Non-Approximability Results in Presence of Hierarchical Communications

Rodolphe Giroudeau, Jean-Claude König

Laboratoire
d'Informatique
de Robotique
et de Microélectronique
de Montpellier

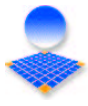
Rapport de Recherche

No : 02-206

Mars 2003

Non-approximability results in presence of hierarchical communications

R. Giroudeau, 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

Non-approximability results in presence of hierarchical communications

Mars 2003

Abstract

We study the problem of minimizing the makespan for the multiprocessor scheduling problem in the presence of a hierarchical communications. We prove that there is no heuristic with performance guarantee smaller than 6/5 (unless $\mathcal{P} = \mathcal{NP}$) for the case where all the tasks of the precedence graph have unit execution times, and the multiprocessor is composed by an unrestricted number of clusters with two identical processors each. We extend the result in the case of the criteria is the sum of the completion time (i.e. we prove that there is no hope to (unless $\mathcal{P} = \mathcal{NP}$) to find a ρ-aproximation algorithm with ρ strictly less than 9/8. We also prove that the problem of deciding whether an instance of $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (2,1); p_i = 1|C_{max}$ has a schedule of length at most three is polynomial.

Keywords: scheduling, hierarchical communications, non-approximability

Résumé

Nous étudions un problème d'ordonnancement dans un système multiprocesseurs en présence de communications hiérarchiques avec pour critère la minimisation de la durée de l'ordonnancement. Nous prouvons qu'il n'existe pas d'heuristique avec une garantie de performance inférieure à 6/5 (sous l'hypothèse que $\mathcal{P} \neq \mathcal{NP}$) pour le cas où toutes les tâches du graphe de précédence ont une durée unitaire et que le système multiprocesseurs est constitué d'un nombre non borné de clusters avec deux processeurs chaque. Le délai de communication pour transférer les données entre une tâche prédécesseur i et une tâche successeur j exécutée sur des clusters différents s'effectue en deux unités de temps (la durée des communication extra-cluster sera dénotée par la suite par c_{ij}), tandis que le coût de transfert est égale à une unité de temps quand les tâches sont ordonnancées sur des processeurs différents à l'intérieur d'un même cluster (la durée des communication intra-cluster sera dénotée par la suite par ϵ_{ij}). Nous allons prouvé que le problème de décider si une instance du problème $\bar{P}, 2|prec; (c_{ij}, \epsilon_{ij}) = (2,1); p_i = 1|C_{max}$ possède un ordonnancement de longueur au plus quatre est $\mathcal{NP}$-complet. Ce résultat sera étendu au problème de la minimisation de la somme des temps de complétude noté par la suite $\sum C_j$ (i.e. qu'il n'existe pas d'algorithme ρ-approché avec $\rho < 9/8$). De plus, nous avons montrer que le problème de décider si une instance du problème $\bar{P}, 2|prec; (c_{ij}, \epsilon_{ij}) = (2,1); p_i = 1|C_{max}$ possède un ordonnancement de longueur au plus trois est polynomial.

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

1 Introduction

With the incresasing 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* [2] 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 clusters. This model captures the hierarchical nature of the communications in todays parallel computers, composed by many networks of PCs or workstations (NOWs) [15, 1]. The use of networks (clusters) of workstations as a parallel computer [15, 1] has renewed the interest of the users in the domain of parallelism, but also created new challenging problems concerning 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 [16, 17, 6, 5]. Only recently, some attempts concerning this issue appeared in the literature [2, 7]. 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, 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 but on different processors 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 all processors may execute other tasks. Our goal is to find a feasible schedule of the tasks minimizing the *makespan*, i.e. the time at which the last task of the precedence graph finishes its execution.

Notice that the hierarchical model that we consider here is a generalization of the classical scheduling model with communication delays ([8], [10]). 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.

Bampis et al. in [4] studied the impact of the hierarchical communications on the complexity of the associated problem. They considered the simplest case, i.e. the prob-

lem $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(1,0);p_i=1|C_{max}$, and they showed that this problem does not possess a polynomial time approximation algorithm with ratio guarantee better than 5/4 (unless $\mathcal{P}=\mathcal{NP}$). If duplication is allowed, Bampis et al. [3] extended the result of [9] in the case of hierarchical communications providing an optimal algorithm for $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(1,0);p_i=1;dup|C_{max}$.

Moreover the authors presented in [2] 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 $\bar{P}(Pm)|prec;(c_{ij},\epsilon_{ij})=(1,0);p_i=1|C_{max}$ leading to a $\frac{4m}{2m+1}$-approximation algorithm.

We study in this paper, in the one hand, 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 simple case where there are only two processors per cluster). The communication delays between the two processors in the same cluster denoted by ϵ_{ij} is equal to one unit of time whereas the communication delays between two processors in a different cluster denoted by c_{ij} is equal to two units of time. Our problem can be denoted as $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|C_{max}$.

We also prove that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|C_{max}$ has a schedule of length at most five is $\mathcal{NP}$-complete. 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. [12].

The paper is organized as follows: in the next section, we prove that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|C_{max}$ has a schedule of length at most five is $\mathcal{NP}$-complete and we extend to this result to the case of the minimization of the completion time. In the last section, we established that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|C_{max}$ has a schedule of length at most three is solvable in polynomial time

2 The non-approximability result

2.1 The minimization of the length of the makespan

In order to prove that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|C_{max}$ has a schedule of length at most five is an $\mathcal{NP}$-complete problem, we use a reduction from a variant of the well known SAT problem [11]. We will call this variant the *One-in-*$(2,3)SAT(2,\bar{1})$ problem denoted in what follows by Π_1. This problem is a $\mathcal{NP}$-complete problem (see [4]).

Π_1 is a restricted variant of SAT with clauses of length two and three, where each variable x occurs three times: two of these occurrences are unnegated (in the form of literal x) and one negated (in the form of literal $\bar{x}$). One of the unnegated occurrences is in a clause of length three and the other in a clause of length two. The literal $\bar{x}$ is necessarily in a clause of length two different from the clause of size two in which the literal x occurs. Thus the clause $(x\vee\bar{x})$ cannot exist in any instance of Π_1. We are interested about the existence or not of a truth assignment in which every clause has *exactly* one true literal.

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 is sufficient to choose $x_0 = 1$, $x_3 = 1$ and $x_i = 0$ for $i = \{1, 2, 4, 5\}$. This gives a truth assignment satisfying the formula, and there is exactly one true literal in every clause.

Formally, *One-in-*$(2,3)SAT(2, \bar{1})$ can be stated as follows:

Instance of problem Π_1:

- Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be a set of variables and $\bar{\mathcal{V}} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ the set of negated variables with $x_i \in \mathcal{V}$.
- Let $\mathcal{C} = \{C_1, \ldots, C_j, C_{j+1}, \ldots, C_q\}$ be a set of clauses where $\forall i, 1 \leq i \leq j, C_i \subset (\mathcal{V} \times \bar{\mathcal{V}})$ and $\forall i, (j+1) \leq i \leq q, C_i \subset (\mathcal{V})^3$, and such that every variable $x_i \in \mathcal{V}$ occurs two times unnegated and one time negated:

  $\forall x_i \in \mathcal{V}, \exists i_1, i_2, i_3$ such that

$$\begin{cases} occurrence(x_i; C_{i_1}) = 1 \quad \text{and} \quad occurrence(x_i; C_{i_2}) = 1, & 1 \leq i_1 \leq j, j+1 \leq i_2 \leq q \\ occurrence(\bar{x}_i; C_{i_3}) = 1 & 1 \leq i_3 \leq j, i_3 \neq i_1 \end{cases}$$

  where $occurrence(x_i; C_{i_k})$ is a function that gives the number of times where variable x_i occurs in the clause C_{i_k}.

  In addition, if $(x_i \in C_k \text{ and } \bar{x}_{i'} \in C_k, 1 \leq k \leq j) \Rightarrow (x_i \in C_l)$ and $(x_{i'} \in C_r)$, with $l \neq r$ and $(j+1) \leq l, r \leq q$.

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?

Recall, that the $\mathcal{NP}$-completeness of Π_1 is based on the two successives polynomial transformation given below: $Monotone\text{-}One\text{-}In\text{-}Three\text{-}3SAT \propto Monotone - One - in - three - 3SAT^*(2 \geq)$ and $Monotone - One - In - Three - 3SAT^*(2 \geq) \propto One - In - (2,3)SAT(2, \bar{1})$.

The definition of *Monotone-one-in-three-*$3SAT$ problem is:

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.

Question:

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

$Monotone - One - in - three - 3SAT^*(2 \geq)$ is defined in the same way as *Monotone-one-in-three-*$3SAT$, except that in $Monotone - One - in - three - 3SAT^*(2 \geq)$ every variable occurs at least twice and there are not two occurrences of a same variable in the same clause.

Theorem 2.1 *The problem of deciding whether an instance of* $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ *has a schedule of length at most five is* $\mathcal{NP}$*-complete.*

Proof

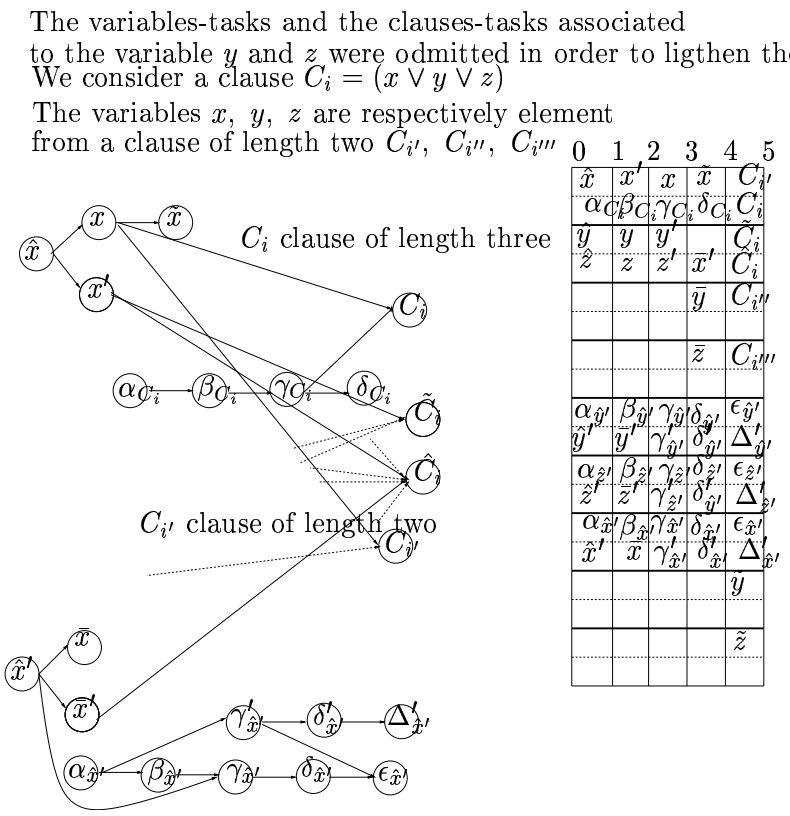


Figure 1: Variable-tasks, clause-tasks

It is easy to see that $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max} = 5 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_1. Let n be the number of variables in the logic formula. The clauses are labelled from 1 to q.

Given an instance π^* of Π_1, we construct an instance π of the problem $\bar{P}(P2)|prec;(c_{ij};\epsilon_{ij}) = (2;1);;p_i = 1|C_{max} = 5$, in the following way:

- For each variable $x \in \mathcal{V}$, we introduce four variable-tasks $\hat{x}, x, x'$ and $\tilde{x}$. The precedence constraints between these tasks are the following:

  $\hat{x} \to x$, $\hat{x} \to x'$, and $x \to \tilde{x}$.

- For each variable $\bar{x} \in \bar{\mathcal{V}}$, we introduce eleven variable-tasks $\hat{x}'$, $\bar{x}$, $\bar{x}'$, $\alpha_{\hat{x}'}$, $\beta_{\hat{x}'}$, $\gamma_{\hat{x}'}$, $\delta_{\hat{x}'}$, $\epsilon_{\hat{x}'}$ and $\gamma'_{\hat{x}'}$, $\delta'_{\hat{x}'}$, $\Delta'_{\hat{x}'}$. The precedence constraints between these tasks are the following

  $\hat{x}' \to \bar{x}$, $\hat{x}' \to \bar{x}'$, $\hat{x}' \to \gamma_{\hat{x}'}$, $\alpha_{\hat{x}'} \to \beta_{\hat{x}'} \to \gamma_{\hat{x}'} \to \delta_{\hat{x}'} \to \epsilon_{\hat{x}'}$ and $\alpha_{\hat{x}'} \to \gamma'_{\hat{x}'}$, $\gamma'_{\hat{x}'} \to \epsilon'_{\hat{x}'}$. $\gamma'_{\hat{x}'} \to \delta_{\hat{x}'} \to \Delta_{\hat{x}'}$

- For every clause C_i, we introduce one clause-task C_i such that for every literal x occurred in C_i, we add the precedence constraint $x \to C_i$.

- For every clause $C_i, \forall i, j+1 \leq i \leq q$ (notice that the length of these clauses is three), we introduce two clause-tasks $\tilde{C}_i,\ \hat{C}_i$ and four variables-tasks $\alpha_{C_i},\ \beta_{C_i},\ \gamma_{C_i},\ \delta_{C_i}$.

  The precedence constraints between these tasks are the following:

  $\alpha_{C_i} \rightarrow \beta_{C_i} \rightarrow \gamma_{C_i} \rightarrow \delta_{C_i}$ and $\gamma_{C_i} \rightarrow C_i$.

  For every literal x occurred in $C_i, \forall i,\ j+1 \leq i \leq q$, we add the precedence constraints $x' \rightarrow \tilde{C}_i, x' \rightarrow \hat{C}_i$ and $\bar{x}' \rightarrow \hat{C}_i$.

- For every clause $C_i, \forall i,\ 1 \leq i \leq j$ (notice that the length of these clauses is two), we introduce one clause-task C_i. For every literal x occurred in $C_i, \forall i,\ 1 \leq i \leq j$, we add the precedence constraint : $x \rightarrow C_i$

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 truth assignment $I : \mathcal{V} \rightarrow \{0,1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal for the problem Π_1. Then we will prove that there is a schedule of length at most five.

  Let us construct this schedule:

  - The chain of variables-tasks $\alpha_{C_i},\ \beta_{C_i},\ \gamma_{C_i},\ \delta_{C_i},\ \forall i,\ j+1 \leq i \leq q$ corresponding to the clause of length three are executed on the differents clusters. In the same way, the chain of variables-tasks $\alpha_{\hat{x}'} \rightarrow \beta_{\hat{x}'} \rightarrow \gamma_{\hat{x}'} \rightarrow \delta_{\hat{x}'} \rightarrow \epsilon_{\hat{x}'}$ and $\alpha_{\hat{x}'} \rightarrow \gamma'_{\hat{x}'},\ \gamma'_{\hat{x}'} \rightarrow \delta'_{\hat{x}'} \rightarrow \Delta'_{\hat{x}'}$ associated to a variable-task $\bar{x}$, is executed on a different cluster.
  - If the literal x is "true", then variable-task x is processed at $t = 2$ on the same cluster as the chain of length four of variables-tasks with initial tasks α_{C_i}, with $j+1 \leq i \leq q$, if the variable x occurred in the clause C_i. So we execute at $t = 3$ (resp. at $t = 1$) on the same processor as the variable-task x, the variable-task $\tilde{x}$ (resp. x'). Consequently the variable-task $\hat{x}$ is scheduled at $t = 0$ on the same processor as x'. In addition, we execute at $t = 4$ on the same cluster as x the clauses-tasks C_i with $j+1 \leq i \leq q$ and C'_i with $1 \leq i' \leq j$ where the variable x occurred.

    Else, we execute at $t = 1$ on the same cluster as the second literal of the clause of length three, denoted by z, valued to "false" (we know that there is exactly one true literal per clause, see the definition of the problem Π_1). Consequently, with the precedence constraints and the communications delays, the variables-tasks x' and z' (resp. $\bar{y}'$) is processed at $t = 2$ (resp. at $t = 3$). Thus, at $t = 4$, the clause-task $\tilde{C}_i$ is executed on the same preocessor as $\bar{y}'$ and on the second processor the clause-task $\tilde{C}_i$ is processed. The variable-task $\tilde{x}$ and $\tilde{z}$ are processed at $t = 4$ on differents clusters.
  - If the literal $\bar{x}$ is "false", then variable-task $\bar{x}$ is scheduled at $t = 1$ on the same cluster as the chain of length five of variables-tasks with initial task $\alpha_{\hat{x}'}$. In addition, the variable-task $\bar{x}'$ is executed at $t = 3$ on the same cluster as $\tilde{C}_i$ and $\hat{C}_i$ with C_i is the clause of length three where the variable x occured.

    Else, we execute $\bar{x}$ at $t = 3$ on the same cluster as the clause-task C_i with $1 \leq i \leq j$ where the variable $\bar{x}$ occurred. In addition, we execute $\bar{x}'$ at $t = 1$, on the same cluster as the chain of variables-tasks with initial task $\alpha_{\hat{x}'}$.

The above way of scheduling the tasks preserves the precedence constraints and the communication delays and gives a schedule of length five, whenever there is a truth assignment with exactly one true literal per clause.

- Conversely, suppose now that there is a schedule of length at most five. We will prove that there is an assignment $I : \mathcal{V} \rightarrow \{0, 1\}$ such that every clause has exactly one true literal.

  We start by making five essential observations. We notice that in every feasible schedule of length at most five.

  1. The chain of variables-tasks $\alpha_{C_i},\ \beta_{C_i},\ \gamma_{C_i},\ \delta_{C_i}$ with $\forall i,\ j+1 \leq i \leq q$ must be executed consecutively on the same processor as the clause-task C_i associated to the clause of length three. Notice that with the chain of this variables-tasks, only one variable-task associated to the variables which occur in the clause of length three, can be executed on the same cluster as this chain (see the Lemma 2.1).
  2. The chain of variables-tasks $\alpha_{\hat{x}'},\ \beta_{\hat{x}'},\ \gamma_{\hat{x}'},\ \delta_{\hat{x}'},\ \epsilon_{\hat{x}'}$ with $\bar{x} \in \bar{\mathcal{V}}$ must be processed consecutively on the same processor as $\hat{x}$, and on the same cluster as the variables-tasks $\gamma'_{\hat{x}},\ \delta'_{\hat{x}},\ \Delta'_{\hat{x}}$ which are respectively scheduled at $t = 2$, $t = 3$ and $t = 4$. Thus, none of the variables-tasks $\bar{x}'$ and $\bar{x}$ cannot be processed at $t = 2$. Notice that with the chain of this variables-tasks, the variables-tasks $\bar{x}$ and $\bar{x}'$ are scheduled on differents clusters and not at the same time (see the Lemma 2.2).
  3. With the precedence constraints and the communications delays, the variables-tasks $\hat{x}$ and $\hat{x}'$ (resp. the clauses-tasks) must be executed at $t = 0$ (resp. $t = 4$).
  4. With the precedence constraints, the task $\bar{x}$ (resp. $\bar{x}'$) is processed at $t = 1$ either at $t = 3$. In the same way, the task x is processed at $t = 2$ either at $t = 1$.
  5. The clauses-tasks $\hat{C}$ and $\tilde{C}$ must scheduled at $t = 4$ on the same cluster.

Lemma 2.1 *In any valid schedule of length at most five, the tasks x, y and z associated to the variables x, y and z from the clause $(x \vee y \vee z)$ cannot be executed simultanously. Consequently, one these three tasks have a starting time at $t = 2$ and the others at $t = 1$ on the same cluster.*

Proof

We suppose that the three variables x, y and z associated to the variables-tasks x, y and z are elements from a clause denoted in what follows by C_i.

It is clear that the tasks x, y and z cannot be executed at the $t = 3$ (resp. at $t = 2$) with the precedence constraints and the communications delays.

We suppose that the three tasks are executed at $t = 1$:

1. We suppose that the three tasks are scheduled on the differents clusters. Thus, the variables-tasks x', y' and z' have a starting time at $t = 2$ and with the precedence constraints and the communications delays, the clauses-tasks $\tilde{C}_i$ and $\hat{C}_i$ have a starting time at $t = 5$. It is impossible.

2. We suppose that two of three tasks, denoted in what follows by x and y, have a starting time at $t = 1$ on the same cluster. Notice that the task x' (resp. y') is processed at $t = 2$ on the same processor as x (resp. as y).

   At the matter of fact of the communications delays, the task z' must be executed at $t = 3$ on the same cluster as the tasks $\hat{x}$ and $\hat{y}$ and, then the clause-task $\tilde{C}_i$ (resp. $\hat{C}_i$) is processed at $t = 4$. But, the variable-task z' admits another successor the clause-task $\hat{C}_i$ (resp. $\tilde{C}_i$), and with the communications delays intra-cluster, the clause-task $\hat{C}_i$ (resp. $\tilde{C}_i$) must be executed at $t = 5$. It is impossible.

Notice that two of three variables-tasks from cannot be executed at $t = 2$ on the same cluster since the two variables-tasks admit three clauses-tasks as successors.

In conclusion, one of three variables-tasks must be executed at $t = 2$ and the other at $t = 1$ on the same cluster. Indeed, if the two variables-tasks are executed at $t = 1$ on the differents clusters, then the variables-tasks successors are scheduled at $t = 3$ and consequently the clauses-tasks are processed at $t = 5$.

□

Lemma 2.2 *In any valid schedule of length at most five, the variables-tasks x and $\bar{x}$ cannot be scheduled simultanously.*

Proof

1. We suppose that the tasks x and $\bar{x}$ are processed on the same cluster:
   (a) at $t = 3$, it's impossible since these two variables-tasks admit three clauses-tasks.
   (b) at $t = 2$, it's impossible since the variable-task x is scheduled at the same time on the same cluster as the variable-task γ_{C_i} where C_i is the clause of the length three where the variable x occurred.
   (c) at $t = 1$, it's impossible since with the Lemma 2.1, if the variable-task x is processed at $t = 1$ then at the same time on the second processor on the same cluster an another variable-task, associated to the variable which occurs in the clause C_i with $1 \leq i \leq j$, must be processed.
2. We suppose that the tasks x and $\bar{x}$ are processed on the differents clusters:
   (a) at $t = 3$, it's impossible since the variable-task x admits three successors: the variable-task $\tilde{x}$ and the clauses-tasks C_i and $C_{i'}$ associated to the clauses C_i and $C_{i'}$ where the variable x occurs.
   (b) at $t = 2$, it's impossible since if $\bar{x}$ is executed at $t = 2$, with the precedence constraints and the communications delays, the variable-task $\bar{x}$ must be executed at the same time and on the same cluster as the variables-tasks $\gamma_{\hat{x}}$ and $\gamma'_{\hat{x}'}$.
   (c) at $t = 1$, it's impossible since in this case the variables-tasks x' and $\bar{x}'$ must be executed at $t = 3$ on the same cluster as the clause-task $\hat{C}_i$. With the communications delays, this clause-task is processed at $t = 5$.

In conclusion, the variables-tasks x and $\bar{x}$ cannot be scheduled at the same time. □

Lemma 2.3 *We consider a clause $C = (x \vee y \vee z)$. In any valid schedule of length at most five, the tasks $\bar{x}', \bar{y}'$ and $\bar{z}'$ cannot be executed simultanously. Consequently, one of these three tasks $\bar{x}$, $\bar{y}$ and $\bar{z}$ must be processed at $t = 1$ and the other at $t = 3$ on a different cluster.*

Proof

It is clear that the variables-tasks $\bar{x}'$, $\bar{y}'$ and $\bar{z}'$ cannot be scheduled at $t = 2$ with the precedence constraints and the communications delays. We suppose that they are scheduled at $t = 1$. Thus, the variables-tasks $\bar{x}$, $\bar{y}$ and $\bar{z}$ must be executed at $t = 3$ on the differents clusters with the chain of length five of variables-tasks associated to the clause of length two where the variables x, y and z occured.

In order to clarify the presentation, we rename the variables of the logic formula in the following way: given that every variable occurs exactly once in a clause of length three, we rewrite the clauses of length three in the following form $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge \ldots \wedge (x_{n-3} \vee x_{n-2} \vee x_{n-1})$ with $x_i \neq x_j$ for all $i, j \in \{0, ..., n-1\}$.

W.l.o.g. we assume in the following that $x = x_0$, $y = x_1$ and $z = x_3$. In order to prove that we cannot assign $\bar{x}'_0$, $\bar{x}'_1$ and $\bar{x}'_2$ at $t = 1$ (and consequently $\bar{x}_0$, $\bar{x}_1$ and $\bar{x}_2$ at $t = 3$), we will show first, that the assignment of $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ implies an unique assignment for "some" literals, that we will call *critical literals*, (below we precise the way of identifying these literals) appearing to the set of clauses of length two and that this assignment implies a schedule of length greater than five, contradicting in this way the assumption. Before explaining the assignment of the literals, we introduce a method for the enumeration of the literals appearing in the set of clauses of length two. For every instance of Π_1, we consider the variable x_0. We know by the definition of Π_1 that the variable x_0 occurs two times in the set of clauses of length two: one time unnegated (in the form of literal x_0) and one time negated (in the form of literal $\bar{x}_0$). Now, we will explain a method to enumerate the *critical literals*:

We consider the variable $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$. Let $(\bar{x}_0 \vee x_{k_1}), (\bar{x}_1 \vee x_{k_2}), (\bar{x}_2 \vee x_{k_3})$ be the clauses of length two where the literals $\bar{x}_0, \bar{x}_1$ and $\bar{x}_3$ appear respectively. The literals $x_{k_i}, i \in \{1, 2, 3\}$ are two to two disjoints. Now we consider the variables x_{k_1}, x_{k_2} and x_{k_3}:

There are two cases to be considered:

1. If it exists the clauses $(x_0 \vee \bar{x}_{k_1}), (x_2 \vee \bar{x}_{k_2})$ and $(x_1 \vee \bar{x}_{k_3})$, then the *criticals literals* are $x_0, \bar{x}_0, x_1, \bar{x}_1, x_2, \bar{x}_2, x_{k_1}, \bar{x}_{k_1}, x_{k_2}, \bar{x}_{k_2}, x_{k_3}$ and $\bar{x}_{k_3}$.
2. If the precedence clauses are not existing, then they continue. The literals $\bar{x}_{k_1}, \bar{x}_{k_2}$ and $\bar{x}_{k_3}$ occur also respectively in a clause of length two denoted respectively by $(\bar{x}_{k_1} \vee x_{k_4}), (\bar{x}_{k_2} \vee x_{k_5})$ and $(\bar{x}_{k_3} \vee x_{k_6})$ such that $x_{k_4} \neq x_0, x_{k_5} \neq x_1, x_{k_6} \neq x_2$ and $x_{k_i}, i \in \{4, 5, 6\}$ are two to two disjoints.
   Up to now the critical literals are $\bar{x}_0, \bar{x}_1, \bar{x}_2, x_{k_1}, x_{k_2}, x_{k_3}, \bar{x}_{k_1}, \bar{x}_{k_2}, \bar{x}_{k_3}, x_{k_4}, x_{k_5}, x_{k_6}$ and $\bar{x}_{k_4}$. We continue in this way until finding the first clause not containing a "new" variable. The following claim shows that there is always such a clause. The set of critical literals will contain all the literals encountered using the above described procedure.

Claim: *During the procedure of enumeration of critical literals given above, we will always find a clause* $(x_{k_j} \vee \bar{x}_{k_{j'}})$ *with* $k_j \neq k_{j'}$ *such that the variables* x_{k_j} *and* $x_{k_{j'}}$ *have been already enumerated.*

Proof of the claim

Let n be the number of variables in the logic formula. Thus, there are n clauses of length two and $2n$ literals.Let us assume that the $(n-1)$ first clauses of length two that we examine by applying the procedure of enumeration do not verify the assumption of the claim. In this case, we have $(2n-2)$ critical literals. It remains only 2 literals to enumerate. We know that a clause of the form $(x_m \vee \bar{x}_m)$ cannot exist in any instance of Π_1 and consequently we can conclude that the remaining clause contains two literals that correspond to two variables already encountered.

□

Now, we will prove that the assumption concerning an assignment of the tasks $\bar{x}'_0, \bar{x}'_1$ and $\bar{x}'_2$ at $t = 1$ (and consequently $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ at $t = 3$) is false. For this, we show that an existence of a schedule of length at most fve implies an unique way for an assignment of the tasks that corresponding to the *criticals literals.*

Two cases have to be taken into account:

By the precedence constraints and the communcations delays the variables-tasks x_{k_1}, x_{k_2} and x_{k_3}, (notice that these tasks cannot be executed on the same cluster at $t = 2$ as $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ respectively, since the variables-tasks x_{k_1}, x_{k_2} and x_{k_3} admit, respectively, a successor-task $\tilde{x}_{k_1}$, $\tilde{x}_{k_2}$ and $\tilde{x}_{k_3}$, which is executed at $t = 3$), must be executed at $t = 1$. By the Lemma 2.3, we know that the variables-tasks x and $\bar{x}$ cannot be scheduled at the same time. Then, the variables-tasks $\bar{x}_{k_0}, \bar{x}_{k_1}$ and $\bar{x}_{k_2}$ are processed at $t = 3$.

1. We consider the clauses $(\bar{x}_0 \vee x_{k_1}), (\bar{x}_1 \vee x_{k_2}), (\bar{x}_2 \vee x_{k_3})$. If it exists the clauses $(x_0 \vee \bar{x}_{k_1}), (x_2 \vee \bar{x}_{k_2})$ and $(x_1 \vee \bar{x}_{k_3})$, then by the same argument x_0, x_1 and x_2 are processed at $t = 1$. Then, at $t = 1$, all variables $(x \in \mathcal{V})$ in an unnegated form (in the form of literal x) are executed. According to the Lemma 2.1, we know that the three variables-tasks associated to the variables occured in a clause of length three cannot be scheduled at $t = 1$. So, the three variables-tasks ($\bar{x}_0$, $\bar{x}_1$ and $\bar{x}_2$ cannot be executed at $t = 3$), and consequently the variables-tasks $\bar{x}'$, $\bar{y}'$ and $\bar{z}'$ cannot be executed at the same time and consequently one of theses three tasks $\bar{x}$, $\bar{y}$ and $\bar{z}$ must be executed at $t = 1$ and the other at $t = 3$.
2. We consider the partial logic formula composed by the clauses of length two containing the criticals literals: $(\bar{x}_0 \vee x_{k_1}) \wedge (\bar{x}_1 \vee x_{k_2}) \wedge (\bar{x}_2 \vee x_{k_3}) \ldots \wedge (x_{k_i} \vee \bar{x}_{k_j}) \wedge (x_{k_{j'}} \vee \bar{x}_{k_{i'}}) \wedge \ldots (x_{k_j} \vee \bar{x}_{k_{j'}})$ with $k_i \neq k_{i'}$ and $k_i \neq k_j, k_{i'} \neq k_{j'}$.
   Recall that the variables-tasks $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ are executed at $t = 3$ and by the precedence constraints and the communications delays the variables-tasks x_{k_1}, x_{k_2} and x_{k_3} must be processed at $t = 1$. Since that the variables $\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ occur in the clause of length two denoted by $(\bar{x}_{k_1} \vee x_{k_4}), (\bar{x}_{k_2} \vee x_{k_5})$ and $(\bar{x}_{k_3} \vee x_{k_6})$, according to the precedence constraints and the communications delays the variables x_{k_4}, x_{k_5} and x_{k_6} are scheduled at $t = 1$. We repeat the assignment for every critical literal $x_{k_i}, k_i \in \{1, 2, \ldots, n-1\}$. Then, at $t = 1$, all variables $(x \in \mathcal{V})$ in an unnegated form (in the form of literal x) are executed.

According to the Lemma 2.1, we know that the three variables-tasks associated to the variables occured in a clause of length three cannot be scheduled at $t = 1$. So, the three variables-tasks ($\bar{x}_0, \bar{x}_1$ and $\bar{x}_2$ cannot be executed at $t = 3$and, consequently the variables-tasks $\bar{x}'$, $\bar{y}'$ and $\bar{z}'$ cannot be executed at the same time. So, consequently one of theses three tasks $\bar{x}$, $\bar{y}$ and $\bar{z}$ must be executed at $t = 1$ and the other at $t = 3$.

□

Lemma 2.4 *In any valid schedule of length at most five, if the task x is scheduled at $t = 2$ (resp. at $t = 1$) then the task $\bar{x}$ is processed at $t = 1$ (resp. at $t = 3$).*

Proof

By the Lemma 2.1, we know that one of these three variables-tasks x, y and z associated to the variables x, y and z, which occur in the clause of length three denoted by $C = (x \vee y \vee z)$, must be executed at $t = 2$ and the two other on the same cluster at $t = 1$.

By the Lemma 2.2, we know that the variables-tasks x and $\bar{x}$ cannot be processed at the same time.

- For the variables-tasks x which are executed at $t = 1$, the variable-task $\bar{x}$ cannot be processed at $t = 2$ since by the precedence constraints and the communications delays $\bar{x}$ should be executed at the same time on the same cluster as the variables-tasks $\gamma_{\hat{x}'}$ and $\gamma'_{\hat{x}'}$. Thus, in any valid schedule of length at most five, if the task x is scheduled at $t = 1$ then the task $\bar{x}$ is processed at $t = 3$.
- We consider now that the task x is processed at $t = 2$. We also suppose that $\bar{x}$ is scheduled at $t = 3$. The variables occurred in the clause of length three denoted by $(x \vee y \vee z)$. By the Lemma 2.1, the tasks y and z are executed at $t = 1$ and by the Lemma 2.3, the variables-tasks $\bar{y}$ and $\bar{z}$ are executed at $t = 3$. We know that $\bar{y}$ and $\bar{z}$ occur in the clause of length two $(\bar{y} \vee t)$ and $(\bar{z} \vee r)$ and the variables-tasks are processed at $t = 1$ and so on for all variables. By a same argument as the Lemma 2.3, all variables, in the unnegated form, except x are executed at $t = 1$. It's impossible by the Lemma 2.1.

□

Remark: Notice that the task x cannot be executed at $t = 3$ in any valid schedule of length at most five, since in the one hand that the interprocessor communication is equal to one and in the other hand that the task x have two successors, the variable-task $\tilde{x}$ and the clause-task where the literal x occurred.

There exist two possibilities for scheduling the variable-task x and $\bar{x}$ on the same processor (see Lemma 2.1) or on the different clusters (see Lemma 2.3).

- Define a literal x as "true" (resp. "false") if the corresponding variable-task x is processed at time $t = 2$ (resp. at $t = 1$).
- Define a literal $\bar{x}$ as "true" (resp. "false") if the corresponding variable-task $\bar{x}$ is processed at time $t = 3$ (resp. at $t = 1$).

Lemma 2.5 *If an instance π of the problem $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ has a schedule of length at most five, then the corresponding instance π^* of Π_1 has a truth assignment satisfying the logic formula and such that at each clause there is exactly one true literal.*

Proof

According to remark 4 and the Lemma 2.4 and as each clause-task has been completed at time 5, we know that each clause contains exactly one true literal.

□

□

Corollary 2.1 *There is no polynomial-time algorithm for the problem $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ with performance bound smaller than $\frac{6}{5}$ unless $\mathcal{P} \neq \mathcal{NP}$.*

Proof

The proof of Corollary 2.1 is an immediate consequence of the Impossibility Theorem, (see [10], [11]).

□

2.2 The problem of the minimization of the completion time

In this section, we will show that there is no polynomial-time algorithm for the problem $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|\sum_j C_j$ with performance bound smaller than $\frac{9}{8}$ 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 [12]).

Theorem 2.2 *There is no polynomial-time algorithm for the problem $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|\sum_j C_j$ with performance bound smaller than $\frac{9}{8}$ unless $\mathcal{P} \neq \mathcal{NP}$.*

Proof

In order to obtain this result, we consider the polynomial transformation used for the proof of the Theorem 2.1. This transformation use a set of M machines, (M is arbitrary high) and a set of unitary tasks which are subject to precedence constraints and communications delays.

Notice that :

- In the case of an instance of the $\mathcal{NP}$-complete problem Π_1 receives a positive answer (i.e. we can conclude to an existence of a truth assignment such that every clause has exactly one true literal, see case a) of the Figure [2]), in the schedule instance, the n tasks are executed during an interval $[0,5]$. Thus, the sum of the completion time is to equal to $C(n)$ units of time.
- In the case of an instance of the $\mathcal{NP}$-complete problem Π_1 receives a negative answer (i.e. we can conclude to the non-existence of a truth assignment such that every clause has exactly one true literal, see case b) of the Figure [2]), for any feasible schedule at least, for the constructed instance, one task have a completion at 6 or more. Thus, the sum of the completion time is least equal to $C'(n) > C(n)$ units of time.

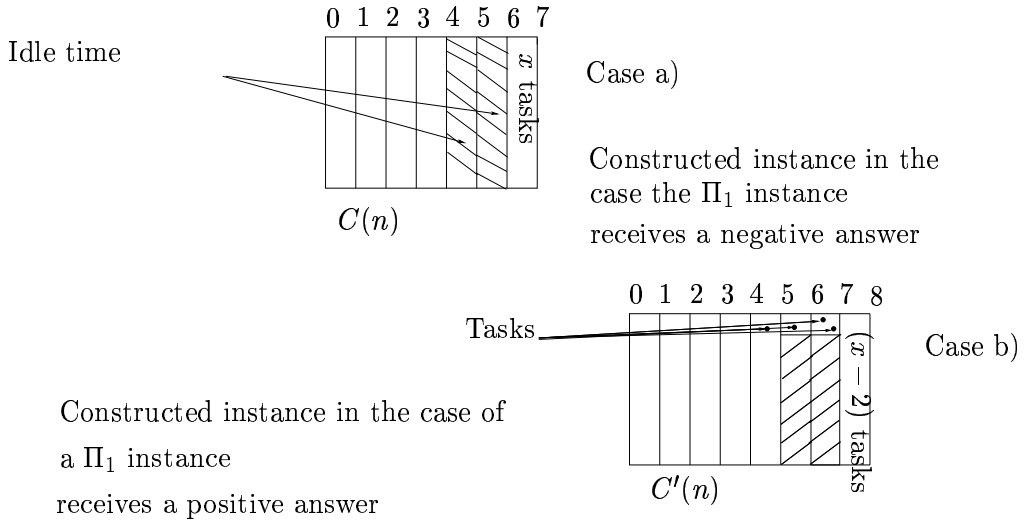


Figure 2: Construction of the polynomial transformation for the completion time criteria from the polynomial transformation for the length of the schedule.

We add x new tasks from an initial instance. In the precedence constraints, each new tasks is a successor of an old tasks (an old tasks are from the polynomial transformation used for the proof of the Theorem 2.1). We obtain a complete graph between the new tasks and an old tasks.

An instance is denoted by I^* and we can observe the following properties:

- In the case of an instance of the $\mathcal{NP}$-complete problem Π_1 receives a positive answer, then from an instance I^* a schedule can be constructed where the sum of the completion time equal to $C(n)+8x$ (see the case a) of the Figure [2]). We add a two idles times between the new tasks and an old tasks in order to respect the communications delays.

- In the case of an instance of the $\mathcal{NP}$-complete problem Π_1 receives a negative answer, then from an instance I^* a schedule can be constructed where the sum of the completion time is at least equal to $C'(n)+16+9(x-2)$ (see the case b) of the Figure 2). Indeed, at least one task is executed at $t=5$. Thus, there are at most two tasks which can be scheduled at $t=7$ on the same cluster as the task executed at $t=5$. Thus, the sum of the completion time is at least equal to $C'(n)+8+8+9(x-2)$.

Therefore, if there is a polynomial time approximation algorithm with performance guarantee bound smaller than $9/8$, it can be used for distinguing in polynomial time the positive instances from the negative instances of the problem Π_1, providing a polynomial time algorithm for a $\mathcal{NP}$-hard problem. Consequently, the problems $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij})=(2,1);p_i=1|\sum C_j$ and does not possess an ρ-approximation, with $\rho<9/8$.

□

3 A polynomial time algorithm for $C_{max} = 3$

Theorem 3.1 *The problem of deciding whether an instance of* $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ *has a schedule of length at most three is solvable in polynomial time.*

Proof

The problem becomes polynomial for $C_{max} = 3$. Indeed, the intercusters communications is forbidden, then each connected component must be constitued by at most six tasks. The problem to determinate if a graph of at most size six can be scheduled in three units of times is clearly polynomial. Notice all the graphs of at most size six are subgraph of the graphs given by the Figure 3. □

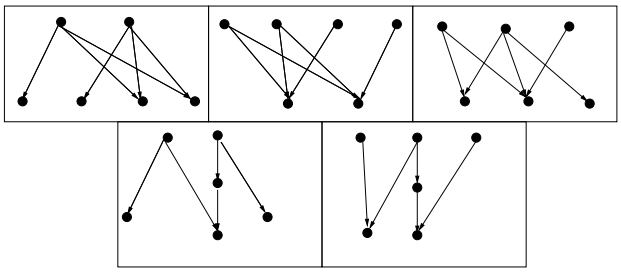


Figure 3: List of graphs with 6 vertices

Conjecture : We conjecture that for $C_{max} = 4$, that it exists a polynomial time algorithm.

4 Conclusion

In this paper, we first proved that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ has a schedule of length at most five is $\mathcal{NP}$-complete.

This result is to be compared with the result of [13], 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 $\rho < \frac{6}{5}$, unless $\mathcal{P} = \mathcal{NP}$.

Second, we established that the problem of deciding whether an instance of $\bar{P}(P2)|prec;(c_{ij},\epsilon_{ij}) = (2,1);p_i = 1|C_{max}$ has a schedule of length at most three is solvable in polynomial time. In addition, we show that there is no hope to find a ρ-approximation algorithm with ρ strictly less than $\rho < 9/8$ for the problem of the minimization of the sum of the completion time.

An interesting question for further research is to find an approximation algorithm with performance guarantee better than the trivial bound of three by combining the 4/3-approximation algorithm [14] for the problem $\bar{P}|prec;c_{ij} = 1;p_i = 1|C_{max}$ and the 8/5-approximation algorithm [2] 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.

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, 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] F. Cappello, P. Fraignaud, B. Mans, and A. L. Rosenberg. HiHCoHP-Towards a Realistic Communication Model for Hierarchical HyperClusters of Heterogeneous Processors, 2000. Proceedings of IPDPS'01,IEEE/ACM,IEEE Press.

[8] 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.

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

[10] P. Chrétienne, E.J. Coffman Jr, J.K. Lenstra, and Z. Liu. *Scheduling Theory and its Applications*. Wiley, 1995.

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

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.