Scheduling with Communication Delay

Rodolphe Giroudeau, Jean-Claude König

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 Π^i) that are used to process n precedence-constrained tasks. Each machine Π^i (cluster) comprises several identical parallel processors (denoted by π^i_k). 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 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 Π^i and Π^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 Π^i but on different processors π^i_k and $\pi^j_{k'}$ (with $k \neq k'$) then 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 (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}, ϵ_{ij}) will be associated with $\epsilon_{ij} \leq c_{ij}\ \forall (i, j) \in E$ such that :

- if $\Pi^i = \Pi^j$ and if $\pi^i_k = \pi^j_k$ then $t_i + p_i \leq t_j$
- else if $\Pi^i = \Pi^j$ and if $\pi^i_k \neq \pi^j_{k'}$ 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_{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 $c \geq c'$.

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 $c_{ij} = \epsilon_{ij}$. 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 $\mathcal{NP}-$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 σ 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 $\bar{P}|prec, p_i = 1, c_{ij} = 1|C_{max}$ problem has a schedule of length 5 is polynomial, see [42].*

Proof
The proof is based on the notion of total unimdularity matrix, see [42] and see [39].
□

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

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

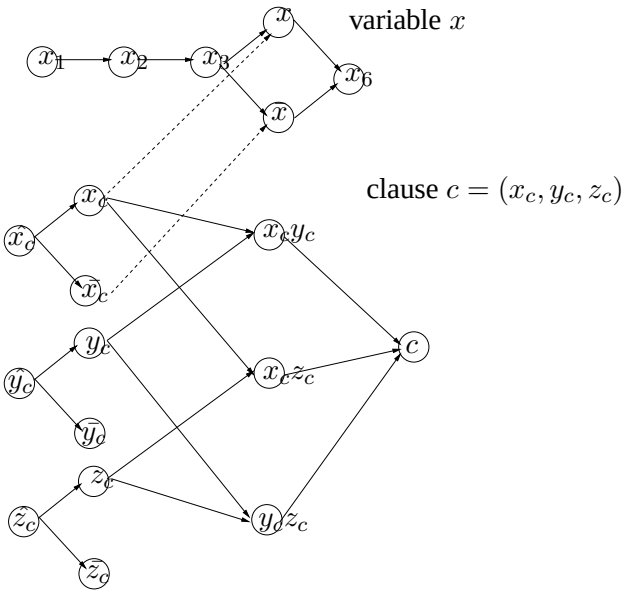


Figure 1.1: *The variables-tasks and the clauses-tasks*

It is clear that the problem is in $\mathcal{NP}$.
Let be π^* an instance of $3SAT$ problem, we construct an instance π of the problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ in the following way:

- For each variable x, six tasks are introduced : $x_1,\ x_2,\ x_3,\ x,\ \bar{x}$ 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:

$\hat{x_c}$, $\hat{y_c}$, $\hat{z_c}$, x_c, $\bar{x_c}$, y_c, $\bar{y_c}$, z_c, $\bar{z_c}$, x_cy_c, y_cz_c, x_cz_c and c: 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 $I : \mathcal{V} \rightarrow \{0, 1\}$ such that each clause in $\mathcal{C}$ has at least one true literal.

□

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

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

□

1.2.2 Approximate solutions with guaranteed performance

Good approximation algorithms seem to be 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 $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$. This algorithm is based on a formulation on a 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 $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ by an integer linear program (ILP) denoted, in what follows, by Π.
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_{max} the makespan of the schedule, $\forall i \in V, t_i + p_i \leq C_{max}$
Thus, in what follows,the following ILP will be considered:

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

Let Π^{inf} denote the linear program corresponding to Π 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 Π^{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 Θ^{inf}.

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

Proof This is true since any optimal feasible solution of the scheduling problem must satisfy all the constraints of the integer linear program Π. □

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_I^{inf}

$$\begin{cases} \text{si } e_{ij} < 0.5 & \Longrightarrow \quad x_{ij} = 0 \\ \text{si } e_{ij} \geq 0.5 & \Longrightarrow \quad x_{ij} = 1 \end{cases}$$

Step 1 [**Computation of starting time**]
if $i \in Z$ **then**
$t_i = 0$
else
$t_i = \max\{t_j + 1 + x_{ji}\}$ avec $j \in \Gamma^-(i)$ et $(j,i) \in A_i$
end if
Step 2 [**Construction of the schedule**]
Let be $G' = (V; E')$ where $E' = E \backslash \{(i,j) \in E | x_{ij} = 1\}$ {G' is generated by the $0 - arcs$.}
Allotted each connected component of G' on a different processor.
Each task is executed at it 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 ech task i admits at most one incoming (resp. outcoming) 0-arcs.

□

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

Proof Let be a path $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{k+1}$ consitutued 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]).

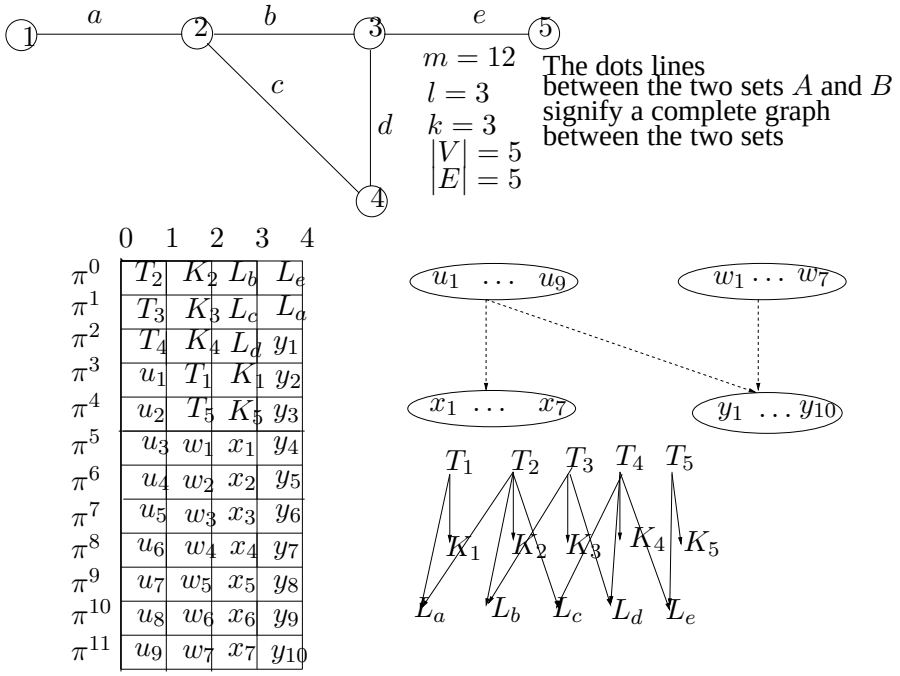
1.2.3 Bounded number of processors

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

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

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

Proof
The proof is based on the $\mathcal{NP}$-complete problem **Clique**.



	0	1	2	3	4
π^0	T_2	K_2	L_b	L_e	
π^1	T_3	K_3	L_c	L_a	
π^2	T_4	K_4	L_d	y_1	
π^3	u_1	T_1	K_1	y_2	
π^4	u_2	T_5	K_5	y_3	
π^5	u_3	w_1	x_1	y_4	
π^6	u_4	w_2	x_2	y_5	
π^7	u_5	w_3	x_3	y_6	
π^8	u_6	w_4	x_4	y_7	
π^9	u_7	w_5	x_5	y_8	
π^{10}	u_8	w_6	x_6	y_9	
π^{11}	u_9	w_7	x_7	y_{10}	

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

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 $clique \propto P|prec, c_{ij} = 1, p_i = 1|C_{max}$. Let be π^* a instance of the $clique$ problem. An instance π of $P|prec, c_{ij} = 1, p_i = 1|C_{max}$ problem is constructed in the following way:

- $\forall v \in V$ the tasks $T_v,\ K_v$ are introduced,

- $\forall e \in E$ a task L_e is created.
  - We add the following precedence constraints: $T_v \rightarrow K_v, \forall v \in V$ and $T_v \rightarrow L_e$ if v is an endpoint of e.
- Four sets of tasks are introduced:
  - $X_x = \{x = 1, \ldots, x = m - l - |V| + k\}$,
  - $Y_y = \{y = 1, \ldots, y = m - |E| + l\}$,
  - $U_u = \{u = 1, \ldots, u = m - k\}$,
  - $W_w = \{w = 1, \ldots, w = m - |V|\}$.
  - the precedence constraints are added: $U_u \rightarrow X_x$, $U_u \rightarrow Y_y$, $W_w \rightarrow Y_y$

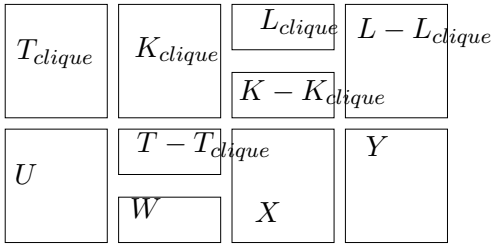


Figure 1.3: *Example of construction in order to illustrate the proof of theorem 1.2.5*

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 $\bar{P}|prec, c_{ij} = 1, p_i = 1|C_{max}$. 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 guarantee ρ for the problem $\bar{P}|prec, c_{ij} = 1, p_i = 1|C_{max}$, we may obtain a*

Algorithm 2 Scheduling on m machines from a schedule σ^{∞} on unbounded number of processors

for $i = 0$ à $C^{\infty}_{max} - 1$ **do**
  Let be X_i the set of tasks executed at t_i in σ^{∞} using a heuristic h^*.
  The X_i tasks are executed in $\lceil \frac{|X_i|}{m} \rceil$ units of time.
end for

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

Proof

$$\begin{aligned} C^{m,A}_{max} &\leq \sum_{i=0}^{C^{\infty}_{max}-1} \lceil \frac{|X_i|}{m} \rceil \leq \sum_{i=0}^{C^{\infty}_{max}-1} (\lfloor \frac{|X_i|}{m} \rfloor + 1) \leq \sum_{i=0}^{C^{\infty}_{max}-1} (\lfloor \frac{|X_i|}{m} \rfloor) + C^{\infty}_{max} \\ &\leq \sum_{i=0}^{C^{\infty}_{max}-1} (\frac{|X_i|}{m}) + C^{\infty}_{max} \leq C^{opt,m}_{max} + C^{\infty,h^*}_{max} \leq C^{opt,m}_{max} + \rho C^{opt,m}_{max} \end{aligned}$$

□

For example, the $4/3$-approximation algorithm gives a $7/3$-approximation algorithm. Munier et al. [29] propose a $(7/3 - 4/3m)$-approximation algorihtm 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 $P|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ for $C_{\max} = c + 3$ is an $\mathcal{NP}$-complete problem, and for $C_{\max} = c + 2$ (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 $Balanced\ Bipartite\ Complete\ Graph,\ BBCG$ [17, 38]. Thus, Bampis et al. [4] proved that the $P|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ problem does not pos-

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

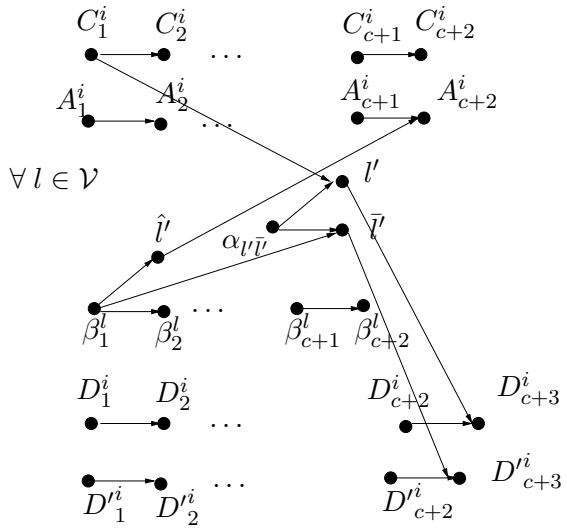


Figure 1.4: *A partial precedence graph for the $\mathcal{NP}$-completeness of the scheduling problem $\bar{P}|prec; c_{ij} = c \geq 3; p_i = 1|C_{max}$.*

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

Proof
It is easy to see that $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max} = c + 4 \in \mathcal{NP}$.
The proof is based on a reduction from Π_1. Given an instance π^* of Π_1, we construct an instance π of the problem $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max} = c + 4$, in the following way (Figure 1.4 helps understanding of the reduction):
n denotes the number of variables of π^*.

1. For all $l \in \mathcal{V}$, we introduce $(c + 6)$ variable-tasks: $\alpha_{l'\bar{l}'},\ l',\ \bar{l}',\ \hat{l}',\ \beta_j^l$ with $j \in \{1, 2, \ldots, c+2\}$. We add the precedence constraints: $\alpha_{l'\bar{l}'} \rightarrow l'$, $\alpha_{l'\bar{l}'} \rightarrow \bar{l}'$, $\beta_1^l \rightarrow \hat{l}'$, $\beta_1^l \rightarrow \bar{l}'$, $\beta_j^l \rightarrow \beta_{j+1}^l$ with $j \in \{1, 2, \ldots, c+1\}$.

2. For all clauses of length three denoted by $C_i = (y \vee z \vee t)$, we introduce $2 \times (2 + c)$ 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 $l \in \{\hat{y}', \hat{z}', \hat{t}'\}$.

3. For all clauses of length two denoted by $C_i = (x \vee \bar{y})$, we introduce $(c+3)$ clause-tasks D^i_j, $j \in \{1, 2, \ldots, c+3\}$ with precedence constraints: $D^i_j \rightarrow D^i_{j+1}$ with $j \in \{1, 2, \ldots, c+2\}$ and $l' \rightarrow D^i_{c+3}$ 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: $\bar{l}'$ is in the clause C' of length two associated with the path $D'^i_1 \rightarrow D'^i_2 \rightarrow \ldots D'^i_{c+2} \rightarrow D'^i_{c+3}$
It easy to see that there is a schedule of length equal or less than $(c+4)$ if only if there is a truth assignment $I : \mathcal{V} \rightarrow \{0, 1\}$ such that each clause in $\mathcal{C}$ 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 $\bar{P}|prec; c_{ij} = 2; p_i = 1|C_{max}$ 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 $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{max}$ with performance bound smaller than $1 + \frac{1}{c+4}$ unless $\mathcal{P} \neq \mathcal{NP}$ (see [22]).*

Theorem 1.3.3 *The problem of deciding whether an instance of $\bar{P}|prec; c_{ij} = c; p_i = 1|C_{max}$ with $c \in \{2, 3\}$ has a schedule of length at most $(c+2)$ is solvable in polynomial time (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 $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{\max}$ will be proposed.
Notation: We denote by σ^∞, the UET-UCT schedule, and by σ^∞_c the UET-LCT schedule. Moreover, we denote by t_i (resp. t^c_i) the starting time of the task i in the schedule σ^∞ (resp. in the schedule σ^∞_c).
Principle: We keep an assignment for the tasks given by a "good" feasible schedule on an unrestricted number of processors σ^∞. We proceed to an expansion of the makespan, while preserving communication delays ($t^c_j \geq t^c_i + 1 + c$) 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 σ^∞, 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, π) on the

schedule σ^{∞} corresponding to: t_i the starting time of the task i for the schedule σ^{∞} and π the processor on which the task i is processed at t_i.
Now, we determine a couple $\forall i \in V,\ (t_i^c, \pi')$ on schedule σ_c^{∞} in the following way: The starting time $t_i^c = d \times t_i = \frac{(c+1)}{2} t_i$ and, $\pi = \pi'$. The justification of the expansion coefficient is given below. An illustration of the expansion is given in Figure 1.5.

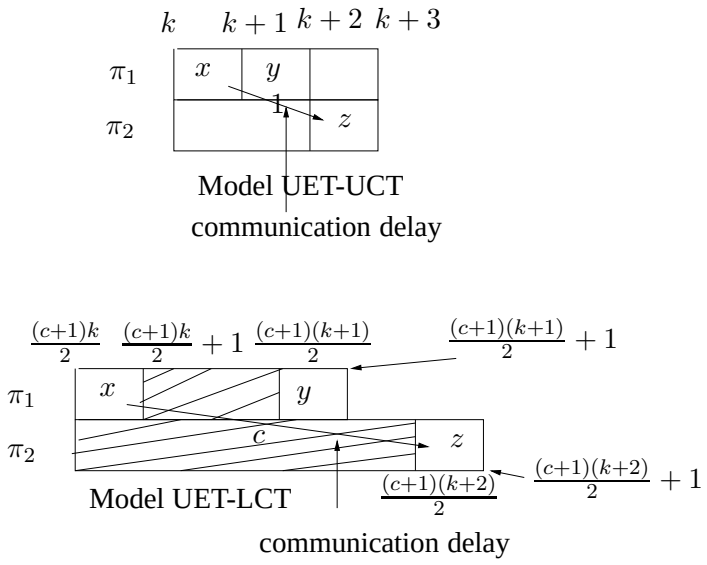


Figure 1.5: *Illustration of notion of an expansion*

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 σ^{∞}. Let be d a coefficient d such that $t_i^c = d \times t_i$ and $t_j^c = d \times t_j$. After an expansion, in order to respect the precedence constraints and the communication delays we must have $t_j^c \geq t_i^c + 1 + c$, and so $d \times t_i - d \times t_j \geq c + 1,\ d \geq \frac{c+1}{t_i - t_j},\ d \geq \frac{c+1}{2}$. It is sufficient to choose $d = \frac{(c+1)}{2}$. □

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

Proof It 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 π_i (resp. π_j) the processor on which the task i (resp. the task j) is executed in the schedule σ^∞. Moreover, we denote by π'_i (resp. π'_j) the processor on which the task i (resp. the task j) is executed in the schedule σ_c^∞. Thus,

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

□

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

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

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

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 $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{max}$, 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 ρ-approximation with $\rho < 6/5$ for the couple of communication delays $(c_{ij}, \epsilon_{ij}) = (2,1)$. If duplication is allowed, Bampis et al. [5] extended the result of [13] in the case of hierarchical communications, providing an optimal algorithm for $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1; dup|C_{max}$. These complexity results are given in Table 1.1.

	Lower bound	
(c_{ij}, ϵ_{ij})	C_{max}	References
$(1, 0)$	$\rho \geq 5/4$	see [6]
$(2, 1)$	$\rho \geq 6/5$	see [19]
(c, c')	$\rho \geq 1 + \frac{1}{c+3}$	see [20]

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 $\bar{P}(P2)|prec; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{max}$ which is based on an integer linear programming formulation. They relax the integrity constraints and they produce a feasible schedule by rounding. This result is extended to the problem $\bar{P}(Pl)|prec; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{max}$ leading to a $\frac{4l}{2l+1}$-approximation algorithm.
The challenge is to determinate a threshold for the approximation algorithm concerning the two more general problems: $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, 1); p_i = 1|C_{max}$ and $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ with $c' < c$.
In the classical scheduling communication delay model, we know that (see [25]) the decision problem associated with $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{\max}$ becomes $\mathcal{NP}$-complete even for $C_{\max} \geq 6$, and that it is polynomial for $C_{\max} \leq 5$ (this problem is denoted in what follows the UET-UCT (Unit Execution Time Unit Communication Time) homogeneous scheduling communication delays problem). Recently, in [22], the authors proved that there is no possibility of finding a ρ-approximation with $\rho < 1 + 1/(c+4)$ (unless $\mathcal{P} = \mathcal{NP}$) for the case where all tasks of the precedence graph have unit execution times, where the multiprocessor is composed of an unrestricted number of machines, and where c denotes the communication delay between two tasks i and j both submitted to a precedence constraint and which have to be processed by two different machines (this problem is denoted in the following UET-LCT (Unit Execution Time Large Communication Time) homogeneous scheduling communication delays problem). The problem becomes polynomial whenever the makespan is at most $(c + 1)$. The case of $(c + 2)$ is still partially opened. In the same way as for the hierarchical communication delay model, for the couple of communication delay values $(1, 0)$, the authors proved in [6] that there is no possibility of finding a ρ-approximation with $\rho < 5/4$ (this problem is detailed in following the UET-UCT hierarchical scheduling communication delay problem).

Theorem 1.4.1 *The problem of deciding whether an instance of* $\bar{P}(Pl \geq 4)|prec;$

$(c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ *having a schedule of length at most* $(c+3)$ *is* $\mathcal{NP}$*-complete, see [20].*

Corollary 1.4.1 *There is no polynomial-time algorithm for the problem* $\bar{P}(Pl \geq 4)|prec; (c_{ij}, \epsilon_{ij}) = (c, c'); p_i = 1|C_{max}$ *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 $\bar{P}(Pl)|prec; (c_{ij}, \epsilon_{ij}) = (c > 0, c'); p_i = 1|C_{max}$ 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 Π.

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 of $V - Z$ 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:

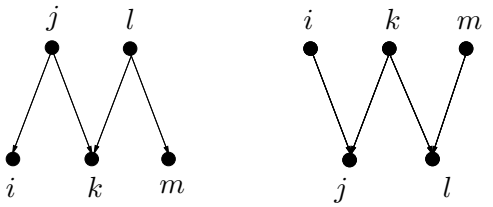


Figure 1.6: Special sub-graphs considered in the ILP.

- For the case (a):

  $\forall i,j,k,l,m \in V,$ 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):

  $\forall i,j,k,l,m \in V,$ 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:

$$(\Pi) \quad \begin{cases} & \min C_{max} \\ \forall (i,j) \in E, & x_{ij} \in \{0,1\} \\ \forall i \in V, & t_i \geq 0 \\ \forall (i,j) \in E, & t_i + p_i + x_{ij} \leq t_j \\ \forall i \in V - U, & \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2 \\ \forall i \in V - Z, & \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2 \\ \forall i,j,k,l,m \in V, \setminus (j,i),(j,k),(l,k),(l,m) \in E, & x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \\ \forall i,j,k,l,m \in V, \setminus (i,j),(k,j),(k,l),(m,l) \in E, & x_{ij} + x_{kj} + x_{kl} + x_{ml} \geq 1 \\ \forall i \in V, & t_i + p_i \leq C_{max} \end{cases}$$

Once again the integer linear program given above does not always imply a feasible solution for thescheduling 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.

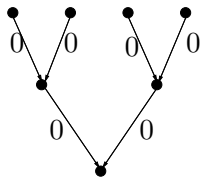


Figure 1.7: An optimal solution of the ILP Π does not always imply a feasible solution.

Let Π^{inf} denote the linear program corresponding to Π 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 Π^{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 Θ^{inf}.

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

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.

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 ρ^h of our heuristic is bounded above by $\frac{8}{5}$ and the bound is tight, see [7].*

Proof
See the proof of the Theorem 1.2.3.

□

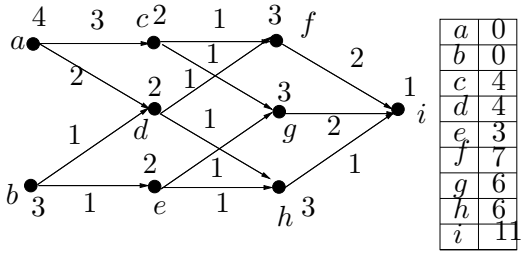
1.5 Duplication

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 developp a 2-approximation algorithm for the problem $\bar{P}|prec; c_{ij} = c \geq 2; p_i = 1; dup|C_{max}$. The problem $\bar{P}|prec; SCT|C_{max}$ (the problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ is a subproblem of $\bar{P}|prec; SCT|C_{max}$) 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 determinated 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.



a	0
b	0
c	4
d	4
e	3
f	7
g	6
h	6
i	11

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 $PRED(i) = \varnothing$ **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 $O(n^2)$.*

	Lower and Upper bound			
$\alpha\|(c_{ij},\epsilon_{ij})$	Lower bound	References	Upper bound	References
$P\|(1,1),dup$	$\rho \geq 5/4$	see [8]	2-approx	[30]
$\bar{P}\|(1,1),dup$	poly	see [13]	poly	see [13]
$P\|(c,c),dup$	$\rho \geq 1+\frac{1}{c+3}$	see [4]	3-approx	[40]
$\bar{P}\|(c,c),dup$	$\mathcal{NP}$-complete	see [32]	2-approx	[32]
$P(P2)\|(1,0),dup$	$\rho \geq 4/3$	see [2]		
$\bar{P}(P2)\|(1,0),dup$	poly	see [5]	poly	see [5]
$P(P2)\|(c,c),dup$	$\rho \geq 1+\frac{1}{c+3}$	see [20]		
$\bar{P}(P2)\|(c,c),dup$				

Table 1.2: Complexity and approximation results in presence of duplication

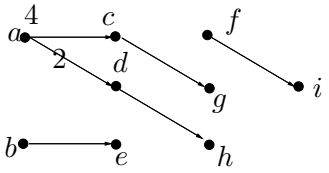


Figure 1.9: *The critical sub-graph of* P_0.

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

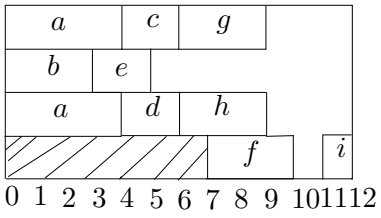


Figure 1.10: *An earliest schedule of* P_0.

The study of duplication in presence of unbounded number of processors is theoritical. Indeed, the results on unbounded processors do not improved the results on limited number of processors. So, concerning the hierachical model, since the number of processors per cluster is limited, the autors in [5] are investigate only on the theoritical aspect of associated scheduling problem.

	Lower bound	
(c_{ij}, ϵ_{ij})	C_{max}	References
$(1,1)$	$\rho \geq 9/8$	see [24]
(c,c)	$\rho \geq 1 + \frac{1}{2c+5}$	see [22]
$(1,0)$	$\rho \geq 7/6$	see [18]
$(2,1)$	$\rho \geq 9/8$	see [19]
(c,c')	$\rho \geq 1 + \frac{1}{2c+4}$	see [20, 21]

Table 1.3: Thresold for the total sum of completion time minimization of unbouned number of machines

1.6 Total sum of completion time minimization

In this section, a thresold 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* $\bar{P}|prec; c_{ij} = 1; p_i = 1|\sum_j C_j$ *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 $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{max}$ obtained by a reduction (see Theorem 1.2.2).
Let I' be the instance of the problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|\sum_j C_j$ 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 news 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 $\rho < 9/8$. 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.

□

1.7 Conclusion

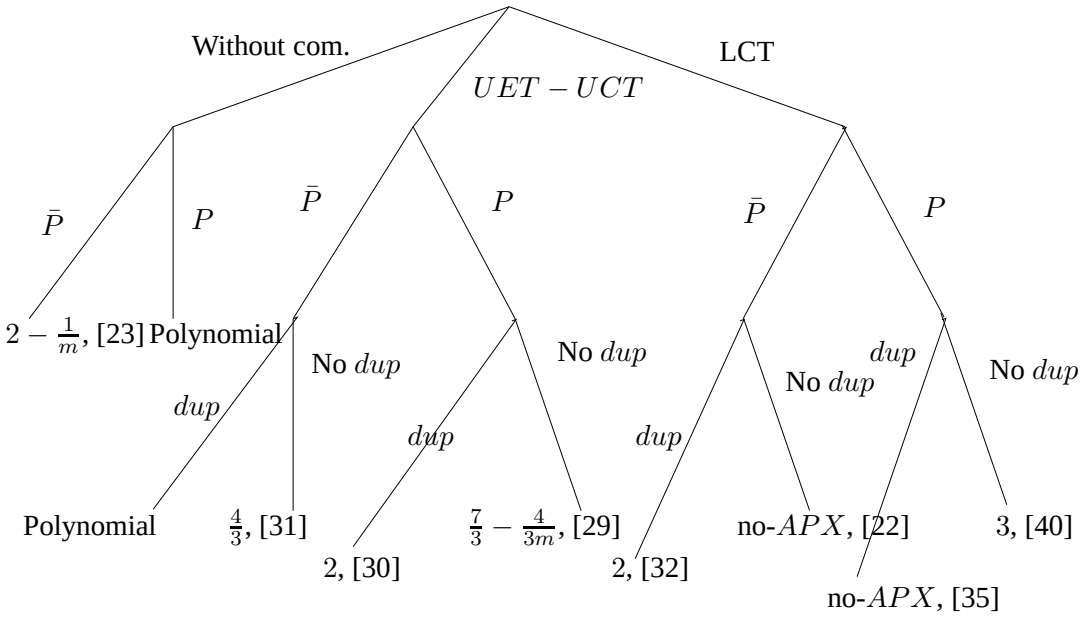


Figure 1.11: *Principal results in UET-UCT model for the minimization of the length of the schedule*

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 $\mathcal{NP}$-complete. Then, for any $\rho < (c+1)/c$, there does not exist a polynomial-time ρ-approximation algorithm A unless $\mathcal{P} = \mathcal{NP}$, see ([14], [3])*

Theorem 1.8.2 (The gap technic) *Let Q' be an $\mathcal{NP}$-complete decision problem and let Q be an 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 $gap > 0$ such that, for any instance x of Q',*

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

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

1.8.1 List of $\mathcal{NP}$-complete problems

In this section, some classical $\mathcal{NP}$-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 $\mathcal{C}$ 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 $(x \vee \bar{y})$ for some $x,\ y \in \mathcal{V}$ with $x \neq y$.
- each of the n literals x (resp. of the literals $\bar{x}$) for $x \in \mathcal{V}$ belongs to one of the n clauses of 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 $(x \vee \bar{y})$ 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: $(x_{c_i} \vee y_{c_i} \vee z_{c_i})$.

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 $\mathcal{K}^h(I)$) and the optimal value (denoted by $\mathcal{K}^{opt}(I)$), i.e.

$$\rho^h \quad = \quad \max_{I} \quad \frac{\mathcal{K}^h(I)}{\mathcal{K}^{opt}(I)}.$$

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]:

- $*\alpha \in \{P, \bar{P}, \bar{P}(P2)\}$
  - If $\alpha = P$ the number of processors is limited,
  - If $\alpha = \bar{P}$, then the number of processors is not limited,
  - If $\alpha = \bar{P}(P2)$, then we have unboued number of clusters constituted by two processors each,
- $\beta = \beta_1\beta_2\beta_3\beta_4$ where:
  - If $\beta_1 =$prec (the precedence graph est quleconsue.

  $*\beta_2 \in \{c\}$
  - If $\beta_2 = c$ (the communication delay betwen to tasks admiting 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).

  $*\beta_4 \in \{dup, .\}$
  - If $\beta_4 =$dup (the duplication of task is allowed)
  - Si $\beta_4 = .$ (the duplication of task is not allowed)
- γ is the objectif 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

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

[Angel et al., 2002] Angel, E., Bampis, E., and Giroudeau, R. (2002). Non-approximability results for the hierarchical communication problem with a bounded number of clusters. In B. Monien, R. F., editor, *EuroPar'02 Parallel Processing*, LNCS, No. 2400, pages 217–224. Springer-Verlag.

[Aussiello et al., 1999] Aussiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation*, chapter 3, pages 100–102. Springer.

[Bampis et al., 1996] Bampis, E., Giannakos, A., and König, J. (1996). On the complexity of scheduling with large communication delays. *European Journal of Operation Research*, 94:252–260.

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

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

[Bampis et al., 2003] Bampis, E., Giroudeau, R., and König, J. (2003). An approximation algorithm for the precedence constrained scheduling problem with hierarchical communications. *Theoretical Computer Science*, 290(3):1883–1895.

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

[Bhatt et al., 1997] Bhatt, S., Chung, F., Leighton, F., and Rosenberg, A. (1997). On optimal strategies for cycle-stealing in networks of workstations. *IEEE Trans. Comp.*, 46:545–557.

[Blayo et al., 1999] Blayo, E., Debreu, L., Mounie, G., and Trystram, D. (1999). Dynamic loab balancing for ocean circulation model adaptive meshing. In et al., P. A., editor, *Proceedings of Europar*, LNCS No. 1685, pages 303–312. Springer-Verlag.

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

[Chen et al., 1998] Chen, B., Potts, C., and Woeginger, G. (1998). A review of machine scheduling: complexity, algorithms and approximability. Technical Report Woe-29, TU Graz.

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

[Chrétienne and Picouleau, 1995] Chrétienne, P. and Picouleau, C. (1995). *Scheduling Theory and its Applications*. John Wiley & Sons. Scheduling with Communication Delays: A Survey, Chapter 4.

[Decker and Krandick, 1999] Decker, T. and Krandick, W. (1999). Parallel real root isolation using the descartes method. In *HiPC99*, volume 1745 of *LNCS*. Sringer-Verlag.

[Dutot and Trystram, 2001] Dutot, P. and Trystram, D. (2001). Scheduling on hierarchical clusters using malleable tasks. In *13th ACM Symposium of Parallel Algorithms and Architecture*, pages 199–208.

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

[Giroudeau, 2000] Giroudeau, R. (2000). *L'impact des délais de communications hiérarchiques sur la complexité et l'approximation des problèmes d'ordonnancement*. PhD thesis, Université d'Évry Val d'Essonne.

[Giroudeau, 2005] Giroudeau, R. (2005). Seuil d'approximation pour un problème d'ordonnancement en présence de communications hiérarchiques. *Technique et Science Informatique*, 24(1):95–124.

[Giroudeau and König, 2004] Giroudeau, R. and König, J. (2004). General non-approximability results in presence of hierarchical communications. In *Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pages 312–319. IEEE.

[Giroudeau and König, pted] Giroudeau, R. and König, J. (accepted). General scheduling non-approximability results in presence of hierarchical communications. *European Journal of Operational Research*.

[Giroudeau et al., 2005] Giroudeau, R., König, J., Moulaï, F., and Palaysi, J. (2005). Complexity and approximation for the precedence constrained scheduling problem with large communications delays. In J.C. Cunha, P. M., editor, *Proceedings of Europar*, LNCS, No. 3648, pages 252–261. Springer-Verlag.

[Graham et al., 1979] Graham, R., Lawler, E., Lenstra, J., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5:287–326.

[Hoogeveen et al., 1998] Hoogeveen, H., Schuurman, P., and Woeginger, G. (1998). Non-approximability results for scheduling problems with minsum criteria. In Bixby, R., Boyd, E., and Ríos-Mercado, R., editors, *IPCO VI*, Lecture Notes in Computer Science, No. 1412, pages 353–366. Springer-Verlag.

[Hoogeveen et al., 1994] Hoogeveen, J., Lenstra, J., and Veltman, B. (1994). Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137.

[Ludwig, 1995] Ludwig, W. T. (1995). *Algorithms for scheduling malleable and nonmalleable parallel tasks*. PhD thesis, University of Wisconsin-Madison, Department of Computer Sciences.

[Mounié, 2000] Mounié, G. (2000). *Efficient scheduling of parallel application : the monotic malleable tasks*. PhD thesis, Institut National Polytechnique de Grenoble.

[Mounié et al., 1999] Mounié, G., Rapine, C., and Trystram, D. (1999). Efficient approximation algorithm for scheduling malleable tasks. In *11th ACM Symposium of Parallel Algorithms and Architecture*, pages 23–32.

[Munier and Hanen, 1996] Munier, A. and Hanen, C. (1996). An approximation algorithm for scheduling unitary tasks on m processors with communication delays. Private communication.

[Munier and Hanen, 1997] Munier, A. and Hanen, C. (1997). Using duplication for scheduling unitary tasks on m processors with communication delays. *Theoretical Computer Science*, 178:119–127.

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

[Papadimitriou and Yannakakis, 1990] Papadimitriou, C. and Yannakakis, M. (1990). Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comp.*, 19(2):322–328.

[Pfister, 1995] Pfister, G. (1995). *In Search of Clusters*. Prentice-Hall.

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

[Rapine, 1999] Rapine, C. (1999). *Algorithmes d'approximation garantie pour l'ordonnancement de tâches, Application au domaine du calcul parallèle*. PhD thesis, Institut National Polytechnique de Grenoble.

[Rosenberg, 1999] Rosenberg, A. (1999). Guidelines for data-parallel cycle-stealing in networks of workstations I: on maximizing expected output. *Journal of Parallel Distributing Computing*, 59(1):31–53.

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

[Saad, 1995] Saad, R. (1995). Scheduling with communication delays. *JCMCC*, 18:214–224.

[Schrijver, 1998] Schrijver, A. (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons.

[Thurimella and Yesha, 1992] Thurimella, R. and Yesha, Y. (1992). A scheduling principle for precedence graphs with communication delay. In *International Conference on Parallel Processing*, volume 3, pages 229–236.

[Turek et al., 1992] Turek, J., Wolf, J., and Yu, P. (1992). Approximate algorithms for scheduling parallelizable tasks. In *4th ACM Symposium of Parallel Algorithms and Architecture*, pages 323–332.

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