Scheduling with Communication Delay
Rodolphe Giroudeau, Jean-Claude König

To cite this version:

HAL Id: lirmm-00195552
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00195552
Submitted on 11 Dec 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Chapter 1

Scheduling with communication delays

R. Giroudeau and J.C. König
LIRMM, 161 rue Ada,34392 Montpellier Cedex 5, France, UMR 5506

1.1 Introduction

More and more parallel and distributed systems (cluster, grid and global computing) are both becoming available all over the world, and opening new perspectives for developers of a large range of applications including data mining, multimedia, and bio-computing. However, this very large potential of computing power remains largely unexploited this being, mainly due to the lack of adequate and efficient software tools for managing this resource.

Scheduling theory is concerned with the optimal allocation of scarce resources to activities over time. Of obvious practical importance, it has been the subject of extensive research since the early 1950’s and an impressive amount of literature now exists. The theory dealing with the design of algorithms dedicated to scheduling is much younger, but still has a significant history.

An application which will be scheduled on a parallel architecture may be represented by an acyclic graph $G = (V, E)$ (or precedence graph) where $V$ designates the set of tasks, which will be executed on a set of $m$ processors, and where $E$ represents the set of precedence constraints. A processing time is allotted to each task $i \in V$.

From the very beginning of the study about scheduling problems, models kept up with changing and improving technology. Indeed,
In the **PRAM's model**, in which communication is considered instantaneous, the critical path (the longest path from a source to a sink) gives the length of the schedule. So the aim, in this model, is to find a partial order on the tasks, in order to minimize an objective function.

In the **homogeneous scheduling delay model**, each arc \((i, j) \in E\) represents the potential data transfer between task \(i\) and task \(j\) provided that \(i\) and \(j\) are processed on two different processors. So the aim, in this model, is to find a compromise between a sequential execution and a parallel execution.

These two models have been extensively studied over the last few years from both the complexity and the (non)-approximability points of view (see [23] and [12]). With the increasing importance of parallel computing, the question of how to schedule a set of tasks on a given 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 they are part of the most studied problems in the domain.

In this chapter, we adopt the **hierarchical communication model** [7] in which we assume that the communication delays are not homogeneous anymore; the processors are connected into **clusters** and the communications inside a same cluster are much faster than those between processors belonging to different ones. This model incorporates the hierarchical nature of the communications using today's parallel computers, as shown by many PCs or workstations networks (NOWs) [33, 1]. The use of networks (clusters) of workstations as a parallel computer [33, 1] has not only renewed the user's interest in the domain of parallelism, but it has also brought forth many new challenging problems related to the exploitation of the potential power of computation offered by such a system.

Several approaches meant to try and model these systems were proposed taking into account this technological development:

- One approach concerning the form of programming system, we can quote work [36, 37, 11, 9].

- In abstract model approach, we can quote work [41, 26, 27, 15, 10, 28, 16] on malleable tasks introduced by [10, 15]. A malleable task is a task which can be computed on several processors and of which the execution time depends on the number of processors used for its execution.

As stated above, the model we adopt here is the **hierarchical communication model** which addresses one of the major problems that arises in the efficient use of 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 communication delays with successively higher latencies. In a formal context where both a set of clusters of identical processors, and a precedence graph $G = (V, E)$ are given, 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 negligible (resp. is equal to what we call inter-processor communication delay). On the contrary, if these tasks are executed on different clusters, then the communication delay is more significant and is called inter-cluster communication delay.

We are given $m$ multiprocessor machines (or clusters denoted by $\Pi^i$) that are used to process $n$ precedence-constrained tasks. Each machine $\Pi^i$ (cluster) comprises several identical parallel processors (denoted by $\pi^i_k$). A couple $(c_{ij}, \epsilon_{ij})$ of communication delays is associated to each arc $(i, j)$ between two tasks in the precedence graph. In what follows, $c_{ij}$ (resp. $\epsilon_{ij}$) is called inter-cluster (resp. inter-processor) communication, and we consider that $c_{ij} \geq \epsilon_{ij}$. If tasks $i$ and $j$ are allotted on different machines $\Pi^i$ and $\Pi^j$, then $j$ must be processed at least $c_{ij}$ time units after the completion of $i$. Similarly, if $i$ and $j$ are processed on the same machine $\Pi^i$ but on different processors $\pi^i_k$ and $\pi^j_{k'}$ (with $k \neq k'$) then $j$ can only start $\epsilon_{ij}$ 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 (inter-cluster or inter-processor delay) does not interfere with the availability of processors and any processor may execute any task. Our goal is to find a feasible schedule of tasks minimizing the makespan, i.e., the time needed to process all tasks subject to the precedence graph.

Formally, in the hierarchical scheduling delay model a hierarchical couple of values $(c_{ij}, \epsilon_{ij})$ will be associated with $\epsilon_{ij} \leq c_{ij} \forall (i, j) \in E$ such that:

- if $\Pi^i = \Pi^j$ and $\pi^i_k = \pi^j_k$ then $t_i + p_i \leq t_j$
- else if $\Pi^i = \Pi^j$ and $\pi^i_k \neq \pi^j_{k'}$ 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 starting 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 communication delays are taken into account and that completion time (makespan) is minimized (the makespan is denoted by $C_{\text{max}}$ and it corresponds to $\max_{i \in V} \{t_i + p_i\}$). In what follows, we consider the simplest case $\forall i \in V, p_i = 1, c_{ij} = c \geq 2, \epsilon_{ij} = c' \geq 1$ with $c \geq c'$. 

3
Note that the hierarchical model that we consider here is a generalization of classical scheduling model with communication delays ([12], [14]). Consider, for instance, that for every arc $(i, j)$ of the precedence graph we have $c_{ij} = e_{ij}$. In such a case, the hierarchical model is exactly the classical scheduling communication delays model.

Note that the values $c$ and $l$ are considered as constant in the following.

The chapter is organized as follow: In the next section, some results for $UET$-$UCT$ model will be presented. In the section 1.3, a lower and upper bound for large communication delays scheduling problem will presented. In the section 1.4, the principal results in hierarchical communication delay model will be presented. In the section 1.5, an influence of an introduction of the duplication on the complexity of scheduling problem is presented. In the section 1.6, some results non-approximability results are given for the total sum of completion time minimization. In the section 1.7, we will conclude on the complexity and approximation scheduling problem in presence of communication delays. In Appendix section, some classical $NP$–complete problems are listed which are used in this chapter for the polynomial-time transformations.

### 1.2 Some results for the $UET$-$UCT$ model

In the homogeneous scheduling delay model, each arc $(i, j) \in E$ represents the potential data transfer between task $i$ and task $j$ provided that $i$ and $j$ are processed on two different processors. So the aim, in this model, is to find a compromise between a sequential execution and a parallel execution. These two models have been extensively studied over the last few years from both the complexity and the (non)-approximability points of view (see [23] and [12]).

1. at any time, a processor executes at most one task;

2. $\forall (i, j) \in E$, if $\pi_i = \pi_j$ then $t_j \geq t_i + p_i$, otherwise $t_j \geq t_i + p_i + c_{ij}$.

The makespan of schedule $\sigma$ is: $C_{\sigma_{\max}} = \max_{i \in V}(t_i + p_i)$.

In the $UET$-$UCT$ model, we have $\forall i, p_i = 1$ and $\forall (i, j) \in E, c_{ij} = 1$.

#### 1.2.1 Unbounded number of processors

In the case of there is no communication delays, the problem becomes polynomial (even if we consider that $\forall i, p_i \neq 1$). In fact, the Bellman algorithm can be used.

**Theorem 1.2.1** The problem of deciding whether an instance of $P|\text{prec, } p_i = 1, c_{ij} = 1|C_{\max}$ problem has a schedule of length 5 is polynomial, see [42].
Proof
The proof is based on the notion of total unimodularity matrix, see [42] and see [39]. □

Theorem 1.2.2 The problem of deciding whether an instance of \( \bar{P} | \text{prec}, p_i = 1, c_{ij} = 1 | C_{\text{max}} \) problem has a schedule of length 6 is \( \mathcal{NP} \)-complete see [42].

Proof
The proof is based on the following reduction \( 3\text{SAT} \propto \bar{P} | \text{prec}; c_{ij} = 1; p_i = 1 | C_{\text{max}} = 6 \).

Figure 1.1: The variables-tasks and the clauses-tasks

It is clear that the problem is in \( \mathcal{NP} \).

Let be \( \pi^* \) an instance of \( 3\text{SAT} \) problem, we construct an instance \( \pi \) of the problem \( \bar{P} | \text{prec}; c_{ij} = 1; p_i = 1 | C_{\text{max}} \) in the following way:

- For each variable \( x \), six tasks are introduced: \( x_1, x_2, x_3, \bar{x}, x \) and \( x_6 \); the precedence constraints are given by Figure 1.1.

- for each clause \( c = (x_c, y_c, z_c) \), where the literals \( x_c \), \( y_c \) and \( z_c \) are occurrences of negated or unnegated, 13 variables are introduced.
JOB SHOP AND MACHINE SCHEDULING

\[ \bar{x}_c, \ y_c, \ z_c, \ x_c, \ y_c, \ z_c, \ \bar{z}_c, \ x_c y_c, \ y_c z_c, \ x_c z_c \text{ and } c : \text{the precedence constraints between these tasks are also given by Figure 1.1.} \]

- If the occurrence of variable \( x \) in the clause \( c \) is unnegated then we add \( x_c \rightarrow x \) and \( \bar{x}_c \rightarrow \bar{x} \).
- If the occurrence of variable \( x \) in the clause \( c \) is negated, then we add \( x_c \rightarrow \bar{x} \) and \( \bar{x}_c \rightarrow x \).

Clearly, \( x_c \) represents the occurrence of variable \( x \) in the clause \( c \); it precedes the corresponding variable tasks. This is a polynomial-time transformation illustrated by Figure 1.1.

It can be proved that, there exists a schedule of length at most six if only if there is a truth assignment \( I : V \rightarrow \{0, 1\} \) such that each clause in \( C \) has at least one true literal.

\[ \square \]

Corollary 1.2.1 There is no polynomial-time algorithm for the problem \( \bar{P}|\text{prec}; c_{ij} = c = 1; p_i = 1|C_{\text{max}} \) with performance bound smaller than \( 7/6 \) unless \( \bar{P} \neq \bar{NP} \), see [42].

Proof

The proof of Corollary 1.2.1 is an immediate consequence of the Impossibility Theorem, (see [14], [17]).

\[ \square \]

1.2.2 Approximate solutions with guaranteed performance

Good approximation algorithms seem to be very difficult to design, since the compromise between parallelism and communication delays is not easy to handle. In this section, we will present a approximation algorithm with a performance ratio bounded by \( 4/3 \) for the problem \( \bar{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\text{max}} \). This algorithm is based on a formulation on an integer linear program. A feasible schedule is obtained by a relaxation and rounding procedure. Notice that it exists a trivial 2-approximation algorithm: the tasks without predecessors are executed at \( t = 0 \), the tasks admitting predecessors scheduled at \( t = 0 \) are executed at \( t = 2 \) and so on.

Given a precedence graph \( G = (V, E) \) a predecessor (resp. successor) of a task \( i \) is a task \( j \) such that \((j, i)\) (resp. \((i, j)\)) is an arc of \( G \). For every task \( i \in V, \Gamma^+(i) \) (resp. \( \Gamma^-(i) \)) denotes the set of immediate successors (resp. predecessors) of \( i \). We denote the tasks without predecessor (resp. successor) by \( Z \) (resp. \( U \)). We call source every task belonging to \( Z \).
The integer linear program  The aim of this section is to model the problem $P|\text{prec}; c_{ij} = 1; p_i = 1|C_{\text{max}}$ by an integer linear program (ILP) denoted, in what follows, by $\Pi$.

We model the scheduling problem by a set of equations defined on the starting times vector $(t_1, \ldots, t_n)$:

For every arc $(i, j) \in E$, we introduce a variable $x_{ij} \in \{0, 1\}$ which indicates the presence or not of an communication delay, and the following constraints: $\forall (i, j) \in E, t_i + p_i + x_{ij} \leq t_j$

In every feasible schedule, every task $i \in V - U$ has at most one successor, w.l.o.g. call them $j \in \Gamma^+(i)$, that can be performed by the same processor as $i$ at time $t_j = t_i + p_i$. The other successors of $i$, if any, satisfy: $\forall k \in \Gamma^+(i) - \{j\}, t_k \geq t_i + p_i + 1$.

Consequently, we add the constraints: $\sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 1$

Similarly, every task $i$ of $V - Z$ has at most one predecessor, w.l.o.g. call them $j \in \Gamma^-(i)$, that can be performed by the same processor as $i$ at times $t_j$ satisfying $t_i - (t_j + p_j) < 1$. So, we add the following constraints: $\sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 1$.

If we denote by $C_{\text{max}}$ the makespan of the schedule, $\forall i \in V, t_i + p_i \leq C_{\text{max}}$

Thus, in what follows, the following ILP will be considered:

$$
\begin{align*}
\forall (i, j) \in E, \quad & x_{ij} \in \{0, 1\} \\
\forall i \in V, \quad & t_i \geq 0 \\
\forall (i, j) \in E, \quad & t_i + p_i + x_{ij} \leq t_j \\
\forall i \in V - U, \quad & \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 1 \\
\forall i \in V - Z, \quad & \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 1 \\
\forall i \in V, \quad & t_i + p_i \leq C_{\text{max}}
\end{align*}
$$

(II)

Let $\Pi^{inf}$ denote the linear program corresponding to $\Pi$ in which we relax the integrity constraints $x_{ij} \in \{0, 1\}$ by setting $x_{ij} \in [0, 1]$. Given that the number of variables and the number of constraints are polynomially bounded, this linear program can be solved in polynomial time. The solution of $\Pi^{inf}$ will assign to every arc $(i, j) \in E$ a value $x_{ij} = e_{ij}$ with $0 \leq e_{ij} \leq 1$ and will determine a lower bound of the value of $C_{\text{max}}$ that we denote by $\Theta^{inf}$.

Lemma 1.2.1  $\Theta^{inf}$ is a lower bound on the value of an optimal solution for $P|\text{prec}; c_{ij} = 1; p_i \geq 1|C_{\text{max}}$.

Proof This is true since any optimal feasible solution of the scheduling problem must satisfy all the constraints of the integer linear program $\Pi$. $\Box$
Algorithm 1 Rounding Algorithm and construction of the schedule

Step 1 [Rounding]
Let be $e_{ij}$ the value of an arc $(i, j) \in E$ given by $PL^\inf_I$

\[
\begin{align*}
\text{si } e_{ij} < 0.5 & \implies x_{ij} = 0 \\
\text{si } e_{ij} \geq 0.5 & \implies x_{ij} = 1
\end{align*}
\]

Step 1 [Computation of starting time]
if $i \in Z$ then
\[ t_i = 0 \]
else
\[ t_i = \max\{t_j + 1 + x_{ji}\} \text{ avec } j \in \Gamma^-(i) \text{ et } (j, i) \in A_i \]
end if

Step 2 [Construction of the schedule]
Let be $G' = (V; E')$ where $E' = E \setminus \{(i, j) \in E | x_{ij} = 1\}$ \{G' is generated by the 0-arcs.\}
Allotted each connected component of $G'$ on a different processor.
Each task is executed at its starting time.

In the following, we call an arc $(i, j) \in E$ a 0-arc (resp. 1-arc) if $x_{ij} = 0$ (resp. $x_{ij} = 1$).

Lemma 1.2.2 Every job $i \in V$ has at most one successor (resp. predecessors) such that $e_{ij} < 0.5$ (resp. $e_{ji} < 0.5$).

Proof We consider a task $i \in V$ and his successors $j_1, \ldots, j_k$ such that $e_{i,j_1} \leq e_{i,j_2} \leq \ldots \leq e_{i,j_k}$. We know that $\sum_{l=1}^{k} e_{i,j_l} \geq k - 1$, then $2e_{i,j_2} \geq e_{i,j_2} + e_{i,j_1} \geq k - 1 - \sum_{l=3}^{k} e_{i,j_l}$. Since that $e_{i,j_l} \in [0, 1]$, $\sum_{l=3}^{k} e_{i,j_l} \leq k - 2$. Then, $2e_{i,j_2} \geq 1$. Therefore $\forall l \in \{2, \ldots, k\}$ we have $e_{ij} \geq 0.5$. We use the same arguments for the predecessors. □

Lemma 1.2.3 The scheduling algorithm described above provides a feasible schedule.

Proof It is clear that each task $i$ admits at most one incoming (resp. outcoming) 0-arcs. □

Theorem 1.2.3 The relative performance $\rho^h$ of our heuristic is bounded above by $\frac{3}{4}$ [31].

8
Proof Let be a path \( x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{k+1} \) constituted by \((k+1)\) tasks such that \( x \) (resp. \((k-x)\)) arcs values, given by linear programming, between two tasks are less (resp. least) than 1/2. So the length of this path is less than \( k+1+1/2(k-x) = 3/2k - 1/2x + 1 \). Moreover, by the rounding procedure, the length of this path at most \( 2k - x + 1 \). Thus, we obtain \( \frac{2k-x+1}{3/2k-1/2x+1} < 4/3, \forall x \). Thus, for a given path, of value \( p^* \) (resp. \( p \)) before (resp. after) the rounding, admitting \( x \) arcs values less than 1/2, we have \( \frac{p}{p^*} \leq \frac{2k-x+1}{3/2k-1/2x+1} < 4/3 \). A critical path before the rounding phase is denoted by \( s^* \). It is true for the critical path after the rounding procedure \( p = s \) then, \( \frac{p}{p^*} < \frac{p}{s^*} = \frac{s}{s} = 4/3 \). □

In fact, the bound is tight (see [31]).

1.2.3 Bounded number of processors

In this section, a lower and upper bound will be presented,

Theorem 1.2.4 The problem of deciding whether an instance of \( P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\max} \) problem has a schedule of length 3 is polynomial, see [34].

Theorem 1.2.5 The problem of deciding whether an instance of \( P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\max} \) problem has a schedule of length 4 is \( \mathcal{NP} \)-complete, see [42].

Proof The proof is based on the \( \mathcal{NP} \)-complete problem Clique.
The dots lines between the two sets $A$ and $B$ signify a complete graph between the two sets.

**Figure 1.2:** Example of polynomial-time reduction $\text{clique} \propto P|\text{prec}; c_{ij} = 1; p_i = 1|C_{\text{max}}$.

Let be $l = \frac{k(k-1)}{2}$ the number of edges of a clique of size $k$. Let be $m' = \max\{|V| + l - k, |E| - l\}$, the number of processors of an instance is $m = 2(m' + 1)$. It is clear that the problem is in $\mathcal{NP}$. The proof is based on the polynomial-time reduction $\text{clique} \propto P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\text{max}}$. Let be $\pi^*$ a instance of the $\text{clique}$ problem. An instance $\pi$ of $P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\text{max}}$ problem is constructed in the following way:

- $\forall v \in V$ the tasks $T_v, K_v$ are introduced,
• ∀e ∈ E a task L_e is created.
  – We add the following precedence constraints: T_v → K_v, ∀v ∈ V and
    T_v → L_e if v is an endpoint of e.

• Four sets of tasks are introduced:
  – X_x = {x = 1, . . . , x = m − l − |V| + k},
  – Y_y = {y = 1, . . . , y = m − |E| + l},
  – U_u = {u = 1, . . . , u = m − k},
  – W_w = {w = 1, . . . , w = m − |V|}.

  – the precedence constraints are added: U_u → X_x, U_u → Y_y, W_w → Y_y

![Figure 1.3: Example of construction in order to illustrate the proof of theorem 1.2.5](image)

It easy to see that the graph G admits a clique of size k if only if it exists a schedule
of length 4.

1.2.4 Approximation algorithm

In this section, we will present a simple algorithm which gives a schedule σ^m
on m machines from a schedule σ^∞ on unbounded number of processors for the
\( P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\text{max}} \). The validity of this algorithm is based on the fact
there is at most a matching between the tasks executed at \( t_i \) and the tasks processed
at \( t_i + 1 \).

**Theorem 1.2.6** From all polynomial-time algorithm \( h^* \) with performance guaran-
tee \( \rho \) for the problem \( P|\text{prec}, c_{ij} = 1, p_i = 1|C_{\text{max}} \), we may obtain a
Algorithm 2 Scheduling on $m$ machines from a schedule $\sigma^\infty$ on unbounded number of processors

\begin{algorithm}
\begin{algorithmic}
\FOR{$i = 0 \text{ à } C^\infty_{\text{max}} - 1$}
\STATE Let be $X_i$ the set of tasks executed at $t_i$ in $\sigma^\infty$ using a heuristic $h^*$. \\
\STATE The $X_i$ tasks are executed in $\lceil \frac{|X_i|}{m} \rceil$ units of time.
\ENDFOR
\end{algorithmic}
\end{algorithm}

A polynomial-time algorithm with performance guarantee $(1 + \rho)$ for the problem $P|\text{prec}, c_{ij} = 1, p_i = 1|C^\text{max}$.

**Proof**

\[
C_{\text{max}}^{m,A} \leq \sum_{i=0}^{C^\infty_{\text{max}} - 1} \left\lceil \frac{|X_i|}{m} \right\rceil \leq \sum_{i=0}^{C^\infty_{\text{max}} - 1} (\lfloor \frac{|X_i|}{m} \rfloor + 1) \leq \sum_{i=0}^{C^\infty_{\text{max}} - 1} (\lfloor \frac{|X_i|}{m} \rfloor) + C^\infty_{\text{max}} \\
\leq \sum_{i=0}^{C^\infty_{\text{max}} - 1} (\frac{|X_i|}{m}) + C^\infty_{\text{max}} \leq C^\text{opt, m}_{\text{max}} + C^\infty_{\text{max}} \leq C^\text{opt, m}_{\text{max}} + \rho C^\text{opt, m}_{\text{max}}
\]

For example, the $4/3$-approximation algorithm gives a $7/3$-approximation algorithm. Munier et al. [29] propose a $(7/3 - 4/3m)$-approximation algorithm for the same problem.

### 1.3 Large communications delays

Scheduling in presence of large communication delays, is one most difficult problem in scheduling theory, since the starting time of tasks and the communication delay are not be synchronized.

If we consider the problem of scheduling a precedence graph with large communication delays and unit execution time (UET-LCT), on a restricted number of processors, Bampis et al. in [4] proved that the decision problem denoted by $P|\text{prec}; c_{ij} = c \geq 2; p_i = 1|C^\text{max}$ for $C^\text{max} = c + 3$ is an $\mathcal{NP}$-complete problem, and for $C^\text{max} = c + 2$ (for the special case $c = 2$), they develop a polynomial-time algorithm. This algorithm can not be extended for $c \geq 3$. Their proof is based on a reduction from the $\mathcal{NP}$-complete problem $\text{Balanced Bipartite Complete Graph}, \text{BBCG} [17, 38]$. Thus, Bampis et al. [4] proved that the $P|\text{prec}; c_{ij} = c \geq 2; p_i = 1|C^\text{max}$ problem does not pos-
SCHEDULING WITH COMMUNICATION DELAYS

A polynomial-time approximation algorithm with ratio guarantee better than $(1 + \frac{1}{c+3})$, unless $P = NP$.

\[ \begin{align*}
C_1^i & \rightarrow C_2^i \\
A_1^i & \rightarrow A_2^i \\
C_{c+1}^i & \rightarrow C_{c+2}^i \\
A_{c+1}^i & \rightarrow A_{c+2}^i \\
\forall l \in V
\end{align*} \]

Figure 1.4: A partial precedence graph for the $NP$-completeness of the scheduling problem $P|prec; c_{ij} = c; p_i = 1|C_{max}$.

Theorem 1.3.1 The problem of deciding whether an instance of $P|prec; c_{ij} = c; p_i = 1|C_{max}$ has a schedule of length equal or less than $(c+4)$ is $NP$-complete with $c \geq 3$ (see [22]).

Proof

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

The proof is based on a reduction from $\Pi_1$. Given an instance $\pi^*$ of $\Pi_1$, we construct an instance $\pi$ of the problem $P|prec; c_{ij} = c; p_i = 1|C_{max} = c + 4$, in the following way (Figure 1.4 helps understanding of the reduction):

1. For all $l \in V$, we introduce $(c + 6)$ variable-tasks: $\alpha_{l_1}, l', \bar{l}', \hat{l}', \beta_{1j}$ with $j \in \{1, 2, \ldots, c+2\}$. We add the precedence constraints: $\alpha_{l_1} \rightarrow l', \alpha_{l_1} \rightarrow \bar{l}', \beta_{1j} \rightarrow \bar{l}', \beta_{1j} \rightarrow \hat{l}'$ with $j \in \{1, 2, \ldots, c+1\}$.

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

The above construction is illustrated in Figure 1.4. This transformation can be clearly computed in polynomial time.

**Remark:** $l'$ is in the clause $C'_{i}$ of length two associated with the path $D_1^i \rightarrow D_2^i \rightarrow \ldots D_{c+2}^i \rightarrow D_{c+3}^i$.

It is easy to see that there is a schedule of length equal or less than $(c + 4)$ if only if there is a truth assignment $I : V \rightarrow \{0, 1\}$ such that each clause in $C$ has exactly one true literal (i.e. one literal equal to 1), see [22].

**Theorem 1.3.2** The problem of deciding whether an instance of $\bar{P}_{\text{prec}}; c_{ij} = 2; p_i = 1 | C_{\text{max}}$ has a schedule of length equal or less than six is $\mathcal{NP}$-complete (see [22]).

**Corollary 1.3.1** There is no polynomial-time algorithm for the problem $\bar{P}_{\text{prec}}; c_{ij} = c \geq 2; p_i = 1 | C_{\text{max}}$ with performance bound smaller than $1 + \frac{1}{c+4}$ unless $P \neq \mathcal{NP}$ (see [22]).

**Theorem 1.3.3** The problem of deciding whether an instance of $\bar{P}_{\text{prec}}; c_{ij} = c; p_i = 1 | C_{\text{max}}$ with $c \in \{2, 3\}$ has a schedule of length at most $(c + 2)$ is solvable in polynomial time (see [22]).

### 1.3.1 Approximation by expansion

In this section, a new polynomial-time approximation algorithm with performance guarantee non-trivial for the problem $\bar{P}_{\text{prec}}; c_{ij} = 2; p_i = 1 | C_{\text{max}}$ will be proposed.

**Notation:** We denote by $\sigma^\infty$, the UET-UCT schedule, and by $\sigma_c^\infty$ the UET-LCT schedule. Moreover, we denote by $t_i$ (resp. $t_i^c$) the starting time of the task $i$ in the schedule $\sigma^\infty$ (resp. in the schedule $\sigma_c^\infty$).

**Principle:** We keep an assignment for the tasks given by a “good” feasible schedule on an unrestricted number of processors $\sigma^\infty$. We proceed to an expansion of the makespan, while preserving communication delays ($t_j^i \geq t_i^c + 1 + c$) for two tasks, $i$ and $j$ with $(i, j) \in E$, processing on two different processors.

Consider a precedence graph $G = (V, E)$, we determine a feasible schedule $\sigma^\infty$, for the model UET-UCT, using a $(4/3)$—approximation algorithm proposed by Munier and König [31]. This algorithm gives a couple $\forall i \in V$, $(t_i, \pi)$ on the
SCHEDULING WITH COMMUNICATION DELAYS

schedule $\sigma^\infty$ corresponding to: $t_i$ the starting time of the task $i$ for the schedule $\sigma^\infty$ and $\pi$ the processor on which the task $i$ is processed at $t_i$.

Now, we determine a couple $\forall i \in V, (t^c_i, \pi')$ on schedule $\sigma^\infty$ in the following way: The starting time $t^c_i = d \times t_i = \frac{(c+1)}{2} t_i$ and, $\pi = \pi'$. The justification of the expansion coefficient is given below. An illustration of the expansion is given in Figure 1.5.

![Diagram](image)

**Figure 1.5**: Illustration of notion of an expansion

**Lemma 1.3.1** The coefficient of an expansion is $d = \frac{(c+1)}{2}$.

**Proof** Consider two tasks $i$ and $j$ such that $(i, j) \in E$, which are processed on two different processors in the feasible schedule $\sigma^\infty$. Let be $d$ a coefficient $d$ such that $t^c_i = d \times t_i$ and $t^c_j = d \times t_j$. After an expansion, in order to respect the precedence constraints and the communication delays we must have $t^c_j \geq t^c_i + 1 + c$, and so $d \times t_i - d \times t_j \geq c + 1$, $d \geq \frac{c+1}{t_i-t_j}$, $d \geq \frac{c+1}{2}$. It is sufficient to choose $d = \frac{(c+1)}{2}$.

**Lemma 1.3.2** An expansion algorithm gives a feasible schedule for the problem denoted by $P|\text{prec}; c_{ij} = c \geq 2; p_i = 1|C_{\text{max}}$.

**Proof** It is sufficient to check that the solution given by an expansion algorithm produces a feasible schedule for the model UET-LCT. Consider two tasks $i$ and $j$
such that $(i, j) \in E$. We denote by $\pi_i$ (resp. $\pi_j$) the processor on which the task $i$ (resp. the task $j$) is executed in the schedule $\sigma^\infty$. Moreover, we denote by $\pi'_i$ (resp. $\pi'_j$) the processor on which the task $i$ (resp. the task $j$) is executed in the schedule $\sigma^\infty$. Thus,

- If $\pi_i = \pi_j$ then $\pi'_i = \pi'_j$. Since the solution given by Munier and König [31] gives a feasible schedule on the model UET-UCT, then we have $t_i + 1 \leq t_j$, $\frac{2}{c+1}t_i^c + 1 \leq t_j^c$; $t_i^c + 1 \leq t_j^c + \frac{c+1}{2}$.

- If $\pi_i \neq \pi_j$ then $\pi'_i \neq \pi'_j$. We have $t_i + 1 + 1 \leq t_j$, $\frac{2}{c+1}t_i^c + 2 \leq \frac{2}{c+1}t_j^c$; $t_i^c + (c + 1) \leq t_j$.

\[ \square \]

**Theorem 1.3.4** An expansion algorithm gives a $\frac{2(c+1)}{3}$-approximation algorithm for the problem $\bar{P}|\text{prec}; c_i = c \geq 2; p_i = 1|C_{\text{max}}$.

**Proof**

We denote by $C_{\text{max}}^h$ (resp. $C_{\text{max}}^{opt}$) the makespan of the schedule computed by the Munier and König (resp. the optimal value of a schedule $\sigma^\infty$). In the same way we denote by $C_{\text{max}}^c$ (resp. $C_{\text{max}}^{opt,c}$) the makespan of the schedule computed by our algorithm (resp. the optimal value of a schedule $\sigma^\infty$).

We know that $C_{\text{max}}^h \leq \frac{4}{3}C_{\text{max}}^{opt}$. Thus, we obtain $\frac{C_{\text{max}}^c}{C_{\text{max}}^{opt,c}} = \frac{\frac{c+1}{2}C_{\text{max}}^h}{C_{\text{max}}^{opt}} \leq \frac{c+1}{3}$.\[ \□\]

This expansion method can be used for other scheduling problems.

### 1.4 Complexity and approximation of hierarchical scheduling model

On negative side, Bampis et al. in [6] studied the impact of the hierarchical communications on the complexity of the associated problem. They considered the simplest case, i.e., the problem $\bar{P}(P2)|\text{prec}; (c_i, e_i) = (1, 0); p_i = 1|C_{\text{max}}$, and they showed that this problem did not possess a polynomial-time approximation algorithm with a ratio guarantee better than $5/4$ (unless $\mathcal{P} = \mathcal{NP}$). Recently, [19] Giroudeau proved that there is no hope to find a $p$-approximation with $p < 6/5$ for the couple of communication delays $(c_i, e_i) = (2, 1)$. If duplication is allowed, Bampis et al. [5] extended the result of [13] in the case of hierarchical communications, providing an optimal algorithm for $\bar{P}(P2)|\text{prec}; (c_i, e_i) = (1, 0); p_i = 1;\text{dup}|C_{\text{max}}$. These complexity results are given in Table 1.1.
SCHEDULING WITH COMMUNICATION DELAYS

<table>
<thead>
<tr>
<th>Lower bound</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>((c_{ij}, \epsilon_{ij}))</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>((1, 0))</td>
<td>(\rho \geq 5/4)</td>
</tr>
<tr>
<td>((2, 1))</td>
<td>(\rho \geq 6/5)</td>
</tr>
<tr>
<td>((c, c'))</td>
<td>(\rho \geq 1 + \frac{1}{c+3})</td>
</tr>
</tbody>
</table>

Table 1.1: Previous complexity results for unbounded number of machines for hierarchical communication delay model.

On positive side, the authors presented in [8] a 8/5-approximation algorithm for the problem \(P(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\text{max}}\) 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 \(P(Pl)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\text{max}}\) leading to a \(\frac{4}{2c+1}\)-approximation algorithm.

Theorem 1.4.1 The problem of deciding whether an instance of \(P(Pl \geq 4)|\text{prec};...
In the same way as the section 1.2.2, the aim is to model the problem \( P_{l} \) \( \in \mathcal{NP} \) in \( \mathcal{P} \)-time since \( l_{0} \Gamma_{j} + \Gamma_{j}^{'} \).

The problem of deciding whether an instance of \( \bar{P}(Pl \geq 4)| prec; (c_{ij}, \epsilon_{ij}) = (c > 0, c'); p_{i} = 1|C_{max} \) having a schedule of length at most \( (c + 1) \) is \( \mathcal{NP} \)-complete, see [20].

**Corollary 1.4.1** There is no polynomial-time algorithm for the problem \( \bar{P}(Pl \geq 4)| prec; (c_{ij}, \epsilon_{ij}) = (c > 0, c'); p_{i} = 1|C_{max} \) with \( c > c' \) performance bound smaller than \( 1 + \frac{1}{c+3} \) unless \( \mathcal{P} \neq \mathcal{NP} \), see [20].

The problem of deciding whether an instance of \( \bar{P}(Pl \geq 4)| prec; (c_{ij}, \epsilon_{ij}) = (c > 0, c'); p_{i} = 1|C_{max} \) having a schedule of length at most \( (c + 1) \) is solvable in polynomial time since \( l \) and \( c \) are constant.

In the same way as the section 1.2.2, the aim is to model the problem \( \bar{P}(P2)| prec; (c_{ij}, \epsilon_{ij}) = (1, 0); p_{i} \geq 1|C_{max} \) by an integer linear program (ILP) denoted, in what follows, by \( \Pi \).

In this section, we will precis only the difference between the ILP given for the problem \( \bar{P}| prec; c_{ij} = 1; p_{i} = 1|C_{max} \) and \( \bar{P}(P2)| prec; (c_{ij}, \epsilon_{ij}) = (1, 0); p_{i} \geq 1|C_{max} \).

In every feasible schedule, every task \( i \in V - U \) has at most two successors, w.l.o.g. call them \( j_{1} \) and \( j_{2} \in \Gamma^{+}(i) \), that can be performed by the same cluster as \( i \) at time \( t_{j_{1}} = t_{j_{2}} = t_{i} + p_{i} \). The other successors of \( i \), if any, satisfy: \( \forall k \in \Gamma^{+}(i) - \{j_{1}, j_{2}\}, t_{k} \geq t_{i} + p_{i} + 1 \). Consequently, the constraints: \( \sum_{j \in \Gamma^{+}(i)} x_{ij} \geq |\Gamma^{+}(i) - 2 \) are added.

Similarly, every task \( i \in V - Z \) has at most two predecessors, w.l.o.g. call them \( j_{1} \) and \( j_{2} \in \Gamma^{-}(i) \), that can be performed by the same cluster as \( i \) at times \( t_{j_{1}}, t_{j_{2}} \) satisfying \( t_{i} - (t_{j_{1}} + p_{j_{1}}) < 1 \) and \( t_{i} - (t_{j_{2}} + p_{j_{2}}) < 1 \). So, the following constraints: \( \sum_{j \in \Gamma^{-}(i)} x_{ji} \geq |\Gamma^{-}(i)| - 2 \) are added.

The above constraints are necessary but not sufficient conditions in order to get a feasible schedule for the problem. For instance, a solution minimizing \( C_{max} \) for the graph of case (a) in Figure 1.6 will assign to every arc the value 0. However, since every cluster has two processors, and so at most two tasks can be processed on the same cluster simultaneously, the obtained solution is clearly not feasible.

Thus, the relaxation of the integer constraints, by considering \( 0 \leq x_{ij} \leq 1 \), and the resolution of the resulting linear program with objective function the minimization of \( C_{max} \), gives just a lower bound of the value of \( C_{max} \).

In order to improve this lower bound, we consider every sub-graph of \( G \) that is isomorphic to the graphs given in Figure 1.6 –cases (a) and (b). It is easy to see that in any feasible schedule of \( G \), at least one of the variables associated to the arcs of each one of these graphs must be set to one. So, the following constraints are added:
SCHEDULING WITH COMMUNICATION DELAYS

Figure 1.6: Special sub-graphs considered in the ILP.

- For the case (a):
  \[ \forall i, j, k, l, m \in V, \text{ such that } (j, i), (j, k), (l, k), (l, m) \in E, x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \]

- For the case (b):
  \[ \forall i, j, k, l, m \in V, \text{ such that } (i, j), (k, j), (k, l), (m, l) \in E, x_{ij} + x_{kj} + x_{kl} + x_{ml} \geq 1 \]

Thus, in what follows, the following ILP will be considered:

\[
\begin{align*}
\min \ C_{\max} \\
& x_{ij} \in \{0, 1\} \\
& t_i \geq 0 \\
& t_i + p_i + x_{ij} \leq t_j \\
& \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2 \\
& \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2 \\
& \forall i, j, k, l, m \in V, \ (j, i), (j, k), (l, k), (l, m) \in E, \\
& \forall i, j, k, l, m \in V, \ (i, j), (k, j), (k, l), (m, l) \in E, \\
& \forall i \in V,
\end{align*}
\]

Once again the integer linear program given above does not always imply a feasible solution for the scheduling problem. For instance, if the precedence graph given in Figure 1.7 is considered, the optimal solution of the integer linear program will set all the arcs to 0. Clearly, this is not a feasible solution for our scheduling problem. However, the goal in this step is to get a good lower bound of the makespan and a solution–eventually not feasible–that we will transform to a feasible one.
Let $\Pi^{inf}$ denote the linear program corresponding to $\Pi$ in which we relax the integrability constraints $x_{ij} \in \{0, 1\}$ by setting $x_{ij} \in [0, 1]$. Given that the number of variables and the number of constraints are polynomially bounded, this linear program can be solved in polynomial time. The solution of $\Pi^{inf}$ will assign to every arc $(i, j) \in E$ a value $x_{ij} = e_{ij}$ with $0 \leq e_{ij} \leq 1$ and will determine a lower bound of the value of $C_{max}$ that we denote by $\Theta^{inf}$.

**Lemma 1.4.1** $\Theta^{inf}$ is a lower bound on the value of an optimal solution for $\bar{P}(P2)|\text{prec}; (e_{ij}, \epsilon_{ij}) = (1, 0); p_i \geq 1|C_{max}$.

**Proof**
See the proof of Theorem 1.2.1.

We use the algorithm 1 for the rounding algorithm by changing the value rounded: $e_{ij} < 0.25$ instead $e_{ij} < 0.5$ The solution given by Step 1 is not necessarily a feasible solution (take for instance the precedence graph of Figure 1.7), so we must transform it to a feasible one. Notice that the cases given in Figure 1.6 are eliminated by the linear program. In the next step we need the following definition.

**Definition 1.4.1** A critical path with terminal vertex $i \in V$ is the longest path from an arbitrary source of $G$ to task $i$. The length of a path is defined as the sum of the processing times of the tasks belonging to this path and of the values $x_{ij}$ for every arc in the path.

1. **Step 2** [Feasible Rounding]: We change the integer solution as follows:

   (a) If $i$ is a source then we keep unchanged the values of $x_{ij}$ obtained in Step 1.

   (b) Let $i$ be a task such that all predecessors are already examined. Let $A_i$ be the subset of incoming arcs of $i$ belonging to a critical path with terminal vertex the task $i$. 

   (c) For each arc $(i, j)$ in $A_i$ we change the value $x_{ij}$ to $0$.

   (d) For each arc $(i, j)$ not in $A_i$ we change the value $x_{ij}$ to $1$.

   (e) We set $x_{ij} = e_{ij}$ for each arc $(i, j)$.

   (f) We set $x_{ij} = 0$ for each arc $(i, j)$ not in $A_i$.

   (g) We set $x_{ij} = 1$ for each arc $(i, j)$ not in $A_i$.

   (h) We set $x_{ij} = e_{ij}$ for each arc $(i, j)$ not in $A_i$.

   (i) We set $x_{ij} = 0$ for each arc $(i, j)$ not in $A_i$.

   (j) We set $x_{ij} = 1$ for each arc $(i, j)$ not in $A_i$.
i. If the set \( A_i \) contains a 0-arc, then all the outcoming arcs \( x_{ij} \) take the value 1.

ii. If the set \( A_i \) does not contain any 0-arc (all the critical incoming arcs are valued to 1), then the value of all the outcoming arcs \( x_{ij} \) remains the same as in Step 1, and all the incoming 0-arcs are transformed to 1-arcs.

In Step 1(b)ii changing the value of an incoming 0-arc to 1 does not increase the length of any critical path having as terminal vertex \( i \), because it exists at least one critical path with terminal vertex \( i \) such that an arc \( (j, i) \in E \) is valued by the linear program to at least 0.25 \( (e_{ji} \geq 0.25) \), and so \( x_{ji} \) is already equal to 1.

Lemma 1.4.2 Every job \( i \in V \) has at most two successors (resp. predecessors) such that \( e_{ij} < 0.25 \) (resp. \( e_{ji} < 0.25 \)) and The scheduling algorithm described above provides a feasible schedule.

Theorem 1.4.2 The relative performance \( \rho^h \) of our heuristic is bounded above by \( \frac{\hat{\rho}}{\beta} \) and the bound is tight, see [7].

Proof
See the proof of the Theorem 1.2.3.

\[ \square \]

1.5 Duplication

The duplication of the tasks has been introduced first by Papadimitriou and Yannakakis [32] in order to reduce an influence of the communication delays on the schedule. In [32], the authors develop a 2-approximation algorithm for the problem \( \bar{P} | \text{prec}; c_{ij} = c \geq 2; p_i = 1; \text{dup} | C_{\text{max}} \). The problem \( \bar{P} | \text{prec}; \text{SCT} | C_{\text{max}} \) (the problem \( \bar{P} | \text{prec}; c_{ij} = 1; p_i = 1 | C_{\text{max}} \) is a subproblem of \( \bar{P} | \text{prec}; \text{SCT} | C_{\text{max}} \)) becomes easy. In the following, we will describe the procedure. We may assume w.l.o.g. that all the copies of any task \( i \in V \) start their execution at the same time, call it \( t_i \).

1.5.1 Colin-Chrétienne Algorithm see [13]

The algorithm uses two steps: the first step computes the release times, and the second step use a critical determined from the first step in order to produces a
optimal schedule in which all the tasks and their copies are executed at their release times.

![Graph Diagram]

**Figure 1.8**: $P_0$ problem.

The $P_0$ problem given by Figure 1.8 will be illustrated the algorithm. The algorithm which computes the release times is given next:

**Algorithm 3** Release date algorithm and Earliest schedule

```
for i := 1 to n do
    if PRED(i) = ∅ then
        $b_i := 0$
    else
        $C := \max\{b_k + p_k + c_{ki} \mid k \in PRED(i)\}$;
        Let be $s$ such that: $b_s + p_s + c_{si} = C$;
        $b_i := \max\{b_s + p_s, \max\{b_k + p_k + c_{ki} \mid k \in PRED(i) - \{s\}\}\}$.
    end if
end for
```

Each connected component $G_c = (V; E_c)$ on different processor;
Each copy is executed at his release time.

Without lost of generality, all copies of the task $i$ admit the same starting , denoted by $t_i$, as the the task $i$. A arc $(i, j) \in E$ is a critical arc if $b_i + p_i + c_{ij} > b_j$. From this definition, it is clear that if $(i, j)$ is a critical arc, then in all as soon as possible schedule , each copy of a task $j$ must be preceded by a copy of a task $i$ on the same processor. In order to construct a earliest schedule, each critical path is allotted on a processor, and each copy is executed at his release date.

**Theorem 1.5.1** Let be $b_i$ the starting time computed by the procedure. For all feasible schedule for a graph $G$, the release date of a task $i$ cannot be less than $b_i$.
All sub-graph is spanning forest. The procedure gives a feasible schedule and the overall complexity is $O(n^2)$.
Table 1.2: Complexity and approximation results in presence of duplication

![Figure 1.9: The critical sub-graph of $P_0$.](image)

An earliest schedule of the precedence graph $P_0$ is given by Figure 1.10.

![Figure 1.10: An earliest schedule of $P_0$.](image)

The study of duplication in presence of unbounded number of processors is theoretical. Indeed, the results on unbounded processors do not improved the results on limited number of processors. So, concerning the hierarchical model, since the number of processors per cluster is limited, the authors in [5] are investigate only on the theoretical aspect of associated scheduling problem.
Job Shop and Machine Scheduling

<table>
<thead>
<tr>
<th>Lower bound</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>((c_{ij}, \epsilon_{ij}))</td>
<td>(C_{max})</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>(\rho \geq 9/8)</td>
</tr>
<tr>
<td>(c, c)</td>
<td>(\rho \geq 1 + \frac{1}{2c+5})</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>(\rho \geq 7/6)</td>
</tr>
<tr>
<td>(2, 1)</td>
<td>(\rho \geq 9/8)</td>
</tr>
<tr>
<td>(c, c')</td>
<td>(\rho \geq 1 + \frac{1}{2c+4})</td>
</tr>
</tbody>
</table>

Table 1.3: Threshold for the total sum of completion time minimization of unbounded number of machines

1.6 Total sum of completion time minimization

In this section, a threshold for total sum of completion time minimization problem is presented for some problems in the homogeneous and hierarchical model. The following table summarize all the results in the homogeneous communication delay model and the hierarchical communication delay model.

**Theorem 1.6.1** There is no polynomial-time algorithm for the problem \(\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1| \sum_j C_j\) with performance bound smaller than 9/8 unless \(\mathcal{P} \neq \mathcal{NP}\) see [24].

**Proof**

We suppose that there is a polynomial-time approximation algorithm denoted by \(A\) with performance guarantee bound smaller than \(1 + \frac{1}{8}\). Let \(I\) be the instance of the problem \(\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1| C_{max}\) obtained by a reduction (see Theorem 1.2.2).

Let \(I'\) be the instance of the problem \(\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1| \sum_j C_j\) by adding \(x\) new tasks from an initial instance \(I\). In the precedence constraints, each group of \(x\) (with \(x > \frac{36+6\rho n}{9-8\rho}\)) new tasks is a successor of the old tasks (old tasks are from the polynomial transformation used for the proof of Theorem 1.2.2). We obtain a complete directed graph from old tasks to new tasks.

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

1. If \(A(I') < 8\rho x + 6\rho n\) then \(A^*(I') < 8\rho x + 6\rho n\). So we can decide that there exists a scheduling of an instance \(I\) with \(C_{max} \leq 6\). Indeed, we suppose that at most one (denoted by \(i\)) task of \(n\) old tasks is executed at \(t = 6\). Among the \(x\) new tasks, at most one task may be executed on the same processor as
i before \( t = 9 \). Then \( A^*(I') > 9(x - 1) \). Thus, \( x < \frac{9 + 6\rho n}{9 - 8\rho} \). A contradiction with \( x > \frac{36 + 6\rho n}{9 - 8\rho} \). Thus, it exists a schedule of length 6 on an old tasks.

2. We suppose that \( A(I') \geq 8\rho x + 6\rho n \). So, \( A^*(I') \geq 8x + 6n \) because an algorithm \( A \) is a polynomial-time approximation algorithm with performance guarantee bound smaller than \( \rho < \frac{9}{8} \). There is no algorithm to decide whether the tasks from an instance \( I \) admit a schedule of length equal or less than 6.

Indeed, if there exists such an algorithm, by executing the \( x \) tasks at time \( t = 8 \), we obtain a schedule with a completion time strictly less than \( 8x + 6n \) (there is at least one task which is executed before the time \( t = 6 \)). This is a contradiction since \( A^*(I') \geq 8x + 6n \).

This concludes the proof of Theorem 1.6.1.

\[ \square \]

1.7 Conclusion

Figure 1.11: Principal results in UET-UCT model for the minimization of the length of the schedule

25
1.8 Appendix

In this section, we will give some fundamentals results in theory of complexity and approximation with guaranteed performance. A classical method in order to obtain a lower for none approximation algorithm is given by the following results called "Impossibility theorem" [14] and gap technic see [3].

Theorem 1.8.1 (Impossibility theorem) Consider a combinatorial optimization problem for which all feasible solutions have non-negative integer objective function value (in particular scheduling problem). Let \( c \) be a fixed positive integer. Suppose that the problem of deciding if there exists a feasible solution of value at most \( c \) is \( \text{NP} \)-complete. Then, for any \( \rho < (c + 1)/c \), there does not exist a polynomial-time \( \rho \)-approximation algorithm \( A \) unless \( \mathcal{P} = \mathcal{NP} \), see ([14], [3]).

Theorem 1.8.2 (The gap technic) Let \( Q' \) be an \( \text{NP} \)-complete decision problem and let \( Q \) be an \( \text{NPO} \) minimization problem. Let us suppose that there exist two polynomial-time computable functions \( f : I_{Q'} \rightarrow I_Q \) and \( d : I_{Q'} \rightarrow \mathbb{N} \) and a constant \( \text{gap} > 0 \) such that, for any instance \( x \) of \( Q' \),

\[
S^*(f(x)) = \begin{cases} 
    d(x) \\
    d(x)(1 + \text{gap}) 
\end{cases}
\]

Then no polynomial-time \( r \)-approximate algorithm for \( Q \) with \( r < 1 + \text{gap} \) can exist, unless \( \mathcal{P} = \mathcal{NP} \), see [3].

1.8.1 List of \( \text{NP} \)-complete problems

In this section, some classical \( \text{NP} \)-complete problems are listed which are used in this chapter for the polynomial-time transformation.

One-in-(2, 3)\( SAT(2, \bar{1}) \) problem

Instances: We consider a logic formula with clauses of size two or three, and each positive literal (resp. negative literal) occurs twice (resp. once). The aim is to find exactly one true literal per clause. Let \( n \) be a multiple of 3 and let \( C \) be a set of clauses of size 2 or 3. There are \( n \) clauses of size 2 and \( n/3 \) clauses of size 3 so that:

- each clause of size 2 is equal to \( (x \lor \bar{y}) \) for some \( x, y \in V \) with \( x \neq y \).
- each of the \( n \) literals \( x \) (resp. of the literals \( \bar{x} \)) for \( x \in V \) belongs to one of the \( n \) clauses of size 2, thus to only one of them.
each of the $n$ literals $x$ belongs to one of the $n/3$ clauses of size 3, thus to only one of them.

- whenever $(x \lor \overline{y})$ is a clause of size 2 for some $x, y \in \mathcal{V}$, then $x$ and $y$ belong to different clauses of size 3.

We would insist on the fact that each clause of size three yields six clauses of size two.

**Question:**
Is there a truth assignment for $I : \mathcal{V} \rightarrow \{0, 1\}$ such that every clause in $\mathcal{C}$ has exactly one true literal?

**Clique problem**

**Instances:** Let be $G = (V, E)$ a graph and $k$ a integer.

**Question:** There is a clique (a complete sub-graph) of size $k$ in $G$?

**3-SAT problem**

**Instances:**

- Let be $\mathcal{V} = \{x_1, \ldots, x_n\}$ a set of $n$ logical variables.
- Let be $\mathcal{C} = \{C_1, \ldots, C_m\}$ a set of clause of length three: $(x_{c_i} \lor y_{c_i} \lor z_{c_i}).$

**Question:** There is $I : \mathcal{V} \rightarrow \{0, 1\}$ a assignment

### 1.8.2 Ratio of approximation algorithm

This value is defined as the maximum ratio, on all instances $I$, between maximum objective value given by algorithm $h$ (denoted by $K^h(I)$) and the optimal value (denoted by $K^{opt}(I)$), i.e.

$$\rho^h = \max_I \frac{K^h(I)}{K^{opt}(I)}.$$

Clearly, we have $\rho^h \geq 1$.

### 1.8.3 Notations

The notations of this chapter will precised by using the three fields notation scheme $\alpha|\beta|\gamma$, proposed by Graham et al. [23]:

27
• \( \alpha \in \{ P, \bar{P}, \bar{P}(P2) \} \)
  – If \( \alpha = P \) the number of processors is limited,
  – If \( \alpha = \bar{P} \), then the number of processors is not limited,
  – If \( \alpha = P(P2) \), then we have unbounded number of clusters constituted
    by two processors each,

• \( \beta = \beta_1 \beta_2 \beta_3 \beta_4 \) where:
  – If \( \beta_1 = \text{prec} \) (the precedence graph is a sequence).
  \( \beta_2 \in \{ c \} \)
  – If \( \beta_2 = c \) (the communication delay between tasks admitting a precedence constraint is equal to \( c \)).
  \( \beta_3 \in \{ p_j \} \)
  – If \( \beta_3 = p_j = 1 \) (the processing time of all the tasks is equal to one).
  \( \beta_4 \in \{ \text{dup}, . \} \)
  – If \( \beta_4 = \text{dup} \) (the duplication of task is allowed)
  – Si \( \beta_4 = . \) (the duplication of task is not allowed)

• \( \gamma \) is the objective function:
  – the minimization of the makespan, denoted by \( C_{max} \)
  – the minimization of the total sum of completion time, denoted by
    \( \sum_j C_j \) where \( C_j = t_j + p_j \).
Bibliography


SCHEDULING WITH COMMUNICATION DELAYS


