

1 Introduction

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

Models used by the scheduling theory keep track of technology evolution:

- In the PRAM's model, in which communications are considered as instantaneous, the critical path gives the makespan of the schedule.
- In the `homogeneous scheduling delay model`, each arc (i, j) modelizes the potential data transfert between the task i and the task j provided that i and j are processed on two different processors.

With the increasing importance of parallel computing, the question of how to schedule a set of tasks on an architecture becomes critical, and has received much attention. More precisely, scheduling problems involving precedence constraints are among the most difficult problems in the area of machine scheduling and are most studied problems.

In this paper, we adopt the *hierarchical communication model* [3] in which we assume that the communication delays are not homogeneous anymore; the processors are connected in *clusters* and the communications inside the same cluster are much faster than those between processors belonging to different ones. This model captures the hierarchical nature of the communications using today parallel computers, as shown by many PCs or workstations networks (NOWs) [18, 1]. The use of networks (clusters) of workstations as a parallel computer [18, 1] has renewed the interest of the users in the domain of parallelism, but also pointed out new challenging problems related to the exploitation of the potential computation power offered by such a system.

Most of the attempts to modelize these systems were in the form of programming systems rather than abstract models [20, 21, 7, 6]. Only recently, some attempts concerning this issue appeared in the literature [3, 8]. As state before, the one that we adopt here is the *hierarchical communication model* which is devoted to one of the major problems appearing in the attempt of efficiently using such architectures, the *task scheduling problem*. The proposed model includes one of the basic architectural features of NOWs: the hierarchical communication assumption i.e. a level-based hierarchy of the communication delays with successively higher latencies. More formally, given a set of clusters of identical processors, and a precedence graph, we consider that if two communicating tasks are executed on the same processor (resp. on different processors of the same cluster) then the corresponding communication delay is neglected (resp. is equal to what we call *interprocessor communication delay*). On the contrary, if these tasks are executed on different clusters, then the communication delay is more important and it is called the *intercluster communication delay*.

We are given m multiprocessors machines (or clusters) that are used to process n precedence constrained tasks. Each machine (cluster) comprises several identical parallel processors. A couple (c_{ij}, ϵ_{ij}) of communication delays is associated to each arc (i, j) between two tasks in the precedence graph. In what follows, c_{ij} (resp. ϵ_{ij}) is called intercluster (resp. interprocessor) communication, and we consider that $c_{ij} \geq \epsilon_{ij}$. If tasks i and j are executed on different machines Π and Π', then j must be processed at least c_{ij} time units after the completion of i. Similarly, if i and j are executed on the same machine Π but on different processors π and π' then the processing of j can only

start ϵ_{ij} units of time after the completion of i. However, if i and j are executed on the same processor then j can start immediately after the end of i. The communication overhead (intercluster or interprocessor delay) does not interfere with the availability of the processors and any processor may execute any task. Our goal is to find a feasible schedule of the tasks minimizing the *makespan*, i.e. the time needed to execute all the tasks subject to the precedence graph.

Formally, in the `hiearchical scheduling delay model` we associate a couple of value (c_{ij}, ϵ_{ij}) with $\epsilon_{ij} \leq c_{ij}$ $\forall (i,j) \in E$ such that :

- if $\Pi^i = \Pi^j$ and if $\pi^i_k = \pi^j_k$ then $t_i + p_i \leq t_j$
- else if $\Pi^i = \Pi^j$ and if $\pi^i_k = \pi^j_{k'}$ with $k \neq k'$ then $t_i + p_i + \epsilon_{ij} \leq t_j$
- else $\Pi^i \neq \Pi^j$ $t_i + p_i + c_{ij} \leq t_j$

where t_i denotes the processing time of the task i and p_i its duration. The objective, is to find a schedule, *i.e.* an allocation of each task to a time interval on one processor, such that the communication delays are taken into account and the completion time (makespan) is minimized (the makespan is denoted by C_{max} and it corresponds to $\max_{i \in V}\{t_i + p_i\}$).

Notice that the hierarchical model that we consider here is a generalization of the classical scheduling model with communication delays ([9], [11]). Consider for instance that for every arc (i,j) of the precedence graph we have $c_{ij} = \epsilon_{ij}$. In that case the hierarchical model is exactly the classical scheduling communication delays model.

Complexity results: On the negative side, Bampis et al. in [5] studied the impact of the hierarchical communications on the complexity of the associated problem. They considered the simplest case, i.e. the problem $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{max}$, and they showed that this problem does not possess a polynomial time approximation algorithm with ratio guarantee better than 5/4 (unless $\mathcal{P} = \mathcal{NP}$). Recently [13] Giroudeau proved that there no hope to find a ρ-approximation with $\rho < 6/5$ for the couple of communication delays $(c_{ij}, \epsilon_{ij}) = (2,1)$. If duplication is allowed, Bampis et al. [4] extended the result of [10] in the case of hierarchical communications providing an optimal algorithm for $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1; dup|C_{max}$.

Approximation results: On the positive side, the authors presented in [3] a 8/5-approximation algorithm for the problem $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{max}$ which is based on an integer linear programming formulation. They relax the integrity constraints and they produce a feasible schedule by rounding. This result is extended to the problem $\bar{P}(Pl)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{max}$ leading to a $\frac{4l}{2l+1}$-approximation algorithm.

The challenge is to determinate a thresold for approximation algorithm for the two more general problems: $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c,1); p_i = 1|C_{max}$ and $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c,c'); p_i = 1|C_{max}$ with $c' < c$.

Remark: Notice that concerning the problem denoted by $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (2,1); p_i = 1|C_{max}$, in the case of $C_{max} = 5$ (resp. $C_{max} = 3$) the problem is $\mathcal{NP}$-complete (resp. polynomial). For $C_{max} = 4$ we conjecture that there exist a polynomial time algorithm see [13].

We study in this article, the impact of introducing the notion of hierarchical communications on the hardness of approximating the multiprocessors scheduling problem such that the processors of the parallel architecture are partitioned into clusters (we study the case where there are only $l \geq 4$ processors per cluster). The communication delays between the l processors in the same cluster denoted by ϵ_{ij} is equal to one (resp. c') unit(s) of time

whereas the communication delays between two processors in a different cluster denoted by c_{ij} is equal to c units of time. Our problem can be denoted as $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, 1); p_i = 1|C_{max}$ (resp. $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$).

In order to give the thresold for the two problems described below, we prove that the problem of deciding whether an instance of $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, 1); p_i = 1|C_{max}$ with $c \geq 3$ (resp. $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ with $c > c' + 1,\ c' > 1$) has a schedule of length at most $c + 3$ is $\mathcal{NP}$-complete (resp. $c + 3$). We also extend the non-approximability result in the case of the completion time, denoted in what follows by $\sum_j C_j$. In order to obtain this result, we use the polynomial time transformation using to $\mathcal{NP}$-completess proof for the minimization of the makespan, and the gap technic proposed by Hoogeveen et al. [15].

We also prove that the problem of deciding whether an instance of $\bar{P}(Pl)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max})$ has a schedule of length at most $(c + 1)$ is polynomial.

This article is organized as follows: in the next section, we prove that the problem of deciding whether an instance of $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, 1); p_i = 1|C_{max}$ (resp. $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$) has a schedule of length at most $(c + 3)$ is $\mathcal{NP}$-complete. We extend this result to the criteria is the minimization of the completion time by proving that there is no hope to find ρ-approximation algorithm with ρ strictly less than $1 + \frac{1}{2c+4}$. In an Appendix, we give some preliminaries results concerning the problem which be used to the polynomial transformation in order to prove the non-approximability results and we show that the problem of deciding whether an instance of $\bar{P}(Pl|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ has a feasible schedule of length at most $(c + 1)$ is solvable in polynomial time.

2 The non-approximability results

In the first part of this section we study the makespan length minimizing problem for $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max}$ and $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ with $c > c' + 1 \geq 1$. In the second part we change the makespan length C_{max} to the sum of completion times $\sum_j C_j$ where $C_j = t_j + 1$.

2.1 The minimization of the length of the makespan

2.1.1 The $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max}$ problem with $c \geq 3$

Theorem 2.1 *The problem of deciding whether an instance of $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max}$ has a schedule of length at most $c + 3$ is $\mathcal{NP}$-complete.*

Proof

It is easy to see that $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max} = c + 3 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_2 (The $\mathcal{NP}$-completeness of the problem Π_2 is given in the section 4.1 in an Appendix).

Given an instance π^* of Π_2, we construct an instance π of the problem $\bar{P}(Pl \geq 4)|prec; (c_{ij}; \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max} = c + 3$, in the following way:

1. For each variable $x \in \mathcal{V}$ we introduce three variables-tasks x, $\bar{x}$ and $\hat{x}$ with precedence constraints: $\hat{x} \rightarrow x$ and $\hat{x} \rightarrow \bar{x}$.

2. For each clause $C_i = (x \vee \bar{y})$ of size two, we introduce one clause-task D_i (these tasks are denoted in what follows clauses-tasks of type D). We add the precedence constraint $x \to D_i$ and $\bar{y} \to D_i$.

3. For each clause of size three $C_i = (y \vee z \vee t)$,

   (a) we introduce $(2 \times (c-2) + 3)$ clauses-tasks yz, yt, zt, C_i^k and $\overline{yzt}_k$ with $k \in \{1, \ldots, c-2\}$. For every literal l occurring in C_i, we add the precedence constraint:
   $l \to C_i^1$, $y \to yz$, $y \to yt$, $z \to zt$, $z \to yz$, $t \to yt$, $t \to zt$, and

   $$C_i^k \to C_i^{k+1}, \text{ and } \overline{yzt}_k \to \overline{yzt}_{k+1}, \ k \in \{1, \ldots, c-3\}$$

   (b) We add $((c+3) \times (l-3) + (c-2))$ dummy tasks denoted $d_i^{k,j}$, $\forall j \in \{1, \ldots, l-3\}$, $k \in \{1, \ldots, c+3\}$ and l_i^m with $m \in \{1, \ldots, c-2\}$ together with constraints:
   - $d_i^{k,j} \to d_i^{k+1,j}$, $\forall j \in \{1, \ldots, l-3\}$, $k \in \{1, \ldots, c+2\}$ and $l_i^m \to l_i^{m+1}$, $m \in \{1, \ldots, c-3\}$.
   - We also add, $d_i^{3,j} \to C_i^1$, $l \to d_i^{5,j}$, $\bar{l} \to d_i^{5,j}$, $d_i^{3,j} \to l_i^1$, $\hat{l} \to d_i^{3,j}$, $\forall j \in \{1, \ldots, l-3\}$ where l design a literal occuring in the clause C_i.
   - At the final, for all literal l occuring in the clause C_i we add the following precedence constraints: $\bar{l} \to \overline{yzt}_1 \to d_i^{6,j}$, $\forall j \in \{1, \ldots, l-3\}$.

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 $(c+3)$. 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.

  In this case, we can remark :

  1. All the tasks of a path of length four (the length of a path is the number of tasks in the path) are executed in the same cluster. Let be $C_i = (y \vee z \vee t)$ a clause. So, in the same cluster K_i we must execute the tasks C_i^k, $d_i^{k,j}$, $\overline{yzt}_k$, y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$, $\hat{y}$, $\hat{z}$, $\hat{t}$, and l_i^j.
  2. The dummy tasks $d_i^{k,j}$ use $(l-3)$ processors of K_i without free time (a communication is not allowed on a path of length $(c+3)$).
  3. The arcs from the tasks y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$ to the tasks $d_i^{5,j}$ imply that the tasks y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$, $\hat{y}$, $\hat{z}$, and $\hat{t}$ are executed at the slots 1, 2 and 3 on the three others processors.
  4. The precedence constraint between the tasks $\overline{ytz}_1$ and $d_i^{6,j}$ implies that the task $\overline{ytz}_1$ is executed at slot 4 and therefore two tasks among $\bar{y}$, $\bar{z}$, $\bar{t}$ are processed at slot 2.
  5. At the slot 2, one task among y, z, t must be executed. Otherwise the tasks C_i^k, l_i^k, yz, yt, zt, $\overline{ytz}_j$, $j > 1$ and the three clauses-tasks of type D admitting the tasks y, z, t as predecessors, must be executed during the slots 5 to (c+3) on the three free processors of K_i. This is not possible because at most $(3c-3)$ tasks can be executed on three processors during $(c-1)$ slots.

Suppose that the tasks $\bar{y}$, $\bar{z}$ and t are executed at slot 2. The clauses-tasks of type D admitting the tasks y, z and $\bar{t}$ as predecessors are executed at slot $(c+3)$ on the same cluster because they can not be executed on an other cluster (since the intercluster communication delay is c) and the tasks corresponding to the others literals are scheduled at slot 2 on an other cluster.

Now we affect the **value true** to the literal if an associated task is executed **at slot 2** and **false otherwise**. This gives a solution at our problem.

- 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 schedule, in the Figure 1 is feasible on the three free processors.

This concludes the proof of Theorem 2.1.

□

In the following, we proof the $\mathcal{NP}$-completeness of the special couple of communication delays $(c_{ij}, \epsilon_{ij}) = (3,2)$ (this case will be generalized in the Theorem 2.4).

Theorem 2.2 *The problem of deciding whether an instance of* $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (3,2); p_i = 1|C_{max}$ *has a schedule of length at most* 6 *is* $\mathcal{NP}$*-complete.*

Proof It is easy to see that $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (3,2); p_i = 1|C_{max} = 6 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_2.

Given an instance π^* of Π_2, we construct an instance π of the problem $\bar{P}(Pl \geq 4)|prec; (c_{ij}; \epsilon_{ij}) = (3,2); p_i = 1|C_{max} = 6$, in the following way:

1. For each variable $x \in \mathcal{V}$ we introduce three variables-tasks x, $\bar{x}$ and $\hat{x}$ with precedence constraints: $\hat{x} \to x$ and $\hat{x} \to \bar{x}$.

2. For each clause $C_i = (x \vee \bar{y})$ of size we introduce one clause-task D_i. For every literal l occurring in C_i, we add the precedence constraint $l \to D_i$.

3. For each clause of size three $C_i = (y \vee z \vee t)$, we introduce two clauses-tasks C_i and $\overline{yzt}$. For every literal l occurring in C_i, we add the precedence constraint $l \to C_i$ and $\bar{l} \to \overline{yzt}$.

   We add $(6 \times (l-3)+1)$ dummy tasks denoted $d_i^{k,j}$, $\forall j \in \{1, \ldots, l-3\}$, $k \in \{1, \ldots, 6\}$ and l_i together with constraints:

   (a) $d_i^{1,j} \to d_i^{2,j} \to \ldots \to d_i^{6,j}$.
   (b) $l_i \to \overline{yzt}$.
   (c) For every literal l occurring in C_i, $\hat{l} \to d_i^{4,j}$ and $\hat{l} \to l_i$.

Sketch of the proof

- Let us first assume that there is a schedule of length at most $(c+3)$. 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.

  In the same way as prevouisly, the tasks from a path of length four must be executed on the same cluster, thus the tasks $d_i^{k,j}$ and $\hat{l}$ for every literal l occurring in the clause

C_i of size three, must be executed on the same cluster K_i. Moreover l (resp. $\bar{l}$) is executed on K_i (otherwise it is executed at slot 5 or 6 and it has two successors).

Among the six tasks l and $\bar{l}$, only three can be executed at slot 2. Thus three are executed at slot 3 or after and the three clauses-tasks of type D successors of these literals must be executed on K_i at slot 6 (the second literal must be executed at slot 2 on other clusters).

The three tasks l (resp. $\bar{l}$) can not be executed at slot 2, otherwise the task $\overline{yzt}$ (resp. C_i) can not be processed at slot 5 on K_i or at slot 6 on an other cluster. Thus, the tasks $\overline{yzt}$ and C_i are executed on K_i at slot 5.

Since the task $\overline{yzt}$ admit four predecessors $\hat{l}$ (for $l = y,\ z$, or t) and l_i. Two of them must be executed at slot 2 (among the three literals $\bar{l}$) and the two others (the last $\bar{l}$ and l_i) on the same processor at slot 3 and 4. As the task C_i has three precedessors l and only one can be executed at slot 2, the two others must be executed on the same processor at slot 3 and 4.

If we affect the **value true** on the literals corresponding to the tasks executed **at slot 2** and **false** to the other literals, we have a solution to our problem.

- 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. It is easy to decide a schedule of length 6.

□

2.1.2 The $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ problem with $c > c' + 1 > 2$

In this section, we generalize the result given by the Theorem 2.1.

Theorem 2.3 *The problem of deciding whether an instance of $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$, with $c > c' + 1 > 2$, has a schedule of length at most $(c + 3)$ is $\mathcal{NP}$-complete.*

Proof

It is easy to see that $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max} = c + 3 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_2.

Given an instance π^* of Π_2, we construct an instance π of the problem $\bar{P}(Pl \geq 4)|prec; (c_{ij}; \epsilon_{ij}) = (c, c'); p_i = 1|C_{max} = c + 3$, in the following way:

1. For each variable $x \in \mathcal{V}$ we introduce three variables-tasks x, $\bar{x}$ and $\hat{x}$ with precedence constraints: $\hat{x} \to x$ and $\hat{x} \to \bar{x}$.

2. For each clause $C_i = (x \vee \bar{y})$ of size two, we introduce one clause-task D_i. For every literal l occurring in C_i, we add the precedence constraint $l \to D_i$.

3. For each clause of size three $C_i = (y \vee z \vee t)$,

   (a) we introduce $((3 \times c') + 3 \times (c - c' - 1))$ clauses-tasks C_i^k, $\overline{yzt}_k$, l_k^i, yz_j, zt_j and yt_j with $k \in \{1, \ldots, c - c' - 1\}$ and $j \in \{1, \ldots, c'\}$.

- For every literal l occurring in C_i, we add the precedence constraint $l \rightarrow C_i^1$, and $y \rightarrow yz_{c'},\ y \rightarrow yt_{c'},\ z \rightarrow zt_{c'},\ z \rightarrow yz_{c'},\ t \rightarrow yt_{c'},\ t \rightarrow zt_{c'}$. We add $w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_{c'}$ where w is a generic task ($yt,\ yz$ or zt).
- We add the following precedence:

$$C_i^k \rightarrow C_i^{k+1},\ l_i^k \rightarrow l_i^{k+1},\ \overline{yzt}_k \rightarrow \overline{yzt}_{k+1},\ k \in \{1, \ldots, c - c' - 2\}.$$

(b) We add $((c+3) \times (l-3) + (c - c' - 1))$ dummy tasks denoted $d_i^{k,j},\ \forall j \in \{1, \ldots, l-3\},\ k \in \{1, \ldots, c+3\}$ and $l_i^k, \forall k \in \{1, \ldots, c - c' - 1\}$ together with constraints:

$$d_i^{k,j} \rightarrow d_i^{k+1,j},\ \forall j \in \{1, \ldots, l-3\},\ k \in \{1, \ldots, c+2\}.$$

- We also add $d_i^{3,j} \rightarrow C_i^1,\ l \rightarrow d_i^{c'+4,j},\ \bar{l} \rightarrow \overline{yzt}_1$, and $\bar{l} \rightarrow d_i^{c'+4,j},\ \hat{l} \rightarrow d_i^{c'+2,j},\ \forall j \in \{1, \ldots, l-3\}$ where l design a literal occuring in the clause C_i.
- Moreover we add the following contraints: $d_i^{3,j} \rightarrow l_1^i,\ \forall j \in \{1, \ldots, l-3\}$.

The above construction is illustrated in Figure 2. This transformation can be clearly computed in polynomial time.

The proof is given in an Appendix (see section 4.2).

□

In the following, the Theorem 2.2 will be generalized.

Theorem 2.4 *The problem of deciding whether an instance of* $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c-1); p_i = 1|C_{max}$ *with* $c \geq 3$*, has a schedule of length at most* $(c+3)$ *is* $\mathcal{NP}$*-complete.*

Proof It is easy to see that $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c-1); p_i = 1|C_{max} = c+3 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_2.

Given an instance π^* of Π_2, we construct an instance π of the problem $\bar{P}(Pl \geq 4)|prec; (c_{ij}; \epsilon_{ij}) = (c, c-1); p_i = 1|C_{max} = c+3$, in the following way:

1. For each variable $x \in \mathcal{V}$ we introduce three variables-tasks $x,\ \bar{x}$ and $\hat{x}$ with precedence constraints: $\hat{x} \rightarrow x$ and $\hat{x} \rightarrow \bar{x}$.

2. For each clause $C_i = (x \vee \bar{y})$ of size two, we introduce one clause-task D_i. For every literal l occurring in C_i, we add the precedence constraint $l \rightarrow D_i$.

3. For each clause of size three $C_i = (y \vee z \vee t)$, we introduce two clauses-tasks C_i and $\overline{yzt}$. For every literal l occurring in C_i, we add the precedence constraint $l \rightarrow C_i$ and $\bar{l} \rightarrow \overline{yzt}$.

   We add $(6 \times (l-3) + 2 \times c - 5)$ dummy tasks denoted $d_i^{k,j},\ \forall j \in \{1, \ldots, l-3\},\ k \in \{1, \ldots, c+3\}, l_i^k,\ k \in \{1, \ldots, c-2\}, b_i^{k'}\ k' \in \{1, \ldots, c-3\}$, together with constraints:

   (a) $d_i^{1,j} \rightarrow d_i^{2,j} \rightarrow \ldots \rightarrow d_i^{c+3,j}$.
   (b) $l_i^1 \rightarrow l_i^2 \rightarrow \ldots \rightarrow l_i^{c-2}$.
   (c) $b_i^1 \rightarrow b_i^2 \rightarrow \ldots \rightarrow b_i^{c-3}$.

(d) $l_i^{c-2} \rightarrow \overline{yzt}$ and $b_i^{c-3} \rightarrow C_i$.

(e) For every literal l occurring in C_i, $\hat{l} \rightarrow d_i^{c+1,j}$ and $\hat{l} \rightarrow l_i^{c-2}$ and $d_i^{2,j} \rightarrow b_i^{c-3}$, $\forall j \in \{1, \ldots, l-3\}$.

Sketch of the proof

- Let us first assume that there is a schedule of length at most $(c+3)$. 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. In the same way as prevouisly, the tasks from a path of length four must be executed on the same cluster. Thus the tasks $d_i^{k,j}$ and $\hat{l}$ for every literal l occurring in the clause C_i of size three, must be executed on the same cluster K_i. Moreover, the literal l (resp. $\bar{l}$) is executed on K_i (otherwise it is executed at slot $(c+2)$ or $(c+3)$ and it has two successors).

  Among the six tasks l and $\bar{l}$, only three tasks can be executed at slot 2. Thus, three tasks are executed at slot 3 or after and the three clauses-tasks of type D admitting these tasks as predecessors must be scheduled on K_i at slot (c+3) (the second literal must be executed at slot 2 on other cluster).

  The three tasks l (resp. $\bar{l}$) can not be executed at slot 2 otherwise the task $\overline{yzt}$ (resp. C_i) can not be executed at slot $(c+2)$ on K_i or at slot $(c+3)$ on an other cluster. Therefore, the tasks $\overline{yzt}$ and C_i are executed on K_i at slot $(c+2)$.

  Since the task $\overline{yzt}$ has (c+1) predecessors $\hat{l}$ (for $l = y$, z, or t) and l_i^k. Two of them must be executed at slot 2 and the $(c-1)$ others on the same processor at slot 3 to $(c+1)$. As the task C_i has c precedessors (the literal l and the tasks $b_i^{k'}$), only one can be executed at slot 2, the $(c-1)$ others must be executed on the same processor at slot 3 to $(c+1)$.

  If we affect the **value true** on the literals corresponding to the tasks executed **at slot 2** and **false otherwise**, we have a solution to our problem.

- Conversely, we suppose that there is a truth assignment $I : \mathcal{V} \rightarrow \{0,1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal. It is easy to deduce a schedule of length $(c+3)$.

□

Corollary 2.1 *There is no polynomial-time algorithm for the problem* $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ *with* $c > c'$ *performance bound smaller than* $1 + \frac{1}{c+3}$ *unless* $\mathcal{P} \neq \mathcal{NP}$.

Proof

Corollary 2.1 stems from an immediate consequence of the Impossibility Theorem, (see [11], [12]) and the Theorems 2.1, 2.2, 2.3, 2.4.

□

2.2 The problem of minimizing the sum of all completion times

In this section, we will show that there is no polynomial-time algorithm for the problem $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, c'); p_i = 1|\sum_j C_j$, with $c > c'$, with performance bound smaller than $1 + \frac{1}{2c+4}$ unless $\mathcal{P} \neq \mathcal{NP}$. This result is obtained by the polynomial transformation used for the proof of the Theorem 2.1 and the gap technic (see [15]).

2.2.1 The $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c \geq 3, c');p_i = 1|\sum_j C_j$ problem

Theorem 2.5 *There is no polynomial-time algorithm for the problem* $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c \geq 3, c');p_i = 1|\sum_j C_j$ *with* $c > c'$ *with performance bound smaller than* $1 + \frac{1}{2c+4}$ *unless* $\mathcal{P} \neq \mathcal{NP}$.

Proof

We suppose that there is a polynomial time approximation algorithm denoted by A with performance guarantee ρ with $\rho < 1 + \frac{1}{2c+4}$,

Let be I the instance of the problem $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c,c');p_i = 1|C_{max}$ obtained by a reduction (see Theorem 2.1). Let be I' the instance of the problem $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c,c');p_i = 1|\sum_j C_j$ by adding x new tasks from an initial instance I. In the precedence constraints, each x (with $x > \frac{(2c+5)lc+(c+3)\rho n}{(2c+5)-(2c+4)\rho}$) new tasks is a successor of the old tasks (an old tasks are from the polynomial transformation used for the proof of the Theorem 2.1, 2.2 , 2.3 , 2.4. We obtain a complete graph from the old tasks and the new tasks.

If there exists such an algorithm A, then it can be used to decide an existence of a truth assigment.

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

1. If $A(I') < (2c+5)\rho x + (c+3)\rho n$ then $A^*(I') < (2c+5)\rho x + (c+3)\rho n$. We can deduce that the last of the old tasks in an optimal schedule had been executed at time $(c+3)$ or before. Indeed, we suppose that one task i among the n old tasks is executed at $t = c+4$ in an optimal schedule. Among the x new tasks, only the tasks which are executed on the same cluster as i can be scheduled before the time $t = 2c+5$ (i.e. at most lc tasks). So $A^*(I') > (2c+5)(x - lc)$. Then $x < \frac{(2c+5)lc+(c+3)\rho n}{(2c+4)-(2c+3)\rho}$. A contradiction with $x > \frac{(2c+5)lc+(c+3)\rho n}{(2c+4)-(2c+3)\rho}$.

   Thus, there exists a schedule of length $c+3$ on an old tasks.

2. We suppose that $A(I') \geq (2c+4)\rho x + (c+3)\rho n$. So, $A^*(I') \geq (2c+4)x + (c+3)n$ because an algorithm A is a polynomial time approximation algorithm with performance guarantee ρ. There is no schedule of length at most $c+3$ for the tasks from an Instance I.

   Indeed, if there exist such an algorithm, by executing the x tasks at time $t = 2c+3$, we obtain a schedule with a completion time stricly less than $(2c+4)x + (c+3)n$ (there is at least one task is executed before the time $t = c+2$). A contradiction since $A^*(I') \geq (2c+4)x + (c+3)n$.

Therefore, if there is a polynomial time approximation algorithm with performance guarantee bound smaller than $1+\frac{1}{2c+4}$, it can be used for distinguing in polynomial time the positive instances from the negative instances to the problem , $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c,c');p_i = 1|C_{max}$ thus providing a polynomial time algorithm for a $\mathcal{NP}$-hard problem. Consequently, the problems $\bar{P}(Pl \geq 4)|prec;(c_{ij},\epsilon_{ij}) = (c,c');p_i = 1|\sum C_j$ and does not possess an ρ-approximation, with $\rho < 1 + \frac{1}{2c+4}$.

□

3 Conclusion

In this paper, we first proved that the problem of deciding whether an instance of $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, c' < c); p_i = 1|C_{max}$ has a schedule of length at most $c + 3$ is $\mathcal{NP}$-complete. We generalize the results given by Bampis et al. [5] and Giroudeau [13].

This result is to be compared with the result of [16], which states that $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max} = 6$ is $\mathcal{NP}$-complete. Our result implies that there is no $\rho-$approximation algorithm with $\rho < 1 + \frac{1}{c+3}$, unless $\mathcal{P} = \mathcal{NP}$. In addition, we show that there is no hope to find a ρ-approximation algorithm with ρ strictly less than $\rho < 1 + \frac{1}{2c+4}$ for the problem of the minimization of the sum of the completion time.

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

An interesting question for further research is to find an approximation algorithm with performance guarantee better than the trivial bound of $(c + 1)$ by combining the 4/3-approximation algorithm [17] for the problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ and the 8/5-approximation algorithm [3] for the problem $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{max}$ and developping ρ-approximation in the case of our goal is to find a feasible scheduling of the tasks minimizing a bicriteria conditions.

Acknowledgment. We would like to thank Olivier Cogis for his comments on this work.

References

[1] T.E. Anderson, D.E. Culler, D.A. Patterson, and the NOW team. A case for NOW (networks of workstations). *IEEE Micro*, 15:54–64, 1995.

[2] 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.

[3] E. Bampis, R. Giroudeau, and J.-C. König. A heuristic for the precedence constrained multiprocessor scheduling problem with hierarchical communications. In H. Reichel and S. Tison, editors, *Proceedings of STACS*, LNCS No. 1770, pages 443–454. Springer-Verlag, 2000.

[4] E. Bampis, R. Giroudeau, and J.C. König. Using duplication for multiprocessor scheduling problem with hierarchical communications. *Parallel Processing Letters*, 10(1):133–140, 2000.

[5] E. Bampis, R. Giroudeau, and J.C. König. On the hardness of approximating the precedence constrained multiprocessor scheduling problem with hierarchical communications. *RAIRO-RO*, 36(1):21–36, 2002.

[6] S.N. Bhatt, F.R.K. Chung, F.T. Leighton, and A.L. Rosenberg. On optimal strategies for cycle-stealing in networks of workstations. *IEEE Trans. Comp.*, 46:545–557, 1997.

[7] R. Blumafe and D.S. Park. Scheduling on networks of workstations. In *3d Inter Symp. of High Performance Distr. Computing*, pages 96–105, 1994.

[8] F. Cappello, P. Fraignaud, B. Mans, and A. L. Rosenberg. HiHCoHP-Towards a Realistic Communication Model for Hierarchical HyperClusters of Heterogeneous Processors, 2001. Proceedings of IPDPS'01,IEEE/ACM,IEEE Press.

[9] 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.

[10] P. Chrétienne and J.Y. Colin. C.P.M. scheduling with small interprocessor communication delays. *Operations Research*, 39(3):680–684, 1991.

[11] P. Chrétienne and C. Picouleau. *Scheduling Theory and its Applications*. John Wiley & Sons, 1995. Scheduling with Communication Delays: A Survey.

[12] M.R. Garey and D.S. Johnson. *Computers and Intractability, a Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, 1979.

[13] R. Giroudeau. Non-approximability results in presence of hierarchical communications. Technical Report 02-206, LIRMM, January 2003.

[14] 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.

[15] 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.

[16] 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.

[17] A. Munier and J.C. König. A heuristic for a scheduling problem with communication delays. *Operations Research*, 45(1):145–148, 1997.

[18] G.F. Pfister. *In Search of Clusters*. Prentice-Hall, 1995.

[19] C. Picouleau. New complexity results on scheduling with small communication delays. *Discrete Applied Mathematics*, 60:331–342, 1995.

[20] A.L. Rosenberg. Guidelines for data-parallel cycle-stealing in networks of workstations I: on maximizing expected output. *Journal of Parallel Distributing Computing*, pages 31–53, 1999.

[21] A.L. Rosenberg. Guidelines for data-parallel cycle-stealing in networks of workstations II: on maximizing guarantee output. *Intl. J. Foundations of Comp. Science*, 11:183–204, 2000.

[22] B. Veltman. *Multiprocessor scheduling with communications delays*. PhD thesis, CWI-Amsterdam, Holland, 1993.

4 Appendix

4.1 Premilaries results

In this section, we give one polynomial transformation in order to prove the $\mathcal{NP}$-completeness of the problem Π_2 (the definition of this problem is given below). This problem is used to the polynomial transformation for the $\mathcal{NP}$-completeness of the scheduling problems.

- **The problem** Π_1 is the problem *Monotone-one-in-three-3SAT*. Let us, first recall the definition of *Monotone-one-in-three-3SAT* problem.

  Instance of problem *Monotone-one-in-three-3SAT*:

  - Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be a set of n variables.
  - Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a collection of clauses over $\mathcal{V}$ such that every clause has size three and contains only unnegated variables (the variables in the same clause are different).

  Question:

  Is there a truth assignment for $\mathcal{V}$ such that every clause in $\mathcal{C}$ has exactly one true literal?

  We know that Π_1 is $\mathcal{NP}$-complete [12].

- **The problem** Π_2 is a variant of the well known SAT problem [12]. We will call this variant the *One-in-*$(2,3)SAT(2,\bar{1})$ problem that we will denote as Π_2. 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 for $I : \mathcal{V} \to \{0,1\}$ such that every clause in $\mathcal{C}$ has exactly a true literal?

  In order to illustrate Π_2, we consider the following example.

  Example The following logic formula is a valid instance of Π_2:

  $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_0 \vee x_3) \wedge (\bar{x}_3 \vee x_0) \wedge (\bar{x}_4 \vee x_2) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_5 \vee x_1) \wedge (\bar{x}_2 \vee x_5)$.

  For this instance, the answer to Π_2 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 the Theorem 4.1.

Theorem 4.1 *Π_2 is $\mathcal{NP}$-complete.*

Proof

It is easy to see that $\Pi_2 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_1.

Given any instance π^* of the problem Π_1, we construct an instance π of Π_2 in the following way:

- If a variable x_i occurs only one time, it is sufficient to add a copy of the clause in which it belongs.
- We can suppose that each variable x_i occurs $k_i \geq 2$ times in π^*, then we rename the j^{th} occurrence ($1 \leq j \leq k_i$) of x_i by introducing a new variable $x_{i_{(j-1)}}$. Let $\mathcal{V}'$ be the set of new variables obtained in this way. In every clause of π^*, we rename the occurring variables in a greedy manner and we complete the corresponding instance π by adding the following clauses of length two: $(x_{i_{(j-1)}} \vee \bar{x}_{i_{(j \bmod k_i)}}), \forall i, \forall j, 1 \leq j \leq k_i$. Let $\mathcal{C}'$ be the set of the obtained clauses.

  It is now easy to verify that every instance π of Π_2 obtained by the above construction respects the following two properties:

Property 1: Every variable of $\mathcal{V}'$ occurs three times in π. More precisely, every variable occurs:

- two times unnegated, and more precisely one time in a clause of length three and one time in a clause of length two,
- one time negated in a clause of length two different from the clause in which its unnegated occurrence appears.

Property 2: The variables of $\mathcal{V}'$ occurring in a same clause of length two are such that their unnegated occurrences belong to disjoint clauses of length three.

Property 3: If it exists the clause $(x \vee \bar{y})$ then the clause $(x \vee y \vee z)$ is not allowed.

- Suppose that there is a truth assignment $I : \mathcal{V} \rightarrow \{0,1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal. 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 also exactly one true literal.

  If we take $I'(x_{i_{(j-1)}}) = I(x_i), \forall i, j, 1 \leq j \leq k_i$, we can see first that all clauses of length three become true and each of them has exactly one true literal, since all clauses of length three in $\mathcal{C}$ respect this property.

  In addition, it is clear that every clause of length two is satisfied and only one literal in each of them is true.

  Consequently, if π^* is satisfiable then π is also satisfiable.

- Conversely, assume that there is a truth assignment $I' : \mathcal{V}' \rightarrow \{0,1\}$ such that each clause of $\mathcal{C}'$ has exactly one true literal. 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. Because of the form of the clauses of length two $(x_{i_{(j-1)}} \vee \bar{x}_{i_{(j \bmod k_i)}}), \forall i, \forall j, 1 \leq j \leq k_i$ and given that I' is such that there is exactly one true literal in every clause, we

can conclude that all the variables x_{i_k}, for every fixed i and any k, have the same assignment in I'. In order to find a truth assignment for $\mathcal{C}$, it is sufficient to put $I(x_i) = I'(x_{i_0})$ for every i. Clearly, this assignment respects the desired property of the uniqueness of a true literal per clause.

Consequently, if π is satisfiable then π^* is also satisfiable.

The above transformation can be computed in polynomial time and so Π_2 is $\mathcal{NP}$-complete.

□

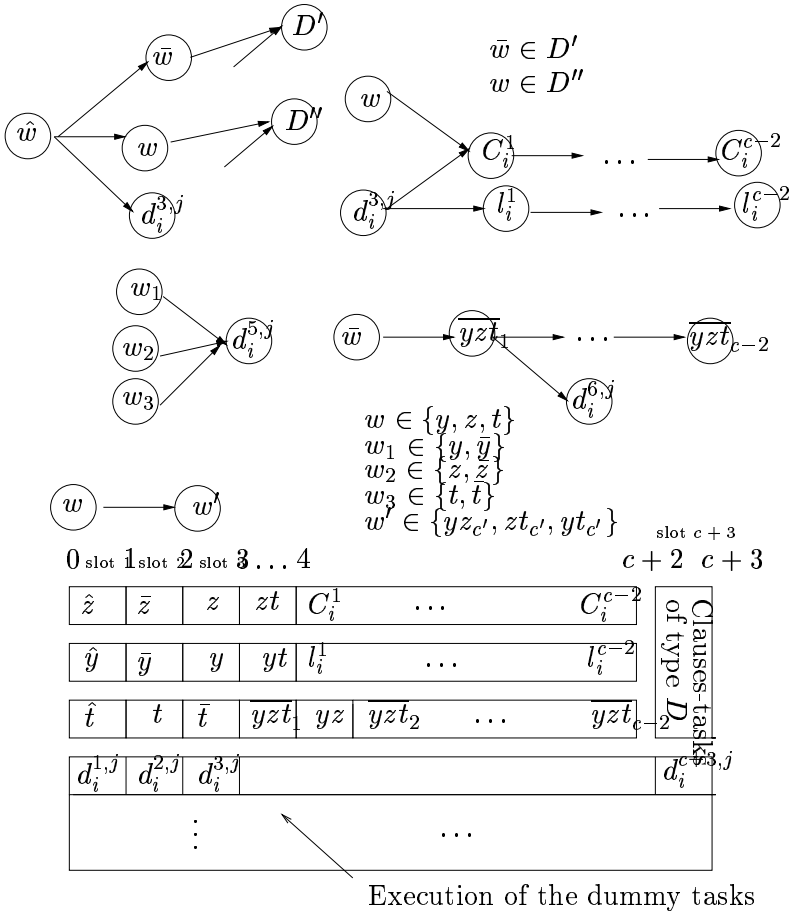


Figure 1: A partial schedule and a partial precedence graph for a clause $C_i = (y \vee z \vee t)$ for the cas $c' = 1$

4.2 Proof of the theorem 2.3

- Let us first assume that there is a schedule of length at most $(c+3)$. 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.

  In this case, we can remark :

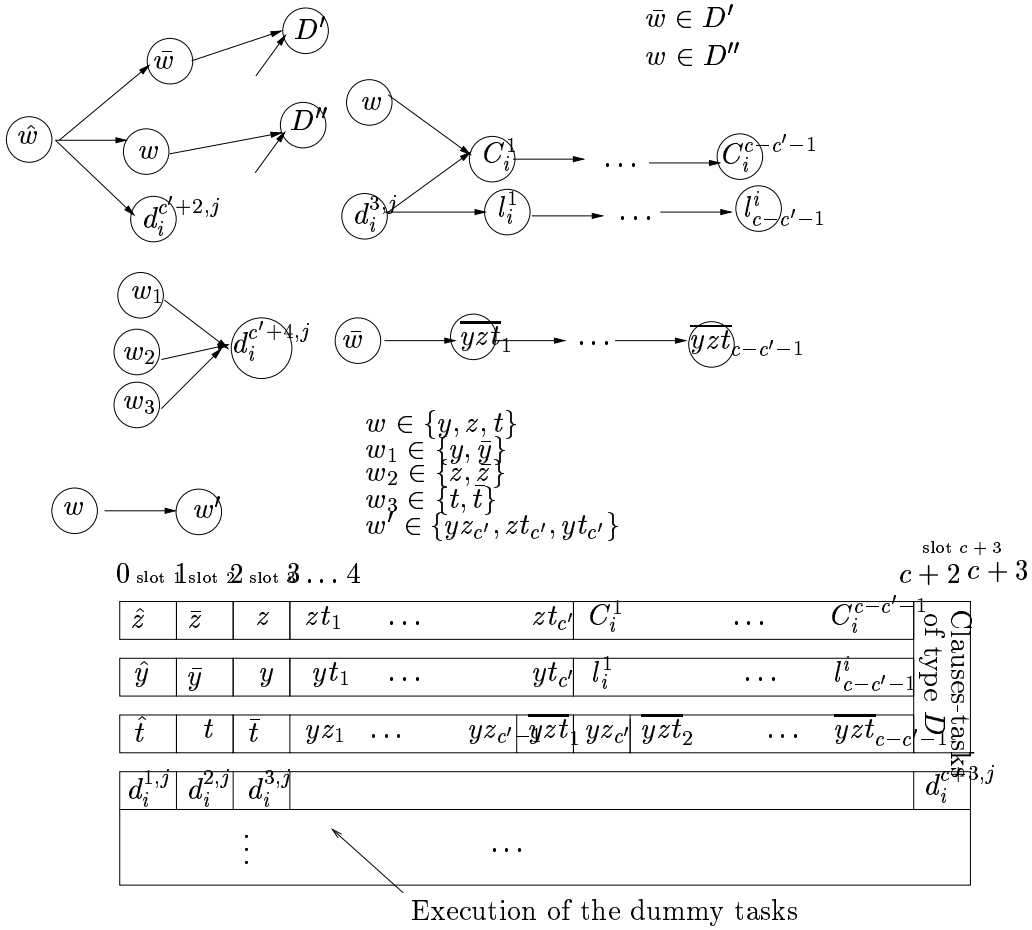


Figure 2: A partial schedule and a partial precedence graph for a clause $C_i = (y \vee z \vee t)$ for the general case

1. All the tasks of a path of length four are executed in the same cluster. Let be $C_i = (y \vee z \vee t)$ a clause. So in the same cluster K_i we must execute the tasks C_i^k, $d_{k,j}^i$, y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$, $\hat{y}$, $\hat{z}$, $\hat{t}$, and l_i^k.
2. The dummy tasks $d_{k,j}^i$ use $(l-3)$ processors of K_i without free time (a communication is not allowed on a path of length $(c+3)$).
3. The arcs from y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$ to the task $d_i^{c'+4,j}$ imply that the tasks y, z, t, $\bar{y}$, $\bar{z}$, $\bar{t}$, $\hat{y}$, $\hat{z}$, and $\hat{t}$ are executed to the slots 1, 2 and 3 on the three others processors. Therefore the tasks $\hat{l}$, l, $\bar{l}$ are processed consecutively on the same processor (with $l = y$ or z or t).
4. At the slot 2, three of the six tasks l and $\bar{l}$ are executed. Thus, three are processed at the slot 3 or after, and the three clauses-tasks of type D containing these literals must be executed on K_i at the slot $(c+3)$ (the second literal of these clauses must be processed at the slot 2).
5. The three tasks $\bar{l}$ can not be executed simultanously at the slot 2. Otherwise, the three tasks l are scheduled at the slot 3 and the tasks $yt_{c'}$, $yz_{c'}$, $zt_{c'}$, C_i^k, l_i^k and $\overline{yzt}_j$ with $k \in \{1, \ldots, c-c'-1\}$, $j \in \{2, \ldots, c-c'-1\}$, must be processed between the slots $(c'+4)$ to $(c+2)$ on three processors. So, there are $(c-c'-1)$

slots on three processors in order to execute $(3 + 2 \times (c - c' - 1) + c - c' - 2)$ tasks (i.e. $3 \times (c - c') - 1$ tasks). It is impossible.

6. At slot 2, one task among the tasks y, z, t must be executed. Otherwise:
   - If $[c > c' + 2]$ The tasks $\overline{yzt}_j$ are executed on K_i (they are on a path of length more than 4). So we must execute on K_i at slot $(c' + 4)$ or after the tasks $\overline{yzt}_j$, $j > 1$, $zt_{c'}$, $yt_{c'}$, $yz_{c'}$, C_i^k, l_i^k and the three clauses-tasks of type D successors of the tasks y, z, and t. Therefore $(3c - 3c' + 2)$ tasks must be executed in $(c - c')$ slots. Impossible.
   - If $[c = c' + 2]$ The tasks $zt_{c'}$, $yt_{c'}$, $yz_{c'}$, C_i^1, l_i^1 and the three clauses-tasks admitting the tasks y, z, t as predecessors must be executed on 2 slots on K_i. Impossible.

Now we affect the **value true** to the literal if the associated task is executed **at slot 2** and **false otherwise**. This gives a solution at our problem.

- 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 $C_i = (y \vee z \vee t)$ is t. Therefore, the schedule, in the Figure 2 is feasible on the three free processors.

This concludes the proof of Theorem 2.3.

4.3 A polynomial time algorithm for $C_{max} = c + 1$

Remark: The problem of deciding whether an instance of $\bar{P}(Pl)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ (resp. $\bar{P}(Pl \geq 3)|prec; (c_{ij}, \epsilon_{ij}) = (c \geq 3, 1); p_i = 1|C_{max}$) has a schedule of length at most $(c+1)$ is solvable in polynomial time since l and c are constants .

Proof

The problem becomes polynomial for $C_{max} = c + 1$. In this case the communication interclusters are forbidden. Therefore, each connected component of the precedence graph must be constitued by at most $l \times (c+1)$ tasks. The problem to determinate if a graph of at most size $l \times (c + 1)$ can be scheduled in $c + 1$ units of times is clearly polynomial.

□