RAIRO-Oper. Res. 46 (2012) 1–22
DOI: 10.1051/ro/2012005

RAIRO Operations Research
www.rairo-ro.org

SCHEDULING IN THE PRESENCE OF PROCESSOR NETWORKS: COMPLEXITY AND APPROXIMATION

Vincent Boudet[1], Johanne Cohen[2], Rodolphe Giroudeau[1] and Jean-Claude König[1]

Abstract. In this paper, we study the problem of makespan minimization for the multiprocessor scheduling problem in the presence of communication delays. The communication delay between two tasks i and j depends on the distance between the two processors on which these two tasks are executed. Lahlou shows that a simple polynomial-time algorithm exists when the length of the schedule is at most two (the problem becomes $\mathcal{NP}$-complete when the length of the schedule is at most three). We prove that there is no polynomial-time algorithm with a performance guarantee of less than 4/3 (unless $\mathcal{P} = \mathcal{NP}$) to minimize the makespan when the network topology is a chain or ring and the precedence graph is a bipartite graph of depth one. We also develop two polynomial-time approximation algorithms with constant ratio dedicated to cases where the processor network admits a limited or unlimited number of processors.

Résumé. Dans cet article, nous étudions le problème de la minimisation de la longueur de l'ordonnancement pour un système multiprocesseur en présence de délais de communication. Le délai de communication entre deux tâches i et j dépend de la distance entre les processeurs exécutant ces deux tâches. Un algorithme simple, de complexité polynomiale quand la longueur de l'ordonnancement est au plus deux (le problème devient $\mathcal{NP}$-complet quand la longueur de l'ordonnancement est au plus trois) existe, voir [C. Lahlou, *Ordonnancement dans les réseaux de processeurs : complexité et approximation.* Ph.D. thesis, Université Paris VI (1998)]. pour ces deux résultats. Nous démontrons qu'il n'existe pas d'algorithme polynomial ρ-approché avec

Received June 25, 2009. Accepted March 27, 2012.

[1] LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, UMR 5055, France.
boudet@lirmm.fr; rgirou@lirmm.fr; konig@lirmm.fr

[2] LORIA, 54506 Vandoeuvre-lès-Nancy Cedex, France. Johanne.Cohen@loria.fr

$\rho < 4/3$ sous l'hypothèse que $\mathcal{P} \neq \mathcal{NP}$ pour la minimisation de la longueur de l'ordonnancement dans le cas où le réseau de processeurs admet pour topologie une chaîne ou un anneau et le graphe de précédence est un graphe biparti de profondeur un. Nous avons également développé deux algorithmes d'approximation avec des garanties de performance constantes pour les versions limitées et non limitées sur le nombre de processeurs.

Keywords. Scheduling, non-approximability, processor network model.

Mathematics Subject Classification. 68W25, 68Q17, 90B35.

1. Introduction

Scheduling theory is concerned with the *optimal allocation of scarce resources to activities over time*. This topic, which is of obvious practical importance, has been the focus of extensive research since the early 1950s and an impressive amount of literature now exists. The *theory* of the design of scheduling algorithms is more recent, but still has a significant history.

An application is scheduled on a parallel architecture and may be represented by a directed acyclic graph (DAG) $G = (V, E)$ (called a precedence graph) where V designates the set of tasks to be executed on a set of m processors, and E represents the set of precedence constraints. Every task $i \in V$ has a processing time denoted p_i and its execution is subject to precedence constraints.

From the very beginning of the study of scheduling problems, models have kept up with the changing and improving technology. Indeed, some models take into account whether there are communication delays or not.

- In the PRAM model (Parallel Random Access Machine), inter-processor communication delays are not taken into account. In other words, if a precedence constraint between task i and task j exists (arc $(i, j) \in E$), then the starting time of task j, denoted t_j, cannot be less than the completion time of task i. So, $\forall (i, j) \in E, t_j \geq t_i + p_i$.

  The goal is therefore to find a partial order on the tasks and to minimize an objective function. The frontier between scheduling problems that accept polynomial-time algorithms and $\mathcal{NP}$-completeness is based on the number of processors. Indeed, when the number of processors is unlimited, the problem of scheduling a set of n tasks under precedence constraints is polynomial. It is sufficient to use the classical algorithm given by Bellman [3] as well as the two techniques widely used in project management: CPM (Critical Path Method) and $PERT$ (Project/Program Evaluation and Review Technique). *A contrario*, when the number of processors is limited, the problem becomes $\mathcal{NP}$-complete and there is a polynomial-time approximation algorithm with a ratio $(2 - 1/m)$, see [12], where m designates the number of processors. A list scheduling algorithm with no priority is used[3].

- In the `homogeneous scheduling delay model`, inter-processor communication delays are taken into account and the execution of tasks is subject to precedence constraints and communication delays. In other words, if there is a precedence constraint between task i and task j (arc $(i, j) \in E$), an integer communication delay c_{ij} is introduced.

  If task i starts its execution at time t on processor π, and if task j is a successor of task i in G, then either task j starts its execution after time $t + p_i$ on processor π, or after time $t + p_i + c_{ij}$ on another processor.

  This model was first introduced by Rayward-Smith [22] for the simple case $\forall i \in V,\ p_i = 1$ and $\forall (i, j) \in E,\ c_{ij} = 1$. In this model, we have a set of identical processors that are able to communicate in an uniform manner. In fact, the problem is to find a trade-off between the two extreme solutions, namely, executing all tasks sequentially without communication, or trying to use all potential parallelism degrees, but at the cost of increased communication overhead.

These two models have been studied extensively over the past few years both from the complexity and the (non)-approximability standpoints (see [4,14]). Moreover, several variants of these two problems have been extensively studied (see [1] for presentation).

However, data in heterogeneous systems may be distributed across several processors that are not necessarily neighbors. Thus, we must take into account these characteristics to develop algorithms that are adapted to this new type of architecture. The heterogeneous characteristics may be based on the communication delay [2] or concern the processing time [10].

Organization of the paper

This paper is organized as follows: in Section 2, problem definition and notations are specified. Related works and our contributions are also described. Section 3 is devoted to the computational complexity. In Section 4, we propose two polynomial-time approximation algorithms with a limited or unlimited number of processors.

[3]One of the general approximation strategies most used for solving scheduling problems is `list scheduling`, whereby a priority list of the tasks is given, and at each step the first available processor is selected to process the first available task in the list. The accuracy of a given list scheduling algorithm depends on the order in which tasks appear on the list. One of the simplest algorithms is the `LPT algorithm` in which tasks are arranged to a non-increasing p_j, see [13].

2. Problem definition, notations, related works and our contributions

2.1. Problem definition, notations and example

The processor network model is based on a relaxation of the structure of the graph of processors proposed in the `classical scheduling model` (which is highly constrained: the graph of processors is represented by a complete graph). In this article, we consider a non-fully-connected graph of processors, denoted in the following by $G^* = (V^*, E^*)$. Therefore, we consider a scheduling model admitting a distance function, denoted $d(\pi^l, \pi^h)$. The distance function measures the distance (the number of edges in the smallest path between π^l and π^h). In a chain, π^l (resp. π^k) is the l^{th} (resp. k^{th}) vertex then $d(\pi^l, \pi^h) = |l - h|$ (this first distance function was introduced by Picouleau [20]).

Note that graph G^* corresponds to the (undirected) graph of processors and that graph G is the DAG graph corresponding to the task precedence graph.

The communication delay c_{i,π^l,j,π^h} (this notation indicates that the value of the communication delay between task i, which is assigned to processor π^l and task j which is executed on processor π^h) is assumed to be $c_{ij} = d(\pi^l, \pi^h)$, provided that there is a precedence constraint exists between tasks i and j. So the starting time of a task i, denoted t_i, depends both on the communication delay given by the precedence graph and on an allocation of task i and its predecessors on processors.

The distance function defined by Picouleau [20] can be justified as being appropriate for linear array networks. For general network structures, a distance between two processors may be defined as the length of the shortest path that connects a pair of processors. Such a model has been considered [6]. El-Rewini and Lewis [7] also considered a distance function, and they also took contention into account. By contention[4], they consider the event that two or more data transmissions have to pass simultaneously through a single communication channel whose limited capacity forces serial transmission. Other results were given by [8, 18].

Formally, the processor network model may be defined as:

$$\forall (i,j) \in E, t_j \geq t_i + p_i + c_{ij} d(\pi^\ell, \pi^h)$$

where:

- π^ℓ (π^h resp.) represents the processor on which task i (task j resp.) is scheduled;
- t_i represents the starting time of task i;
- p_i represents the processing time of task i;
- $d(\pi^\ell, \pi^h)$ represents the shortest path in graph G^* between π^ℓ and π^h;
- c_{ij} is the communication delay if two tasks are executed on two neighboring processors. Note that the precedence graph is weighted such that a weight of arc (i,j) corresponds to the communication delay between tasks i and j.

[4]We refer the reader to the following position [23] for communication contention problems in scheduling.

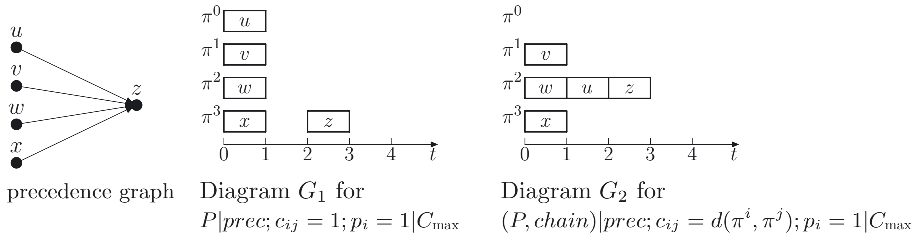

u
v
w
x
z
precedence graph
π^0 π^1 π^2 π^3
u v w x z
0 1 2 3 4 t
Diagram G_1 for $P|prec; c_{ij} = 1; p_i = 1|C_{\max}$
π^0 π^1 π^2 π^3
v w u z x
0 1 2 3 4 t
Diagram G_2 for $(P, chain)|prec; c_{ij} = d(\pi^i, \pi^j); p_i = 1|C_{\max}$

FIGURE 1. Difference between problems $P|prec; c_{ij} = 1; p_i = 1|C_{\max}$ and $(P, chain)|prec; c_{ij} = d(\pi^i, \pi^j); p_i = 1|C_{\max}$.

Note that the previous definition is a generalization of the classic scheduling model with communication delays (or `homogeneous scheduling delay model`, [4,5]). For instance, consider that all distances in the network are equal to one, *i.e.* $d(\pi^\ell, \pi^h) = 1$ for all processors π^ℓ, π^h. In that case, the considered model is exactly the classical scheduling communication delay model.

We consider the classic scheduling $UET - UCT$ (Unit Execution Time-Unit Communication Time, *i.e.* $\forall i \in V, p_i = 1$, and $\forall (i,j) \in E, c_{ij} = 1$) problem on a bounded number of processors, such that the processor network is a ring or a chain (the graph is undirected). In these topologies, processors are numbered as π^0, $\pi^1, \dots,$ π^{n-1} and processor π^k can communicate with processor $\pi^{k'}$ with a communication cost equal to $d(\pi^k, \pi^{k'})$, where $d(\pi^k, \pi^{k'})$ represents the shortest path on graph G^* between processor π^k and $\pi^{k'}$.

In scheduling theory, a problem type is categorized by its machine environment, job characteristic and objective function. Thus, using the *three field* notation scheme $\alpha|\beta|\gamma$ (where α designates the environment processors, β the characteristics of the jobs and γ the criteria), proposed by Graham *et al.* [14],

we consider the problem of minimizing the makespan of a schedule (denoted $C_{\max}$) with unitary task and unitary communication delay ($UET - UCT$) from a precedence graph G on a network of processors having a chain (graph G^*) topology such that the communication delay depends on the shortest path on graph G^*. This problem is denoted $(P, chain)|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$.

Example. Figure 1 shows the difference between the two problems $P|prec; c_{ij} = 1; p_i = 1|C_{\max}$ and $(P, chain)|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$. The processing time of the tasks and the communication delay between tasks are unitary ($UET - UCT$ problem). Gantt diagram G_1 represents an optimal solution for the $P|prec; c_{ij} = 1; p_i = 1|C_{\max}$ problem. Note that task z can be executed on any processor at $t = 2$.

Moreover, Gantt diagram G_2 represents an optimal solution for the problem $(P, chain)|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$. First, we assume that the graph of processors is a chain $C = (V, E)$, such that $V = \{\pi^i : 0 \le i \le 3\}$ and $E = \{(\pi^i, \pi^{i+1}) : 0 \le i \le 2\}$. Second, we can see that task z may be executed on

TABLE 1. Previous complexity results on the processor network model.

Topology	Precedence graph	Complexity	Reference
Unbounded chain	$Tree$	$\mathcal{NP}$-complete	[20]
	$Anti-tree$	$\mathcal{NP}$-complete	
Star	$Tree$	$\mathcal{NP}$-complete	[20]
Star	$Prec$	$\rho \geq 6/5$	[11]
Cycle	$Prec, bipartite$ of depth two	$\rho \geq 4/3$	[18]
Chain	$Prec, bipartite$ of depth two	$\rho \geq 4/3$	[18]

processor π^2 at $t = 2$ only if task u (for example) is delayed by one unit of time and executed on processor π^2 that also executes task z.

2.2. Related works

2.2.1. Complexity results

To the best of our knowledge, the first complexity result was given by Picouleau [20]. The considered problem was to schedule unit execution time tasks with a precedence graph on an unbounded number of processors and on a chain or star[5] topology. Picouleau proved that this problem is $\mathcal{NP}$-complete if the precedence graph is a tree or an out-tree. Recently in [11], the authors proved that there is no polynomial-time algorithm with a performance guarantee smaller than $\frac{6}{5}$ for the minimization of the makespan on a processor network represented by a star. This model is close to the master-slave architecture.

Lahlou in [17], proves whether there is a schedule having the makespan less than k on ring or on a chain network of processors for a precedence graph is $\mathcal{NP}$-complete for $C_{\max} = 3$ (the proof is not given in this article. The fact is that Thm. 3.2 was proved by [18] with the same proof). The previous complexity results are summarized by Table 1.

2.2.2. Approximation results

If the network is a ring there are two approximation results for Rayward-Smith's [22] algorithm:

- in the general case, the performance ratio is upper-bounded by $\frac{m}{2} + \frac{3}{2} - \frac{1}{m}$, and there exists an instance for which the performance ratio is equal to $\frac{m}{8} + \frac{1}{2}$ [18];
- if the number of processors is even, the upper-bound can be improved to $1 + \frac{3}{8}m - \frac{1}{2m}$, and there exists an instance such that the performance ratio is equal to $\lceil \sqrt{m} \rceil$ [17].

Moreover, Hwang *et al.* [16] studied approximation list algorithms for scheduling problems where the communication times depend on contention, on a distance function for the tasks involved and on the processors that execute the tasks. The

[5]A star is a tree of diameter two.

Table 2. Complexity results for $(P, topology)|bipartite; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ with $topology \in \{chain, ring\}$.

	Type of precedence graph and $C_{\max}$	
Topology	Bipartite and $C_{\max} = 2$	Bipartite of depth one and $C_{\max} = 3$
Chain	Poly. (Lem. 3.1)	$\mathcal{NPC}$ (Cor. 3.13)
Ring	Poly. (Lem. 3.1)	$\mathcal{NPC}$ (Cor. 3.13)

authors examined a simple strategy called *Extended List Scheduling, ELS*, which is a straightforward extension of list scheduling. They proved that the *ELS* strategy is unsatisfactory, but improves a strategy called *Earliest Task First*. Lastly in [11], the authors proposed a sophisticated polynomial-time approximation algorithm with a ratio equal to four based on three steps for the problem for the makespan minimization problem on a processor network forming a star.

2.3. Our contributions

First, we study the impact of the processor network hypothesis on the hardness of approximation (hypothesis for which the potential communication delay between two tasks i and j depends only on the distance $d(\pi^i, \pi^j)$ between processors assigned to execute tasks i and j) for the $UET-UCT$ problem in the presence of a specified precedence graph (bipartite). Second, as far as we know, no result concerning a lower bound for any approximation algorithm for this problem has been determined. Thus, the challenge is to determine a threshold for any approximation algorithm for the problem $\alpha|bipartite$ of depth one; $c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$, where $\alpha \in \{(P, chain), (P, ring)\}$. In this paper, we show that there is no polynomial-time algorithm with a performance guarantee smaller than 4/3 for makespan minimization (unless $\mathcal{NP} = \mathcal{P}$). Lastly, we also develop two polynomial-time approximation algorithms for a processor network with limited or unlimited resources.

3. Computational complexity

In this section, we prove some non-approximability results by considering the following scheduling problems: $\alpha|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ with $\alpha \in \{(P, chain), (P, ring)\}$ and $\alpha|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1, dup|C_{\max}$ with $\alpha \in \{(P, chain), (P, ring)\}$. The notation *dup* means that the task duplication is allowed. In other words, a task can be executed by several processors.

We also notice that this problem becomes polynomial for the case $C_{\max} = 2$. The results of this section are summarized in Table 2.

3.1. Makespan minimization for general precedence graphs

3.1.1. A polynomial-time algorithm

Let us first focus on the problem where the makespan is less than or equal to two.

Lemma 3.1. *The problem of deciding whether an instance of* $\alpha|prec; c_{ij} = d(\pi^{\ell}, \pi^{k});\ p_i = 1|C_{\max}$ *has a schedule of length two at most is polynomial with* $\alpha \in \{(P, chain),\ (P, ring)\}$.

Proof. See [18]. □

The remainder of this section is devoted to proving Theorem 3.2.

3.1.2. $\mathcal{NP}$-completeness proof

Recall that the result was given first in [18]. We propose a similar polynomial-time transformation as [18]. We give the proof in order to simplify the understanding of the proof of Lemma 3.9.

Theorem 3.2. *The problem of deciding whether an instance of* $(P, chain)|prec; c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$ *has a schedule of length at most three is* $\mathcal{NP}$*-complete.*

Proof. The proof is established by a reduction of the 3-partition problem [9]:

Instance: a finite set $\mathcal{A}$ of $3M$ elements $\{a_1, \dots, a_{3M}\}$, a bound $B \in \mathbb{N}^+$ and a size $s(a) \in \mathbb{N}$ for each $a \in \mathcal{A}$ such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and such that $\sum_{a\in\mathcal{A}} s(a) = MB$.

Question: can A be partitioned into M disjoint sets $\mathcal{A}_1, \dots, \mathcal{A}_M$ of $\mathcal{A}$ such that for all $i \in [1, \dots, M]$, $B = \sum_{a\in\mathcal{A}_i} s(a) = \dfrac{\sum_{a\in\mathcal{A}} s(a)}{M} \in \mathbb{N}$?

3-partition is known to be $\mathcal{NP}$-complete in the strong sense [9] (even if B is polynomially bounded by the instance size, the problem is still $\mathcal{NP}$-complete).

First, we construct two graphs Z^i and W^b depending on two natural numbers i and b, and we describe the scheduling structure if graph Z^i is the precedence graph in our context. Second, we describe a polynomial-time reduction from 3-partition, that is, we encode numbers $\{a_1, \dots, a_{3M}\}$ by graphs Z^i and each element of partition $\mathcal{A}$ by graph W^B.

3.1.2.1. **Graphs Z^i**

Let i be an integer such that $i > 1$. Graph Z^i consists of $4 \times i$ vertices denoted $Z^i[k, 0]$ and $Z^i[k, 1]$, where $0 \leq k < 2i$. The precedence constraints between these tasks are defined as follows:

- arcs $Z^i[j, 0] \rightarrow Z^i[j, 1]$ for any j, $0 \leq j \leq 2i - 1$;
- arcs $Z^i[2j, 0] \rightarrow Z^i[2j + 1, 1]$ for any j, $0 \leq j \leq i - 1$;

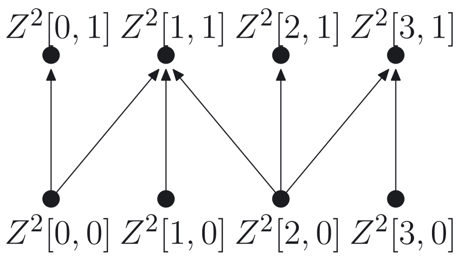


FIGURE 2. Graph Z^2.

- arcs $Z^i[2j,0] \to Z^i[2j-1,1]$ for any j, $1 \leq j \leq i-1$.

Remark 3.3. Valid scheduling of length three for the case where the precedence graph is Z^i in a path of $2i$ processors is as follows, for any j, $0 \leq j \leq 2i-1$,

- tasks $Z^i[j,0]$ and $Z^i[j,1]$ are executed on π^j;
- tasks $Z^i[j,\ell]$ are executed at time ℓ, for any $\ell \in \{0,1\}$, if j is even;
- tasks $Z^i[j,\ell]$ are otherwise executed at time $\ell+1$, for any $\ell \in \{0,1\}$.

See Figure 2 for graph Z^2 and Figure 4 for the valid scheduling described in Remark 3.3.

3.1.2.2. **Graph W^b**

Let b be an integer such that $b > 0$. Graph W^b admits $2(b+3)$ vertices and its set of vertices is defined as follows:
$\{W^b[k,j], k \in \{0, 2b+1\} \text{ and } j \in \{0,1,2\}\} \cup \{W^b[\ell,0] \text{ with } \ell \in \{1,\ldots,2b\}\}$.
The precedence constraints between these tasks are defined as follows:

- arcs $W^b[2j,0] \to W^b[2j+1,0]$ for any j, $0 \leq j \leq b-1$;
- arcs $W^b[2j,0] \to W^b[2j-1,0]$ for any j, $0 \leq j \leq b$;
- arcs $W^b[0,0] \to W^b[0,1]$ and $W^b[0,1] \to W^b[0,2]$;
- arc $W^b[2b,0] \to W^b[2b+1,2]$;
- arcs $W^b[2b+1,0] \to W^b[2b+1,1]$ and $W^b[2b+1,1] \to W^b[2b+1,2]$.

Remark 3.4. Valid scheduling of length three for the case where the precedence graph is W^i in a chain of $2(b+1)$ processors is as follows:

- tasks $W^b[0,j]$ are executed on π^0, for any j, $0 \leq j \leq 2$ at time j;
- tasks $W^b[2b+1,j]$ are executed on π^{2b+1}, for any j, $0 \leq j \leq 2$ at time j;
- task $W^b[j,0]$ is executed on π^j for any j, $0 \leq j \leq 2b$ at time 0 if j is even, otherwise at time 2.

In Figure 3 graph W^b is represented, whereas in Figure 4 a valid scheduling for this graph is described in Remark 3.4 for graph W^b where $b = 3$.

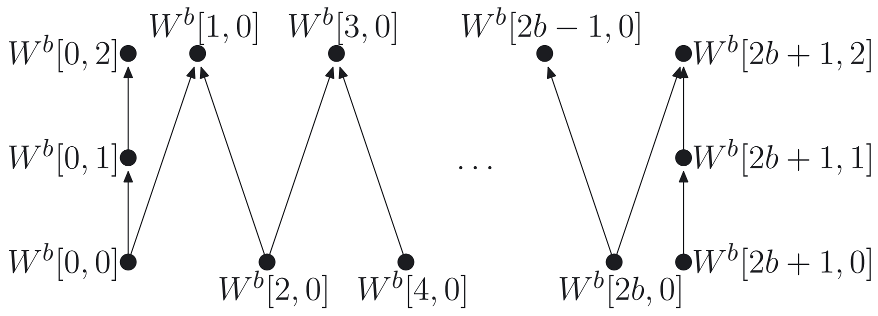
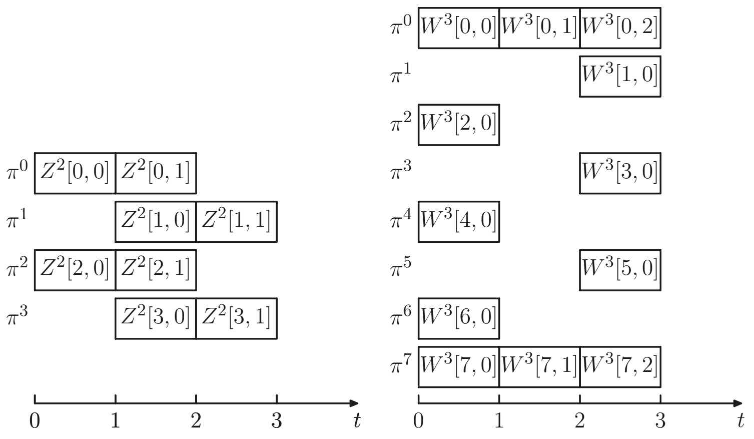


FIGURE 3. Graph W^b.



π^0 $Z^2[0,0]$ $Z^2[0,1]$
π^1 $Z^2[1,0]$ $Z^2[1,1]$
π^2 $Z^2[2,0]$ $Z^2[2,1]$
π^3 $Z^2[3,0]$ $Z^2[3,1]$
0 1 2 3 t
π^0 $W^3[0,0]$ $W^3[0,1]$ $W^3[0,2]$
π^1 $W^3[1,0]$
π^2 $W^3[2,0]$
π^3 $W^3[3,0]$
π^4 $W^3[4,0]$
π^5 $W^3[5,0]$
π^6 $W^3[6,0]$
π^7 $W^3[7,0]$ $W^3[7,1]$ $W^3[7,2]$
0 1 2 3 t

FIGURE 4. Valid scheduling of length three for graphs Z^2 and W^3.

From the definitions in graphs Z^i and W^b, we will proceed to the polynomial-time reduction from 3-PARTITION to the scheduling problem.

Consider an instance $\mathcal{I}$ of 3-PARTITION, $(\mathcal{A} = \{a_1, \dots, a_{3M}\}, B)$, where $\sum_{a\in\mathcal{A}} s(a) = MB$. For simplicity, we assume without loss of generality that $\forall a \in \mathcal{A}, s(a) \geq 2$.

Given an instance $\mathcal{I}$ of the 3-PARTITION problem, we construct an instance $\mathcal{I}'$ of the scheduling problem $(P, chain)|prec; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max} = 3$, in the following manner:

The precedence graph G is decomposed into two disjointed graphs, denoted $\mathcal{W}$ and $\mathcal{Z}$.

- Graph $\mathcal{W}$ has connected components W_i, *i.e.* $\mathcal{W} = \cup_i W_i, \forall i \in \{1, \dots, M\}$, where W_i is a copy of graph W^B (defined previously);

- graph $\mathcal{Z}$ is a collection of graphs $Z^{s(a_j)}$ *i.e.* $\mathcal{Z} = \cup_{a_j \in \mathcal{A}} Z^{s(a_j)}$. Each element a_j of collection A is represented by a graph $Z^{s(a_j)}$ in the precedence graph G (see Fig. 2). Recall that all graphs $Z^{s(a_j)}$ form graph $\mathcal{Z}$ and the number of tasks in graph $Z^{s(a_j)}$ is $4s(a_j)$.

To complete this transformation, the chain consists of $2M(B+1)$ processors numbered $\pi^0, \dots, \pi^{2M(B+1)-1}$.

To sum up, precedence graph G is composed of M copies of graph W^B having $(2B+6)$ vertices, and of graph $Z^{s(a)}$ for each element a of collection $\mathcal{A}$ (having $4s(a)$ vertices). Thus, G admits $(6B+6)M$ vertices.

The size of the instance of our problem and of the instance of 3-PARTITION are thus polynomially related (see [9]). Note that the running time of this transformation depends on B. Since problem 3-PARTITION[6] is $\mathcal{NP}$-complete in the strong sense, the whole transformation can be carried out in polynomial time.

Remark 3.5. With the defined precedence constraints, in any valid schedule of length three, there is no idle time.

We will now prove that this transformation is a reduction.

- Let us assume that $\mathcal{A} = \{a_1, \dots, a_{3M}\}$ can be partitioned into M disjoint subsets $\mathcal{A}_1, \dots, \mathcal{A}_M$ with each summing up to B. We will then prove that there is a schedule of length three at most.

  Let us construct this schedule;

  Consider a partition $\mathcal{A}_i = \{a_{i^1}, a_{i^2}, a_{i^3}\}$ of $\mathcal{A}$. The tasks of graphs $Z^{\mathcal{A}_i} = \cup_{a \in \mathcal{A}_i} Z^{s(a)}$ are executed on $2B$ processors as described in Remark 1, denoted π^{k_i+1} and π^{k_i+2B} (for a fixed $k_i = (i-1)(2B+1)$, $1 \le i \le M$. Concerning a widget W_i, paths of length two must be executed on the same processors, so we execute $W_i[0,j]$ (resp. $W_i[2B+1,j]$) at $t = j, j \in \{0,1,2\}$ on processor π^{k_i} (resp. on π^{k_i+2B+1}). The other tasks of W_i are scheduled as described in Remark 2.
- Let us assume now that there is a schedule S of length at most three. We will prove that $\mathcal{A} = \{a_1, \dots, a_{3M}\}$ can be partitioned into M disjoint subsets $\mathcal{A}_1, \dots, \mathcal{A}_M$ with each summing up to B.

  Lemma 3.6. *In any valid schedule of length three, for any i, $1 \le i \le M$, tasks from a widget W_i are processed by processors π^k, …, $\pi^{k+(2B+1)}$, where task $W_i[0,0]$ is executed on π^k or on $\pi^{k+(2B+1)}$, for a fixed k.*

  Proof. We suppose, that task $W_i[0,0]$ is executed at $t = 0$ on processor π^k for a fixed k. Task $W_i[0,1]$ (resp. $W_i[0,2]$) is thus allotted at $t = 1$ (resp. $t = 2$) on processor π^k. Task $W_i[1,0]$ is processed at $t = 2$ on processor π^{k+1} or π^{k-1}. Without loss of generality, we consider the case π^{k+1}, as the other case is similar.

  Assume that task $W_i[2,0]$ is scheduled on processor π^{k+1}. Since this task allows two successors ($W_i[1,0]$ and $W_i[3,0]$), then this task is executed at $t = 0$.

[6]The numbers of the instance are naturally represented in unary.

Remark 3.7. Task $W_i[3,0]$ cannot be processed on processor π^{k+1}. If that were the case, then it would be executed at $t=1$ and this task has another predecessor (task $W_i[4,0]$). This is not possible due to the communication delay.

On processor π^{k+1}, tasks $W_i[2,0]$ and $W_i[1,0]$ are actually scheduled at $t=0$ and $t=2$. So only one task of graph $Z^{s(a_j)}$ may be executed on processor π^{k+1}. This is impossible since there is no isolated task in graph $Z^{s(a_j)}$. Thus, task $W_i[2,0]$ is allotted on processor π^{k+2} at $t=0$.

By recurrence, it is easy to see that all tasks $W_i[2l,0]$ (resp. $W_i[2l+1,0]$) are executed at $t=0$ on processors π^{k+2l} (resp. at $t=2$ on processor π^{k+2l+1}).

Therefore, task $W_i[2B+1,1]$ (resp. $W_i[2B+1,2]$) is scheduled at $t=1$ (resp. $t=2$) on processor π^{k+2B+1}.

Note that two different graphs W_i and $W_{i'}$ cannot be interlaced. Indeed, let π (resp. π') be the processor on which task $W_i[0,1]$ (resp. $W_i[2B+1,2]$) is executed. Between π and π', we know there is no processor with three idle times. Since widget $W_{i'}$ admits two paths of length three, then no path of length three may be scheduled between processor π and π', and so two widgets cannot be interlaced.

In conclusion, all tasks from widget W_i in any valid schedule of length three are scheduled between processors π^k and $\pi^{k+(2B+1)}$.

This concludes the proof of Lemma 3.6. □

Supposing that $W_i[0,0]$ is allotted on processor π^k and $W_i[0,2B+1]$ is scheduled on π^{k+j} (with $j=2B+1$ or $j=-(2B+1)$. Since there is no idle time, if a task from graph $Z^{s(a_j)}$ is executed between these two processors, then all other tasks from graph $Z^{s(a_j)}$ must also be executed between these two processors.

Next, we will construct a partition $\{\mathcal{A}_1,\ldots,\mathcal{A}_M\}$ with the desired properties from schedule S of length three.

Since there are $2M(B+1)$ processors, it follows from Lemma 3.6 and the remark above that for any widget W_i there are exactly three graphs Z^{j_1}, Z^{j_2} and Z^{j_3} executed between the two three-task paths of W_i. The three elements in correspondence with Z^{j_1}, Z^{j_2} and Z^{j_3} make $\mathcal{A}_i$.

This concludes the proof of Theorem 3.2. □

It is easy to extend the result given by Theorem 3.2 to the ring topology, in which processor π^{m-1} is connected to processors π^{m-2} and π^0.

Proposition 3.8. *The problem of deciding whether an instance of* $(P, ring)|prec;c_{ij} = d(\pi^\ell,\pi^k);p_i = 1|C_{\max}$ *has a schedule of length at most three is* $\mathcal{NP}$*-complete.*

3.2. Makespan minimization for bipartite graphs of depth one

The construction suggested previously can be easily adapted to obtain a bipartite graph of depth one. For that purpose, we consider the graph shown in Figure 5. The polynomial-time construction is proposed below.

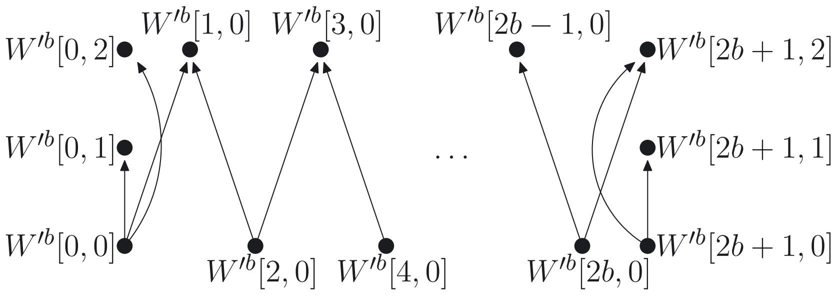


FIGURE 5. Graph W'^b for problem $(P, chain)|bipartite$ of depth one; $c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$.

Figure 5 illustrates a new graph W'^B that will be used to prove Lemma 3.9 and Corollary 3.11. Graph W'^B is the same as graph W^B, except that:

- There is no arc between $W'^B[0,1]$ and $W'^B[0,2]$, and between $W'^B[2b+1,1]$ and $W'^B[2b+1,2]$;
- There is an arc from $W'^B[0,0]$ to $W'^B[0,2]$, and between $W'^B[2b+1,0]$ and $W'^B[2b+1,2]$.

As the proof of Theorem 3.2, the precedence graph contains M copies of graph W'^B denoted $W_1'^B$, $W_2'^B$, ..., $W_M'^B$. We add some precedence constraints: for every integer i between 1 and $M-1$, there are one arc from $W_{i+1}'^B[0,0]$ to $W_i'^B[2B+1,2]$ and one from $W_i'^B[2B+1,0]$ to $W_{i+1}'^B[0,2]$. The graph obtained by this procedure is denoted W'.

Lemma 3.9. *For every integer ℓ between* 1 *and* $M-1$, *for every integer* $j \in \{0, 2B+1\}$, *the three tasks* $W_{i+1}'^B[j,k]$ *with* $k \in \{0,1,2\}$ *must be executed on the same processor.*

Proof. First, we consider the tasks executed on processor π^0. From Remark 3.5, in any valid schedule of length three, there are three tasks executed on it.

Note that there is no valid schedule (with no idle time) of length three where all three tasks executed on processor π^0 belong to a graph of type Z^α (see Fig. 6).

So there exists at most one task of graph W' executed on processor π^0. Only tasks in $W_1'^B$ and $W_M'^B$ can be executed on it (due to the definition of W'). Without loosing generality, we assume that task $W_1'^B$ can be executed on π^0 (otherwise, the index of processors can be reversed).

- Assume that task $W_1'^B[1,0]$ is allotted on processor π^0 at $t=2$. By communication delay, task $W_1'^B[2,0]$ is scheduled at $t=0$ on a processor π.
  - If $\pi = \pi^1$ then task $W_1'^B[0,0]$ have to be executed on π^0 at $t=0$. Therefore one of the both successor tasks ($W_1'^B[0,1]$ and $W_1'^B[0,2]$) of $W_1'^B[0,0]$ must be executed on π^1 at $t=2$. It is impossible because there is a idle time at $t=1$ on π^1;

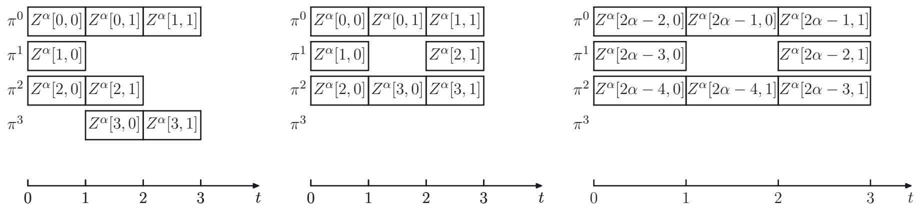

π^0 $Z^\alpha[0,0]$ $Z^\alpha[0,1]$ $Z^\alpha[1,1]$
π^1 $Z^\alpha[1,0]$
π^2 $Z^\alpha[2,0]$ $Z^\alpha[2,1]$
π^3 $Z^\alpha[3,0]$ $Z^\alpha[3,1]$
π^0 $Z^\alpha[0,0]$ $Z^\alpha[0,1]$ $Z^\alpha[1,1]$
π^1 $Z^\alpha[1,0]$ $Z^\alpha[2,1]$
π^2 $Z^\alpha[2,0]$ $Z^\alpha[3,0]$ $Z^\alpha[3,1]$
π^3
π^0 $Z^\alpha[2\alpha-2,0]$ $Z^\alpha[2\alpha-1,0]$ $Z^\alpha[2\alpha-1,1]$
π^1 $Z^\alpha[2\alpha-3,0]$ $Z^\alpha[2\alpha-2,1]$
π^2 $Z^\alpha[2\alpha-4,0]$ $Z^\alpha[2\alpha-4,1]$ $Z^\alpha[2\alpha-3,1]$
π^3
0 1 2 3 t

FIGURE 6. How to schedule graph Z^α in order to have a schedule of length three.

– if $\pi = \pi^0$, then it is impossible since on π^0 there is a idle time at $t = 1$ (the precedence graph admits none isolated tasks);
– if $\pi = \pi^j$ where $j \geq 2$, then it is impossible due to the communication delay.

So, in any case, it is impossible that task $W_1'^B[1,0]$ is allotted on processor π^0 at $t = 2$.

- Assume now that task $W_1'^B[1,0]$ is allotted on processor π^j with $j \geq 2$ at $t = 2$. So task $W_1'^B[0,0]$ (resp. $W_1'^B[2,0]$) is scheduled at $t = 0$ on π^{j-1} (resp. on π^{j+1}). Thus on π^{j-2} (resp. π^{j-1}) there are three (resp. none) idle slots or on π^{j-2} (resp. π^{j-1}) there two (resp. one) idle slots. This case is also impossible.

Therefore, task $W_1'^B[1,0]$ is allotted at $t = 2$ on processor π^1. It is impossible that $W_1'^B[0,0]$ or $W_1'^B[2,0]$ can be executed in processor π^1 (otherwise there is an idle time on processor, if the communication constraint is respected).

Suppose that $W'[2,0]$ is processed on π^0 at $t = 0$:

- suppose that $W'[3,0]$ is processed on π^0 at $t = 2$. Impossible, there is a idle time at $t = 1$ on π^0;
- now, $W'[3,0]$ is scheduled on π^0 at $t = 2$. Impossible by the remark given at the beginning of the proof.

Thus, $W'[2,0]$ is processed on π^1 at $t = 0$. So tasks $W_1'^B[0,0]$ and $W_1'^B[2,0]$ are processed at $t = 0$ on π^0 and π^2. The other successor tasks of $W_1'^B[0,0]$ ($W_1'^B[0,1]$ and $W_1'^B[0,2]$) are scheduled on π^0 from $t = 1$ to $t = 3$.

Figure 7 gives an illustration of Lemma 3.9 first part proof.

Using the same arguments as previously, it is clear that tasks $W_1'^B[2B+1,0]$, $W_1'^B[2B+1,1]$ and $W_1'^B[2B+1,2]$ are allotted to the same processor π^{2B+1}.

Moreover, we can deduce that task $W_2'^B[0,0]$ (resp. $W_2'^B[2,0]$) is executed on π^{2B+2} at time 0 (resp. at time 2) because there is an arc from $W_2'^B[0,0]$ (resp. $W_1'^B[2B+1,0]$) to $W_1'^B[2B+1,2]$ (resp. $W_2'^B[2,0]$). So $W_2'^B[1,0]$ is executed on π^{2B+2} at time 1 (since there is no idle time in any valid schedule) and $W_2'^B[0,1]$ is executed on π^{2B+3} at time 2. Using the same arguments as previously, it is clear that tasks $W_2'^B[2B+1,0]$, $W_2'^B[2B+1,1]$ and $W_2'^B[2B+1,2]$ are allotted to the same processor. We can apply the same reasoning by replacing by $W_2'^B$ by $W_\ell'^B$ (by adapting also the index of the processor).

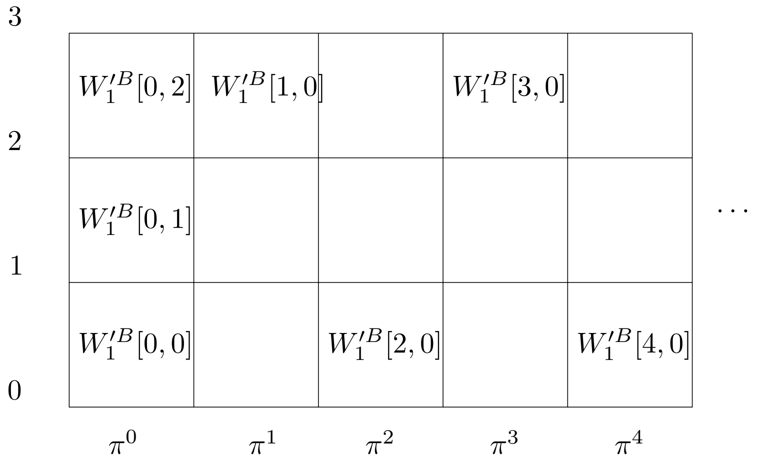


FIGURE 7. Illustration of the Lemma 3.9 first part proof.

Therefore, for every integer ℓ between 1 and $M-1$, for every integer $j \in \{0, 2B+1\}$, the three tasks $W'^B_{i+1}[j,k]$ with $k \in \{0,1,2\}$ must be executed on the same processor. □

Remark 3.10. With the previous lemma, an allocation of the W'-tasks on the processors are identical as W-tasks related to Theorem 3.2 and the same assertions remain valid. Therefore, we get an identical schedule for W' as W.

Moreover, from the proof of Theorem 3.2, we can derive the following corollary:

Corollary 3.11. *The problem of deciding whether an instance of $\alpha|$bipartite of depth one; $c_{ij} = d(\pi^\ell, \pi^k)$; $p_i = 1|C_{\max}$ has a schedule of length at most three is $\mathcal{NP}$-complete with $\alpha \in \{(P, chain),\ (P, ring)\}$.*

The proof of Theorem 3.2 therefore implies that the problem where the tasks can be duplicated is also $\mathcal{NP}$-complete.

Corollary 3.12. *The problem of deciding whether an instance of $\alpha|$bipartite of depth one; $c_{ij} = d(\pi^\ell, \pi^k)$; $p_i = 1, dup|C_{\max}$ has a schedule of length at most three is $\mathcal{NP}$-complete with $\alpha \in \{(P, chain), (P, ring)\}$.*

Proof. The proof comes directly from Theorems 3.2 and 3.11. In fact, Lemma 3.5 implies that no task can be duplicated (the number of tasks is equal to the threefold the number of processors). □

Moreover, a non-approximability result can be deduced.

Corollary 3.13. *No polynomial-time algorithm exists with a performance guarantee bound of less than* $\frac{4}{3}$ *unless* $\mathcal{P} = \mathcal{NP}$ *for the problems* $\alpha|\mathit{bipartite\ of\ depth\ one}; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ *and* $\alpha|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1, \mathit{dup}|C_{\max}$ *with* $\alpha \in \{(P, \mathit{chain}),\ (P, \mathit{ring})\}$.

Proof. The proof of Corollary 3.13 is an immediate consequence of the impossibility theorem, see page 4 of [5]. □

Remark 3.14. Note that in case the topology is a cycle or a path (oriented version of ring and chain), the problem of deciding whether an instance $\alpha|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ with $\alpha \in \{(P, \mathit{cycle}), (P, \mathit{path})\}$ and $\beta \in \{\mathit{prec}, \mathit{bipartite}\}$ has a schedule of at most two is polynomial. It is sufficient to use the same argument as proof of Lemma 3.1. It seems that it is at least as difficult, in the complexity theory sense, to schedule a precedence graph on a cycle/chain as on a ring/chain. Nevertheless, the polynomial-time transformation proposed in the proof of Theorem 3.2 cannot be easily extended to the case where the processor graph topology is an oriented graph (cycle or chain).

4. Polynomial-time approximation algorithms

We design two polynomial-time approximation algorithms with limited or unlimited resources. The classic polynomial-time approximation algorithm for the scheduling problem $\bar{P}|\mathit{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ proposed by Munier and König (see [19]) is used. In [19], Munier and Köning develop a polynomial-time (4/3)-approximation algorithm for the scheduling problem with a precedence graph with unitary tasks and unitary communication time.

The principle of our polynomial-time approximation algorithms is the following. We determine a feasible schedule for problem $\bar{P}|\mathit{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ using the polynomial-time (4/3)-approximation algorithm (see [19]). Note that our problem is different from the problem $\bar{P}|\mathit{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ because in our problem, the communication delay depends on the distance between the processors performing tasks. From this solution, we derive a solution for the problem taking into account the communication delays. This solution is a $\sqrt{n}$-approximation solution for problem $(\bar{P}, \mathit{chain})|\mathit{prec};\ c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$. For the case where the number of processors is limited, we adapt this solution with this additional constraint.

For the remaining section, we give some notations:

- For $\bar{P}|\mathit{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ problem, σ^∞ (resp. $C^\infty_{\max}$) designates the schedule obtained by the polynomial-time (4/3)-approximation algorithm (resp. its length). The length of the optimal schedule is denoted $C^{\infty,\mathrm{opt}}_{\max}$.
- For $(\bar{P}, \mathit{chain})|\mathit{prec}; c_{ij} = d(\pi^l, \pi^k); p_i = 1|C_{\max}$ problem, $\sigma^{\mathrm{chain},\infty}$ (resp. $C^{\mathrm{chain},\infty}_{\max}$) designates the schedule obtained by our approximation algorithm (resp. its length) when the number of processors is unlimited. The length of the optimal schedule is denoted $C^{\mathrm{chain},\infty,\mathrm{opt}}_{\max}$). When there are m processors,

$C_{\max}^{\text{chain},m}$ (resp. $C_{\max}^{\text{chain},m,\text{opt}}$) represents the length of the schedule computed by our approximation algorithm (resp. the optimal length).

Definition 4.1. We consider two tasks x and y in σ^{∞}. We say that x and y are matched in σ^{∞}, if x and y are executed on the same processor such that the starting time of x is just after the completion of y and there is an arc from x to y in the precedence graph or conversely.

Theorem 4.2. *There is a $\sqrt{n}$-approximation for the problem $(\bar{P}, chain)|prec; c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$ and it exists a instance which for the ratio is strictly less than $\frac{2\sqrt{n}}{3}$.*

Proof. Let I be the instance of the problem $(\bar{P}, chain)|prec;\ c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$. We split our proof into two parts. The first part considers the instances where the length of the optimal scheduling is less than three. And the second case deals with all other instances.

First, we consider the instances where the the length of the optimal scheduling is less than 3 i.e: $C_{\max}^{\text{chain},\infty,\text{opt}} < 3$. There are two simple cases :

- if $C_{\max}^{\text{chain},\infty,\text{opt}} = 1$ then it is easy to find an optimal schedule;
- if $C_{\max}^{\text{chain},\infty,\text{opt}} = 2$ then $C_{\max}^{\infty,\text{opt}} = 2$. Therefore, since the algorithm proposed in [19] is a $\frac{4}{3}$-approximation algorithm, this algorithm finds an optimal solution. Indeed, the gap between a feasible solution given by the Munier-König algorithm and an optimal solution is at most 4/3 and $(4/3 * 2 < 3)$.

Second, we consider the instances where $C_{\max}^{\text{chain},\infty,\text{opt}} \geq 3$. Now, for the problem $(\bar{P}, chain)|prec;\ c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$, we construct a schedule $\sigma^{\text{chain},\infty}$ from the schedule σ^{∞}. A polynomial-time (4/3)-approximation algorithm (see [19]) returns a feasible schedule σ^{∞} for problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{\max}.$. Note that $C_{\max}^{\infty} \leq 4/3 C_{\max}^{\infty,\text{opt}}$) by definition of the (4/3)-approximation algorithm.

For the remaining of the proof, we introduce some notations. We denote by m' the maximal numbers of processors used (at the same time) by σ^{∞} and by X_i the set of tasks executed at step t_i by σ^{∞}.

Recall that n is the number of tasks in the schedule.

From σ^{∞}, we describe the construction of schedule $\sigma^{\text{chain},\infty}$ according to the values of m' and n. To explain this construction,

(1) If $m' \leq \sqrt{n}$ then we keep schedule σ^{∞} and we add $m' - 2$ idle times between the X_i-tasks and X_{i+1}-tasks with i even (see Fig. 8 for an illustration). The obtained schedule respects all constraints of problem $(\bar{P}, chain)|prec;\ c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$ since there is, in σ^{∞}, at most a matching between X_i and X_{i+1}-tasks.

Moreover, the distance between the starting time of X_i-tasks and X_{i+2}-tasks is at least $m' - 1$ as the distance between any pair of processors by definition of parameter m'.

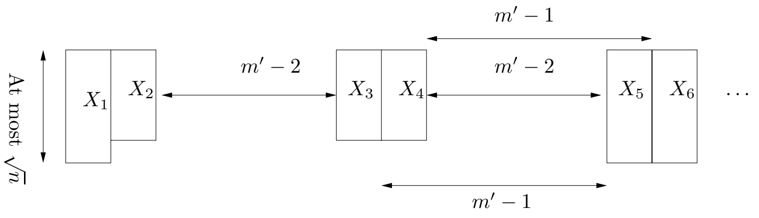


FIGURE 8. Feasible schedule for $\sigma^{\text{chain},\infty}$ for the case where $m' \leq \sqrt{n}$.

Therefore the length of $\sigma^{\text{chain},\infty}$ is at most the length of σ^{∞} plus the length of σ^{∞} multiplied by the factor $\frac{(m'-2)}{2}$.

$$\begin{aligned}
C_{\max}^{\text{chain},\infty} &\leq C_{\max}^{\infty} + \frac{(m'-2)}{2} C_{\max}^{\infty} \\
&\leq \frac{m'}{2} C_{\max}^{\infty} \\
&\leq \frac{2m'}{3} C_{\max}^{\infty,\text{opt}} (\text{since } C_{\max}^{\infty} \leq 4/3 C_{\max}^{\infty,\text{opt}}) \\
&\leq \frac{2m'}{3} C_{\max}^{\text{chain},\infty,\text{opt}} \\
&\leq \sqrt{n} C_{\max}^{\text{chain},\infty,\text{opt}}.
\end{aligned}$$

(2) Otherwise ($m' > \sqrt{n}$), we apply the same way to transform set X_i for all $i = 1, \ldots, C_{\max}^{\infty}$ (See Fig. 9 for an illustration). The X_i tasks are split into $\max\left(\left\lceil \frac{|X_i|}{\sqrt{n}} \right\rceil, 1\right)$ pairwise disjoint subsets. The unit time i of schedule σ^{∞} correspond to $\max\left(\left\lceil \frac{|X_i|}{\sqrt{n}} \right\rceil, 1\right)$ $\sigma^{\text{chain},\infty}$ unit times of schedule $\sigma^{\text{chain},\infty}$. In other words, all of these tasks in X_i are executed within $\max\left(\left\lceil \frac{|X_i|}{\sqrt{n}} \right\rceil, 1\right)$ time units in $\sigma^{\text{chain},\infty}$. As expressed previously, we add $\sqrt{n}-2$ idle times between X_i-tasks and X_{i+1}-tasks for i even. Moreover, let us have two sets of tasks X_{i_1} and X_{i_1+1} with $i_1 \geq 1$ odd. Let p be the size of the matching between X_{i_1}-tasks and X_{i_1+1}-tasks. So we have $|X_{i_1}| \geq p$ and $|X_{i_1+1}| \geq p$. We first execute the matched tasks on the same processors, and other tasks as soon as possible.

$$\begin{aligned}
C_{\max}^{\text{chain},\infty} &\leq \sum_{i=1}^{n} \left\lfloor \frac{|X_i|}{\sqrt{n}} \right\rfloor + C_{\max}^{\infty} + \frac{(\sqrt{n}-2)}{2} C_{\max}^{\infty} \\
&\leq \sqrt{n} + C_{\max}^{\infty} + \frac{(\sqrt{n}-2)}{2} C_{\max}^{\infty} \\
&\leq \sqrt{n} + \frac{2\sqrt{n}}{3} C_{\max}^{\infty,\text{opt}}.
\end{aligned}$$

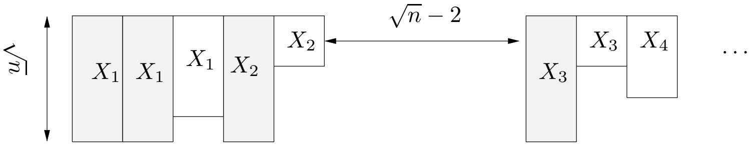


FIGURE 9. Feasible schedule for $\sigma^{\text{chain},\infty}$ for the case where $m' > \sqrt{n}$.

Since $3 \leq C_{\max}^{\text{chain},\infty,\text{opt}}$ (by assumption), we have,

$$\begin{aligned} C_{\max}^{\text{chain},\infty} &\leq \frac{\sqrt{n}}{3} C_{\max}^{\text{chain},\infty,\text{opt}} + \frac{2\sqrt{n}}{3} C_{\max}^{\text{chain},\infty,\text{opt}} \\ &\leq \sqrt{n} C_{\max}^{\text{chain},\infty,\text{opt}}. \end{aligned}$$

Therefore, the theorem follows.

Quasi-Tightness of the bound:

Now, we will construct the instance for which the ratio is strictly less than $\frac{2\sqrt{n}}{3}$.

First, we will focus on the instance where for a given integer x, the precedence graph $G' = (V', E')$ is as follows : $V' = \{u_i, v_i, s_i, u_{x+1} : 1 \leq i \leq x\}$ and $E' = \{(u_i \to v_i), (u_i \to s_i), (v_i \to u_{i+1}), (s_i \to u_{i+1}) : 1 \leq i \leq x\}$.

For the problem $(\bar{P}, chain)|prec;\ c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$, an optimal scheduling solution corresponds to execute all tasks on the same processor and $C_{\max}^{\text{chain},\infty,\text{opt}} = 3x + 1$.

For the problem $\bar{P}|prec; c_{ij} = 1; p_i = 1|C_{\max}$, the Munier-König algorithm returns a feasible solution such that for any i, $1 \leq i \leq x+1$, u_i is executed at time $4(i-1)+1$. For any i, $1 \leq i \leq x$, v_i and s_i are executed at time $4(i-1)+3$. This schedules uses 2 processors. Thus, $C_{\max}^{\infty} = 4x + 1$.

Second, we consider an instance where the graph precedence is $\frac{\sqrt{n}}{2}$ copies of graphs G' where x is equal to $2\sqrt{n}$ (see Fig. 10). Using the same argument as previously, the schedule uses $\sqrt{n}$ processors. We get $C_{\max}^{\text{chain},\infty,\text{opt}} = 6\sqrt{n} + 1$ and $C_{\max}^{\infty} = 8\sqrt{n} + 1$. From these equalities, this implies that the algorithm described in the first part of the proof returns a schedule such that $C_{\max}^{\text{chain},\infty} = 4n + \frac{\sqrt{n}}{2}$. □

Remark 4.3. We conjecture that the ratio is strictly less than $\frac{2\sqrt{n}}{3}$.

Now we focus on the problem where the number of processors is limited.

Theorem 4.4. *There is a $(1 + \frac{2m}{3})$-approximation algorithm with m representing the number of processors for the problem $(P, chain)|prec;\ c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ and the bound is tight.*

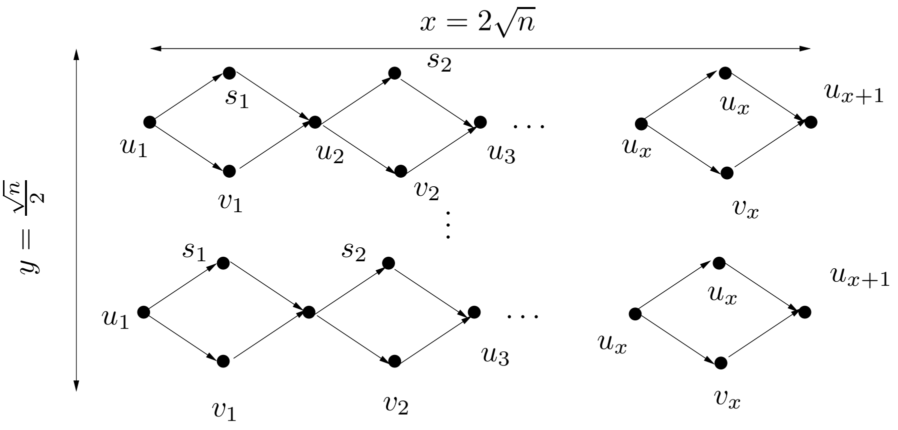


FIGURE 10. Instance for which the ratio is strictly less than $\frac{2\sqrt{n}}{3}$.

Proof. We use the same algorithm as that given in proof of Theorem 4.2 case (4) by replacing $\sqrt{n}$ by m. So we obtain

$$C_{\max}^{\text{chain},m} \leq \sum_{i=1}^{n} \left\lfloor \frac{|X_i|}{m} \right\rfloor + C_{\max}^{\infty} + \frac{(m-2)}{2} C_{\max}^{\infty}$$

Since $\sum_{i=1}^{n} \lfloor \frac{|X_i|}{m} \rfloor \leq C_{\max}^{\infty,\text{opt}}$, we have:

$$\begin{aligned} C_{\max}^{\text{chain},m} &\leq C_{\max}^{\infty,\text{opt}} + \frac{4}{3} \times \frac{m}{2} C_{\max}^{\infty,\text{opt}} (\text{from [19]}) \\ &\leq C_{\max}^{\text{chain},m,\text{opt}} + \frac{4}{3} \times \frac{m}{2} C_{\max}^{\text{chain},m,\text{opt}} \\ &\leq \left(1 + \frac{2}{3} m\right) C_{\max}^{\text{chain},m,\text{opt}}. \end{aligned}$$

Therefore, the theorem follows.

Tightness of the bound: now, we will construct the instance for which the ratio is $1 + \frac{2m}{3}$. We consider the graph G having $(\frac{m}{2} + 1)$ copies of graph G' with $x = \frac{1}{3}(\frac{2n}{m+2} - 1)$ described in the proof of Theorem 4.2. So using the same argument using in the proof of Theorem 4.2, we get $C_{\max}^{\text{chain},\infty,\text{opt}} = \frac{2n}{m+2}$ and $C_{\max}^{\infty} = \frac{4}{3}(\frac{2n}{m+2} - 1) + 1$. This implies that the algorithm described in the first part of the proof returns a schedule such that $C_{\max}^{\text{chain},m} = (2m+3)x + 1$. So we obtain ρ. □

5. Conclusion

We proved that the problem of deciding whether an instance of $\alpha|\textit{bipartite of depth one}; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ with $\alpha \in \{(P, \textit{chain}), (P, \textit{ring})\}$ has a schedule of length at most three is $\mathcal{NP}$-complete. The complexity result is an extension of [18]. This result may be compared to that in [15], which states that $P|\textit{prec}; c_{ij} = 1; p_i = 1|C_{\max} = 4$ is $\mathcal{NP}$-complete (they use a reduction from the $\mathcal{NP}$-complete problem *Clique*), whereas Picouleau [21] develops a polynomial-time algorithm for $C_{\max} = 3$. Their result implies that there is no ρ-approximation algorithm with $\rho < \frac{4}{3}$, unless $\mathcal{P} = \mathcal{NP}$.

The previous results have been extended to the case where the duplication is allowed.

Lastly, we complete our complexity results by developing a polynomial-time algorithm for $(\bar{P}, \textit{chain})|\textit{prec}, c_{ij} = d(\pi^l, \pi^k) = 1, p_i = 1|C_{\max}$ (resp. $(P, \textit{chain})|\textit{prec}, c_{ij} = d(\pi^l, \pi^k) = 1, p_i = 1|C_{\max}$) with a worst-case relative performance of $\sqrt{n}$ (resp. $1 + \frac{2m}{3}$).

An interesting question for further research is to find a polynomial-time approximation algorithm with a constant ratio for the scheduling problem on a limited number of processors.

Acknowledgements. The authors would like to thank the three anonymous referees and D. Manley for their helpful corrections and suggestions which improved the readability of this article. We also want to thank C. Lahlou for pointing out reference [18].

References

[1] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt and J. Węglarz, *Handbook on Scheduling.* Springer (2007).

[2] E. Bampis, A. Giannakos and J.C. König, On the complexity of scheduling with large communication delays. *Eur. J. Oper. Res.* **94** (1996) 252–260.

[3] R.E. Bellman, On a routing problem. *Quart. Appl. Math.* **16** (1958) 87–90.

[4] B. Chen, C.N. Potts and G.J. Woeginger, Handbook of Combinatorial Optimization, in *A review of machine scheduling: Complexity, algorithms and approximability* **3**. Kluwer Academic Publishers (1998).

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

[6] K.H. Ecker and H. Hodam, *Heuristic algorithms for the task scheduling under consideration of communication delays.* Technical Report, T.U. Clausthal (1996).

[7] H. El-Rewini and T.G. Lewis, Scheduling parallel program tasks onto arbitrary target machines. *J. Parallel Distribut. Comput.* **9** (1990) 138–153.

[8] L. Finta and Z. Liu, Complexity of task graph scheduling with fixed communication capacity. *Int. J. Found. Comput. Sci.* **8** (1997) 43–66.

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

[10] A. Giannakos, *Algorithmique pour le parallélisme : certains problèmes d'ordonnancement de tâches et algorithmes de couplage.* Ph.D. thesis, Université de Paris-XI Orsay (1997).

[11] R. Giroudeau, J.C. König and B. Valéry, Scheduling UET-tasks on a star network: complexity and approximation. *Quart. J. Oper. Res.* **9** (2011) 29–48.

[12] R.L. Graham, Bounds for certain multiprocessing anomalies. *Bell System Tech. J.* **45** (1966) 1563–1581.

[13] R.L. Graham, Bounds on the performance of scheduling algorithms, *Computer and job-shop scheduling theory.* E.G. Coffman edition, John Wiley Ltd. (1976).

[14] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Ann. Discrete Math.* **5** (1979) 287–326.

[15] J.A. Hoogeveen, J.K. Lenstra and B. Veltman, Three, four, five, six, or the complexity of scheduling with communication delays. *Oper. Res. Lett.* **16** (1994) 129–137.

[16] J.-J. Hwang, Y.C. Chow, F.D. Anger and C.-Y. Lee, Scheduling precedence graphs in systems with interprocessor communication times. *SIAM J. Comput.* **18** (1989) 244–257.

[17] C. Lahlou, Scheduling with unit processing and communication times on a ring network: Approximation results, in *Proceedings of Europar.* Springer-Verlag (1996) 539–542.

[18] C. Lahlou, *Ordonnancement dans les réseaux de processeurs : complexité et approximation.* Ph.D. thesis, Université Paris VI (1998).

[19] A. Munier and J.C. König, A heuristic for a scheduling problem with communication delays. *Oper. Res.* (1997) 145–148.

[20] C. Picouleau, *UET – UCT schedules on arbitrary networks.* Technical Report, LITP, Blaise Pascal, Université Paris VI (1994).

[21] C. Picouleau, New complexity results on scheduling with small communication delays. *Disc. Appl. Math.* **60** (1995) 331–342.

[22] V.J. Rayward-Smith, UET scheduling with unit interprocessor communication delays. *Disc. Appl. Math.* **18** (1987) 55–71.

[23] O. Sinnen, *Task Scheduling for Parallel System.* Chap. 7, Wiley (2007).