Hindawi Publishing Corporation
Advances in Operations Research
Volume 2011, Article ID 476939, 20 pages
doi:10.1155/2011/476939

Research Article

Inapproximability and Polynomial-Time Approximation Algorithm for UET Tasks on Structured Processor Networks

M. Bouznif[1] and R. Giroudeau[2]

[1] *Laboratoire G-SCOP, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1, France*
[2] *LIRMM, 161 rue Ada, UMR 5056, 34392 Montpellier Cedex 5, France*

Correspondence should be addressed to R. Giroudeau, rgirou@lirmm.fr

Received 26 October 2010; Revised 22 March 2011; Accepted 4 April 2011

Academic Editor: Ching-Jong Liao

We investigate complexity and approximation results on a processor networks where the communication delay depends on the distance between the processors performing tasks. We then prove that there is no heuristic with a performance guarantee smaller than 4/3 for makespan minimization for precedence graph on a large class of processor networks like hypercube, grid, torus, and so forth, with a fixed diameter $\delta \in \mathbb{N}$. We extend complexity results when the precedence graph is a bipartite graph. We also design an efficient polynomial-time $O(\delta^2)$-approximation algorithm for the makespan minimization on processor networks with diameter δ.

1. Introduction

1.1. Problem Statement

In this paper, we consider the processor network model, which is a generalization of the `homogeneous scheduling delay model` in which task allocation on the processors does not have any influence over the length of scheduling. Indeed, since the graph of processors (denoted hereafter $G^* = (V^*, E^*)$ where $V^* = \{\pi^1, \ldots, \pi^m\}$ is a set of m processors and E^* is the set relationship between them) is fully connected, the starting of a task i depends only on the potential communication delay, given by precedence graph between i and its own predecessors.

In the processor network model, this assumption is relaxed in order to take into account the fact that the processor graph may not be fully connected. Thus, task allocation on the processors can be expressed by its essential and fundamentals characteristics. We

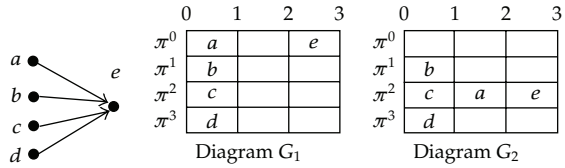


Figure 1: Difference between the problem $P|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ and $(P, \text{grid } 2 \times 2)|\text{prec}; c_{ij} = d(\pi^i, \pi^j); p_i = 1|C_{\max}$.

consider a model in which a distance function (which is defined hereafter), denoted $d(\pi^l, \pi^h)$ between two processors π^l and π^h in the graph of processors impacts computation of the communication delay between two tasks i and j (subject to a precedence constraint) and consequently on the starting time of task j. The communication time, using c_{i,π^l,j,π^h} for computing the starting time of a task (this notation indicates that the value of the communication delay between task i, which is allotted to processor π^l and task j which will be executed on the processor π^h), is assumed as $c_{ij}d(\pi^l, \pi^h)$, where c_{ij} is the communication delay given by the precedence graph.

Formally, the processor network model may be defined as

$$\forall (i,j) \in E, t_j \geq t_i + p_i + c_{ij} d\left(\pi^{\ell}, \pi^h\right), \tag{1.1}$$

where π^{ℓ} (resp. π^h) represents the processor on which task i (resp. task j) 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^* (the graph of processor $G^* = (V^*, E^*)$) between π^{ℓ} and π^h, and c_{ij} represents the communication delay if two tasks are executed on two neighboring processors (this value is given by the precedence graph).

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 structured graph with a diameter δ. In these topologies, processors are numbered as $\pi^1, \pi^2, \ldots, \pi^m$ and processor π^h may be communicated with processor π^l with a communication cost equal to $d(\pi^h, \pi^l)$ where $d(\pi^h, \pi^l)$ represents the shortest path on graph G^* between processors π^h and π^l. The communication delay is therefore the distance function proposed above.

In scheduling theory, a problem type is categorized by its machine environment, job characteristic, and objective function. Thus, using the *three fields* notation scheme $\alpha|\beta|\gamma$,(where α designates the environment processors, β the characteristics of the job, and γ the criteria.) proposed by Graham et al. [1], we consider the problem of makespan minimization (denoted in follows by $C_{\max}$) with unitary task and unitary communication delay (UET-UCT) in presence of a precedence graph G on a processors network having a graph G^* such that the communication delay depends on the shortest path on graph G^*. This problem is denoted by $(P, G^*)|\text{prec}; c_{ij} = d(\pi^{\ell}, \pi^k); p_i = 1|C_{\max}$.

Example 1.1. Figure 1 shows the difference between the two problems $P|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ and $(P, \text{grid } 2 \times 2)|\text{prec}; c_{ij} = d(\pi^{\ell}, \pi^k); p_i = 1|C_{\max}$. (The relationship between processors is as follows: π^0 and π^3 are connected to π^1 and π^2.) The processing time of the tasks and the communication delay between the tasks are unitary (UET-UCT problem). Gantt diagram G_1 represents an optimal solution for the $P|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ problem. We

Table 1: Previous complexity results on the processors network model.

Topology	Precedence graph	Complexity	Reference
Unbounded chain	Tree	$\mathcal{NP}$-complete	[2]
	Antitree	$\mathcal{NP}$-complete	
Star	Tree	$\mathcal{NP}$-complete	[2]
Cycle/chain	Prec	$\rho \geq 4/3$	[4]
Star	Prec	$\rho \geq 6/5$	[3]

can notice that task z can be executed on any processor at $t = 2$. Moreover, Gantt diagram G_2 represents an optimal solution for the problem $(P, \text{grid } 2\times 2)|\text{prec}; c_{ij} = d(\pi^{\ell}, \pi^{k}); p_i = 1|C_{\max}$. In order to obtain an optimal solution, the task a must be delayed by one unit of time and must be processed on the same processor π^2 as task c at $t = 1$. Thus, task e may be executed at $t = 2$ only on the processor π^2.

1.2. Organization of the Paper

This paper is organized as follows: the next section is devoted to the related works. In Section 3, after defining the class graph $\mathcal{G}$ we propose a general nonapproximability result for a nonspecified precedence graph. We also extend the previous result when the precedence graph is a bipartite graph and when the duplication is allowed. In the last section, we design a polynomial-time approximation algorithm with a performance ratio within $O(\delta)$.

2. Related Works

2.1. Complexity Results

To the best of our knowledge, the first complexity result was given by Picouleau [2]. 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 (a star is a tree of depth one) topology. Picouleau proved that this problem is $\mathcal{NP}$-complete if the precedence graph is a tree or an outtree. Recently in [3], the authors proved that there is no heuristic with a performance guarantee smaller than $6/5$ for minimizing the makespan on a processor network represented by a star. This model is closest to the master-slave architecture. In [4], the authors proved that there is no hope to finding a polynomial-time approximation algorithm with a ratio $\rho > 4/3$ for the problem to schedule a set of tasks on a ring or a chain as processors network (see Table 1).

2.1.1. Approximation Results

In ring topology, Lahlou developed, in [5], using the list scheduling proposed by Rayward-Smith [6], a ρ-approximation algorithm with $\lceil\sqrt{m}\rceil \leq \rho \leq 1 + (3/8)m - 1/2m$ where m is the number of processors.

Moreover, Hwang et al. [7] studied approximation list algorithms for scheduling problems where the communication times depend on contention and a distance function

for the tasks involved and on the processors that execute the tasks. The 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 improved a strategy called *earliest task first*.

Recently, in [3] 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 networks as a star forms. In [4] the authors develop two polynomial-time approximation algorithms for processor networks with limited or unlimited resources.

2.2. Our Contributions

In this paper, we answer the following interesting question: *is there a large class of graphs, for which it exists a polynomial-time reduction from* n-PARTITION, *to show the* $\mathcal{NP}$*-completeness? Therefore, it is sufficient to show if the graph* G *is belonging to this class, in order to prove the nonexistence of* $\mathcal{PTAS}$*?* In order to complete the study of processor networks, we design a polynomial-time approximation algorithm within a ratio at most $((\delta + 1)^2/3) + 1$ where δ designates the diameter of the graph G^*.

3. Computational Complexity for a Large Class of Graph

3.1. The Class Graph $\mathcal{G}$

We propose a large class of graph $\mathcal{G}$ for which the problem of deciding whether an instance $(P, G^*)|\text{prec}; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max} \leq 3$ is $\mathcal{NP}$-complete.

We present now a graph class for which we may apply the same polynomial-time transformation mechanism from 3-PARTITION problem to show that our scheduling problem when processor networks belong to this class is $\mathcal{NP}$-complete. Hereafter, we give the definition of the prism graph.

Definition 3.1. A prism $P = (V_P, E_P)$ of size k and length L $(k, L \in \mathbb{N})$ is a connected undirected graph for that

(i) there are two sets of vertices K and K' such as $K \subset V_P$, $K' \subset V_P \setminus \{K\}$, and $|K| = |K'| = k$. The vertices are denoted $s_1, \ldots, s_k$ (resp. $s'_1, \ldots, s'_k$);

(ii) it exists an order on K and K' vertices such that $(\forall s_i \in K, s'_i \in K, 1 \leq i \leq k)$ there is a path of length L denoted C_i between s_i and s'_i;

(iii) $(i \neq j) \wedge x \in C_i \setminus \{s_i, s'_i\} \wedge y \in C_j \setminus \{s_j, s'_j\} \Rightarrow (x, y) \notin E_P$.

Moreover, the size of a prism is polynomial in k. An illustration is given in Figure 2.

Definition 3.2. Let $\mathcal{G}$ be a collection of graphs. $\mathcal{G}$ possess the prism property if and only if $\forall n_0, \forall n_1 \in \mathbb{N} \exists G \in \mathcal{G}$, such that G contains *a unique* subgraph $G_1 = (V_1, E_1)$ of G induced by vertices $V_1 \subset V$ with a prism of size $k = n_0$ and length $L = n_1$.

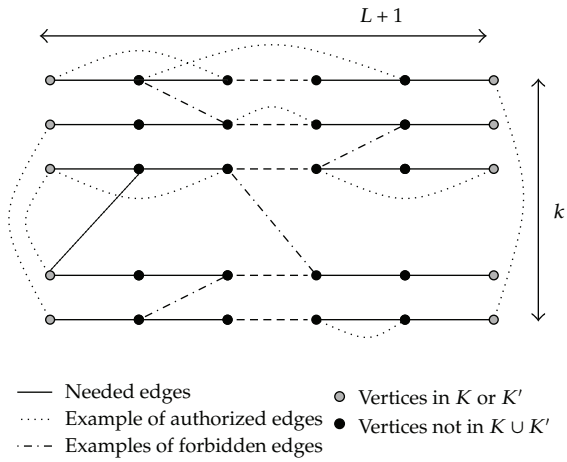


Figure 2: An example of a prism of size k and length L.

Lemma 3.3. *The class graph $\mathcal{G}$ is not empty.*

Proof. In particular we will see in Section 3.2 classic structured graph like torus, grid, complete binary tree, and so forth, belonging to this class graph. □

Theorem 3.4. *The problem of deciding whether an instance of $(P, G^*)|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ has a schedule of length at most two is polynomial with $\beta \in \{prec, bipartite\}$ and $G^* \in \mathcal{G}$.*

Proof. No communication is allowed between two pairs of tasks. □

The remainder of this section is devoted to proving Theorem 3.5.

Theorem 3.5. *The problem of deciding whether an instance of $(P, G^*)|prec; c_{ij} = d(\pi^k, \pi^l); p_i = 1|C_{\max}$ has a schedule of length at most three is $\mathcal{NP}$-complete with $G^* \in \mathcal{G}$.*

Proof. The proof is established by a reduction of the 3-PARTITION problem [8].

Instance

A finite set $\mathcal{A}$ of $3M$ elements $\{a_1, \ldots, a_{3M}\}$, 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 1. Can A be partitioned into M disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_M$ of $\mathcal{A}$ such that for all $i \in [1, \ldots, M]$, $B = \sum_{a\in\mathcal{A}_i} s(a) = \sum_{a\in\mathcal{A}} s(a)/M \in \mathbb{N}$?

3-PARTITION is known to be $\mathcal{NP}$-complete in the strong sense [8]. (Even if B is polynomially bounded by the instance size, the problem is still $\mathcal{NP}$-complete.)

It is easy to see that $(P, G^*)|\text{prec}, c_{ij} = d(\pi^l, \pi^k) = 1, p_i = 1|C_{\max} \leq 3 \in \mathcal{NP}$.

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

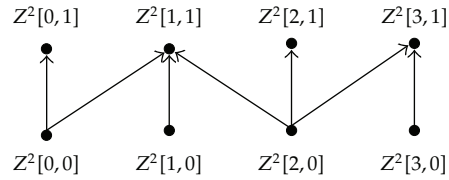


Figure 3: Graph Z^2.

The precedence graph $\widetilde{G} = \mathcal{W} + \mathcal{Z}$, which will be scheduled on the processors network G^*, is decomposed into two disjointed graphs, denoted as follows by $\mathcal{W}$ and $\mathcal{Z}$ (the graph $\mathcal{Z}$ is a collection of graphs $Z^{s(a_j)}$, i.e., $\mathcal{Z} = \cup_{a_j \in \mathcal{A}} Z^{s(a_j)}$). Hereafter, graphs $\mathcal{Z}$ and $\mathcal{W}$ are characterized. □

Graph Z^i

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

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

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

(iii) arcs $Z^i[2j,0] \rightarrow Z^i[2j-1,1]$ for any j, $1 \leq j \leq i-1$.

Remark 3.6. 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, $0 \leq j \leq 2i-1$,

(i) tasks $Z^i[j,0]$ and $Z^i[j,1]$ are executed on π^j,

(ii) tasks $Z^i[j,\ell]$ are executed at time ℓ, for any $\ell \in \{0,1\}$, if j is even,

(iii) tasks $Z^i[j,\ell]$ are otherwise executed at time $\ell+1$, for any $\ell \in \{0,1\}$.

See Figure 3 for graph Z^2 and Figure 4 for the valid scheduling described in Remark 3.6.

Graph $\mathcal{W}$

Remark 3.7. A path of length l admits $l+1$ vertices.

The $\mathcal{W} = (\mathcal{U} \cup \mathcal{U}'; E_{\mathcal{W}})$ graph will be defined as follows. Let $G^* = (V^*, E^*)$ be a graph such that $G^* \in \mathcal{G}$, with $V^* = \{v_1^*, \ldots, v_{n^*}^*\}$. By Definition 3.2, we know that it exists a unique subgraph $G = (V \subset V^*, E \subset E^*)$ of size k and length L with desired properties. In the following we set $k = n$ and $L = 2B+1$ and the size of $G^* = (V^*, E^*)$ is polynomial in k. Note that $n^* \gg 2B$.

The $\mathcal{W}$-graph is defined by polynomial-time transformations from the G^*-graph. The graph given in Figure 5 will be used to illustrated the following construction.

(i) The paths of length three are created and precedence constraints are added (see Figure 6). The two sets of tasks $\mathcal{U}_1$ and $\mathcal{U}'$ are created.

(ii) The tasks are partitioned into three subsets $\mathcal{U}'$, $\mathcal{K}$, and $\mathcal{U}$ (see Figure 7).

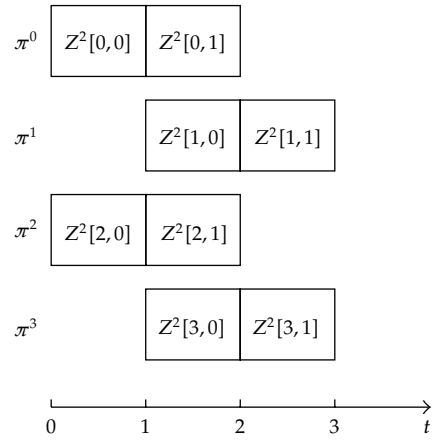


Figure 4: Valid schedule of length three for graph Z^2.

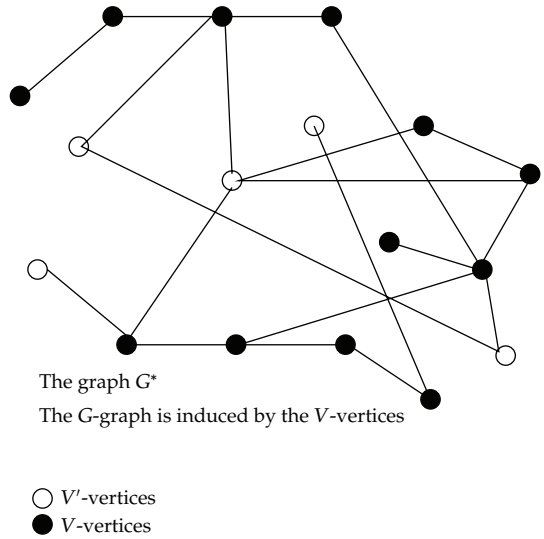


Figure 5: The beginning of the construction of $\mathcal{W}$ graph from $G^* \in \mathcal{G}$.

(iii) The $\mathcal{U}_1$-tasks are now partitioned into two subsets $\mathcal{K}$ and $\mathcal{U}$. We consider the subgraph induced by the $\mathcal{U} \cup \mathcal{U}'$-tasks (see Figure 8) as the $\mathcal{W}$–graph.

The purpose of removing these tasks is to allow the tasks of $\mathcal{K}$-graph when the tasks of $\mathcal{W}$-graph, deprived of these tasks, will be executed on the graph of processors.

The set of vertices V^* is partitioned into two sets $V^* = V' \cup V$:

(i) $V = \{v_1^*, \ldots, v_{2n(B+1)}^*\}$ the vertices of G, and defined the vertices of the n unique paths of length $(2B + 1)$ respecting the characteristics given by Definition 3.1,

(ii) $V' = \{v_{2n(B+1)+1}^*, \ldots, v_{n^*}^*\}$, the set of an other vertices. Note that these vertices do not belong to G graph.

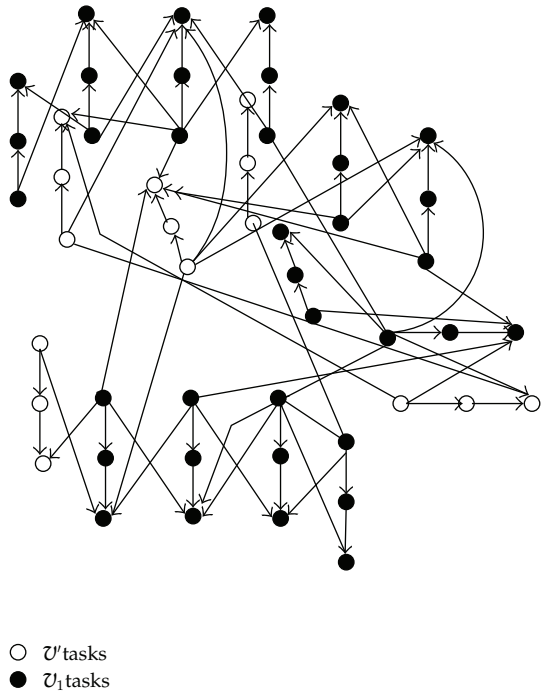


Figure 6: Next step of the construction of $\mathcal{W}$ graph. Path of length three is created and precedence constraints between tasks are added.

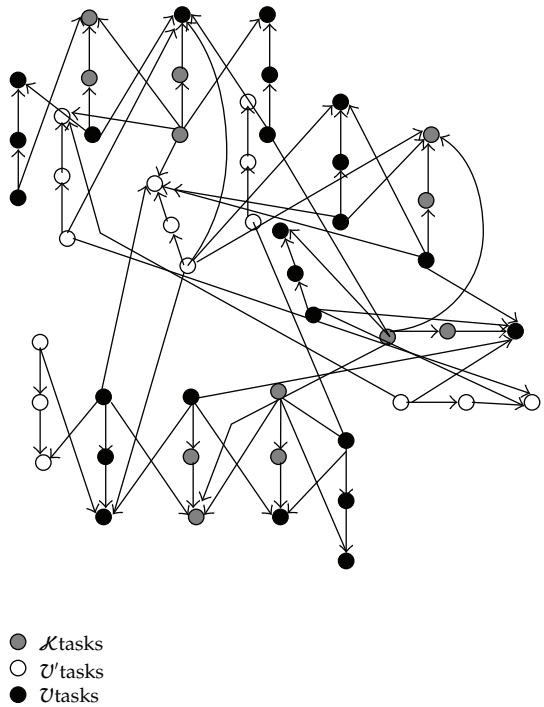


Figure 7: Partition of G^* graph into tasks sets $\mathcal{V}$, $\mathcal{K}$, and $\mathcal{V}$.

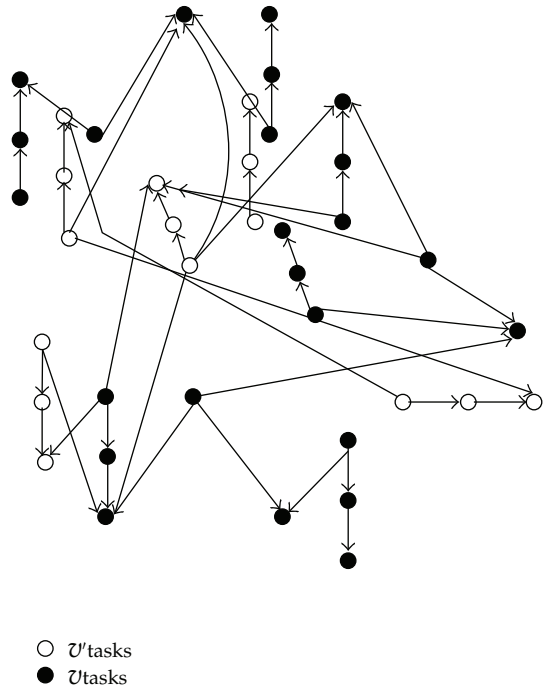


Figure 8: The final $\mathcal{W}$ graph issue from several transformations.

The definition of the $\mathcal{W}$ graph is given below.

(i) $\forall i \in \{1, \ldots, 2n(B+1)\}$, we create a path of length three $v_i^*[0], v_i^*[1]$, and $v_i^*[2]$, with edges $v_i^*[0] \rightarrow v_i^*[1] \rightarrow v_i^*[2]$. The set of tasks will be denoted $\mathcal{U}_1 = \{v_i^*[j] | \ \forall i \in \{1, \ldots, 2n(B+1)\}, j \in \{0,1,2\}\}$. The cardinality of $\mathcal{U}_1$ is $6n(B+1)$ (see Figure 6).

(ii) $\forall i \in \{2n(B+1)+1, \ldots, n^*\}$, we create a path of length three $v_i^*[0] \rightarrow v_i^*[1] \rightarrow v_i^*[2]$. This set of tasks will be denoted $\mathcal{U}'$. The number of tasks is $3(n^* - 2n(B+1))$ with $n^* = |V^*|$.

(iii) $(k,l) \in E^*$, we add the edges $v_k^*[0] \rightarrow v_l^*[2]$ and $v_l^*[0] \rightarrow v_k^*[2]$ (see Figure 6).

Now, $4nB$ tasks are removed from $\mathcal{W}$-graph. (In order to clarify the polynomial-time transformation, we give priority to create tasks and remove some ones instead of enumerating all precedence constraints.) Therefore, we consider the following index sets:

(i) $J_0 = \{2i(B+1) \mid i = \{1, 2, \ldots, n\}\}$,

(ii) $J_1 = \{2i(B+1)+1 \mid i \in \{0, 1, 2, \ldots, n-1\}$,

(iii) $I_0 = \{k \in \{1, \ldots, 2n(B+1)\} \setminus \{J_0 \cup J_1\}$ and $|k$ is even$\}$,

(iv) $I_1 = \{k \in \{1, \ldots, 2n(B+1)\} \setminus \{J_0 \cup J_1\}$ and $|k$ is odd$\}$.

We remove from the $\mathcal{U}_1$-set the following tasks $v_k^*[0]$, $v_k^*[1]$ with $k \in I_0$, (resp. $v_k^*[1]$, $v_k^*[2]$ with $k \in I_1$). $\mathcal{K}$ denotes the set of removed tasks (see Figure 7). Finally, we put $\mathcal{U} = \mathcal{U}_1 \setminus \mathcal{K}$ with $|\mathcal{U}| = 2nB + 6n$ (see Figure 8).

Figures 5, 6, 7, and 8 describe the construction of $\mathcal{W}$-graph from $G^* \in \mathcal{G}$.

$E_{\mathcal{W}}$ is the set of arcs as described above.

Lastly, the number of processors is $m = n^*$, and they are numbered as π^i with $i \in [1, n^*]$.

In summary the precedence graph $\widetilde{G} = \mathcal{W} + \mathcal{Z}$ is composed by $\mathcal{W} = (\mathcal{U} \cup \mathcal{U}', E_{\mathcal{W}})$ with $3n^* - 4nB$ tasks and the precedence constraints given before and the graph $\mathcal{Z} = \{\bigcup_{a_j \in \mathcal{A}} Z^{s(a_i)}\}$ with $4nB$ tasks.

The transformation is computed in polynomial time.

(i) Let us assume that $\mathcal{A} = \{a_1, \ldots, a_{3M}\}$ can be partitioned into M disjoint subsets $\mathcal{A}_1, \ldots, \mathcal{A}_M$ 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.

First, the task $v_i^*[j] \in \mathcal{U}' \cup \mathcal{U}$ is executed on the processors π^i to $t = j$ with $j \in \{0, 1, 2\}$ (if this task exists).

Consider the processors on which the set of $\mathcal{U}$-tasks are scheduled. By the previous allocation, these processors are numbered as $\pi^1, \ldots, \pi^{2n(B+1)}$.

Let $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be a partition of $\mathcal{A}$. Consider $\mathcal{A}_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$ with a fixed i. The tasks of $Z^{s(a_j)}$, $a_j \in \mathcal{A}_i$ are executed between processors $\pi^{1+2(i-1)(B+1)}$ and $\pi^{2i(B+1)}$. Moreover, the tasks $Z^{s(a_j)}[l, k]$, $k \in \{0, 1\}$, $l \in J_0$ (resp., $k \in \{1, 2\}$, $l \in J_1$) are scheduled on $2s(a_{i_j})$ processors in succession in order to respect a schedule of length three.

Thus without loss of generality, we suppose that the tasks of $Z^{s(a_{i_1})}$ are scheduled between processors $\pi^{1+2(i-1)(B+1)}$ and $\pi^{2(i-1)(B+1)+2s(a_{i_1})}$. In similar way, the tasks $Z^{s(a_{i_2})}$ (resp., $Z^{s(a_{i_3})}$) are executed between processors $\pi^{2+2(i-1)(B+1)+2s(a_{i_1})}$ and $\pi^{1+2(i-1)(B+1)+2s(a_{i_1})+2s(a_{i_2})}$ (resp. $\pi^{2+2(i-1)(B+1)+2s(a_{i_1})+2s(a_{i_2})}$ and $\pi^{2i(B+1)}$).

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

Lemma 3.8. *In any valid schedule of length three there is no idle time.*

Proof. The number of processors is $m = n^*$ and the number of tasks is $3n^*$ ($4nB$ for $\mathcal{Z}$-graph and $3n^* - 4nB$ for $\mathcal{W}$ graph). □

Lemma 3.9. *In any valid schedule of length three, the subgraph induced by $\mathcal{U}$ tasks must be executed on $2(B + 1)$ processors in succession.*

Proof. Consider the subgraph induced by the $\mathcal{U}$ tasks. This precedence graph admits paths of length two and these paths must be executed on the same processor (no communication delay is allowed).

Consider the tasks of path of length one. Let $v_i^*[0] \in \mathcal{U}$ be a task without predecessor. By construction $v_i^*[0]$ admits one successor denoted by $v_{i+1}^*[2] \in \mathcal{U}$.

Suppose that these two tasks are allotted on the same processor π^l. Since that $v_{i+1}^*[2]$ admits another predecessor denoted by $v_{i+2}^*[0] \in \mathcal{U}$ then $v_{i+1}^*[2]$ is allotted at $t = 2$.

The task $v_{i+2}^*[0]$ cannot be executed at $t = 1$ on π^l since this task admits another successor as $v_{i+1}^*[2]$. Therefore, it exists an idle slot at $t = 1$ on the processor π^l. By construction there is no independent task and since the $\mathcal{Z}$ graph admits only path of length one, then no task can be allotted on this idle slot. This is impossible

In conclusion, the subgraph induced by $\mathcal{U}$ tasks must be executed on $2(B+1)$ processors in succession. □

Lemma 3.10. *In any valid schedule of length three, two subgraphs induced by the $\mathcal{U}$ tasks from two disjoint paths of length $2(B+1)$ cannot be allotted on the same processors.*

Proof. Consider the $\mathcal{U}$ tasks which are elements of two disjoints paths of length $2(B+1)$. A task without predecessor of one path cannot be allotted on the same processor as a task without successor of other path since there is no isolated task to schedule. □

Lemma 3.11. *In any valid valid schedule of length three the $Z^{s(a_j)}$ tasks must be executed on the same processors as the $\mathcal{U}$ tasks.*

Proof. Let $\Pi = \{\pi^l \mid \mathcal{U}$ tasks allotted on $\pi^l\}$ be the set of processors on which the $\mathcal{U}$ tasks are executed.

Suppose that the $Z^{s(a_j)}$-tasks are executed on processors $\pi^k \notin \Pi$. By Lemma 3.8, there is no idle slot, then the tasks on the path of length three are necessarily allotted on processor $\pi^* \in \Pi$. This is impossible by Lemma 3.9. □

With previous lemmas, we know that $6n(B+1)$ tasks (the $\mathcal{U}$ tasks and the $Z^{s(a_j)}$-tasks) are executed on the n disjoints paths of length $2B+1$. By Definition 3.2, we know that the graph G^* admits a unique set of n disjoints paths of length $2B+1$ with desired properties. Moreover with the precedence constraints, these tasks are allotted on a processor path of length $(2B+1)$. Without loss of generality, we suppose that a task v_lßU is executed on the processor π^l with $l \in \{2n(B+1)+1, \ldots, n^*\}$.

Building the partition $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ with desired property from S schedule of length three, we know that two tasks of the same subgraph $Z^{s(a_j)}$ (see Lemma 3.11) cannot be executed on two different paths. The edge distance between these two processors is at least two.

We define $\mathcal{A}_\ell$ such that $a_j \in \mathcal{A}_\ell$ if and only if the tasks of the graph $Z^{s(a_j)}$ are executed between the processors numbered as $\pi^{1+(j-1)2(B+1)}$ to $\pi^{2j(B+1)}$ with a fixed j.

Now, we will compute $\sum_{a_i \in \mathcal{A}_\ell} s(a_i)$.

Using previous remarks, without loss of generality, we suppose that $v_i^*[k]$ with $i \in \{1, \ldots, 2n(B+1)\}$ and $k \in \{0,1,2\}$ (if it exists) are executed on π^l with $l \in \{1, \ldots, 2n(B+1)\}$. Consider the $Z^{s(a_j)}$-tasks which are scheduled between processors $\pi^{1+(j-1)2(B+1)}$ and $\pi^{2j(B+1)}$ for a fixed $j \in \{1, \ldots, 2n(B+1)\}$ except the index such that paths of length three constituted by tasks from $\mathcal{U}$, are allotted on π^l.

Using Lemma 3.9, we know that the number of $\mathcal{U}$ tasks executed on processors $\pi^{1+(j-1)2(B+1)}$ and $\pi^{2j(B+1)}$ for a fixed j is $6+2B$.

In conclusion we have $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ which forms a $\mathcal{A}$ with desired properties.

The construction suggested previously can be easily adapted to obtain a bipartite graph of depth one. Moreover, from the proof of Theorem 3.5, we can derive the following theorem.

Theorem 3.12. *The problem of deciding whether an instance of $(P, G^*)|\beta, c_{ij} = d(\pi^\ell, \pi^k) = 1, p_i = 1|C_{\max}$ has a schedule of length at most three is $\mathcal{NP}$-complete with $\beta \in \{prec, bipartite\}$.*

Proof. The proof is similar as the proof of Theorem 3.5 by considering the graph $\widetilde{G}'$ instead of widget $\widetilde{G}$. Nevertheless each path of length two induced by the $\mathcal{U}$ tasks is transformed into two paths of length one.

We use the same construction as it is proposed for the proof of Theorem 3.5. Nevertheless, all paths of length three are transformed into two paths in the following way: $v_i^*[0] \rightarrow v_i^*[1]$ and $v_i^*[0] \rightarrow v_i^*[2]$. These three must be executed on the same processors.

Indeed, if $v_i^*[2]$ admits several predecessors, it is obvious. Otherwise, suppose that $v_i^*[0]$ is allotted on a processor π. So $v_i^*[1]$ must be executed at $t = 1$ on π. The task $v_i^*[2]$ is scheduled at $t = 2$ on a neighborhood processor. Therefore no task from the graphs $\mathcal{R}$ and $\widetilde{G}'$ can be executed on processor π at $t = 2$. Now using the same arguments as previously there is a schedule of length three if and only if the set $\mathcal{A} = \{a_1, \ldots, a_{3n}\}$ can be partitioned into n disjoint subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ each summing up to B. □

The proof of Theorem 3.5 therefore implies that the problem where the tasks can be duplicated is also $\mathcal{NP}$-complete.

Corollary 3.13. *The problem of deciding whether an instance of* $(P, G^*)|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1, dup|C_{\max}$ *with* $G^* \in \mathcal{G}$ *has a schedule of length at most three is* $\mathcal{NP}$*-complete with* $\beta \in \{prec, bipartite\}$.

Proof. The proof comes directly from Theorems 3.5 and 3.12. In fact, Lemma 3.8 implies that no task can be duplicated (the number of the tasks is equal to the number of processors times 3). □

Moreover, nonapproximability results can be deduced.

Corollary 3.14. *No polynomial-time algorithm exists with a performance bound less than 4/3 unless* $\mathcal{P} = \mathcal{NP}$ *for the problems* $(P, G^*)|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ *and* $(P, G^*)|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1, dup|C_{\max}\beta \in \{prec, bipartite\}$ *with* $G^* \in \mathcal{G}$.

Proof. The proof of Corollary 3.14 is an immediate consequence of the impossibility theorem; see [9, page 4]. □

3.2. Discussion

In the previous section, we propose a class graph $\mathcal{G}$ for which the problem of deciding whether an instance of $(P, G^*)|\beta; c_{ij} = d(\pi^\ell, \pi^k); p_i = 1|C_{\max}$ has a schedule of length at most three is $\mathcal{NP}$-complete with $\beta \in \{\text{prec}, \text{bipartite}\}$ and $G^* \in \mathcal{G}$.

Hereafter, we will exhibit the parameters (L, k) for some classic structured graphs in order to prove that the class graph $\mathcal{G}$ is not empty.

(i) For a grid $G^* = \text{Grid}(m, p)$ $(m, p \in \mathbb{N}$, where the couple (i, j) designates the j the position in the i the line; $1 \leq i \leq m, 1 \leq j \leq p)$ (or torus) topology, we need $k = 2n + 1$ lines and $L = 2B + 2$ columns. The set of vertices for the graph G a subgraph of G^* with the desired properties given by Definition 3.2 is $V = \{(i, j),\ 2 \leq i \leq 2n, i \text{ even}, 2 \leq j \leq 2B + 3\}$ and $V' = \{(i, 1), 1 \leq i \leq 2n + 1\} \cup \{(i, j), 1 \leq i \leq 2n + 1, i \text{ odd}; 1 \leq j \leq 2B + 3\}$.

(ii) For the complete binary tree, it is sufficient to consider a tree with height of $\lceil \log(n) \rceil + 2B + 1$.

(iii) For the Hypercube $H(d)$ topology (or cube connected cycles), it is sufficient to have $d = 2\lceil \log(n) \rceil + B + 2$.

(iv) ….

4. An Approximation Algorithm for Processor Networks with a Fixed Diameter

4.1. Description and Correctness of an Algorithm

In order to design an efficient polynomial-time approximation algorithm, the classic strategy consists of taking an instance of the combinatorial optimization problem and applying some transformations and/or using polynomial-time algorithms as subroutines (shortest path, spanning tree, maximum matching, etc.). Afterwards, it is sufficient to evaluate the best lower bound for any optimal solution, and this lower bound may be compared to the feasible solution for the combinatorial optimization problem in order to determine the ratio of an approximation algorithm.

Here, instead of considering an instance I and trying to directly develop a feasible solution for the $(P, G^*)|\text{prec}; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{\max}$ problem, we consider a partial instance of I of our scheduling problem (An instance I is constituted by a precedence graph with unit execution time and unit communication time, m processors in G graph form, with the distance function.), denoted I^*. (The partial instance I^* of I is constituted only by the precedence graph with unitary tasks and unitary communication time) For any instance I^*, we use the classic approximation algorithm proposed by Munier and König [10] for the $\overline{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ problem. We obtain a feasible schedule, denoted S (we omit consideration of the processor graph for the moment) for the previous problem. Nevertheless, this solution is not feasible for our scheduling problem.

We proceed with polynomial-time chain of transformations, from schedule S to a schedule S'', in order to get a feasible schedule. It is only in the last step, only for schedule S'', that we guarantee a feasible schedule for the problem $(P, G^*)|\text{prec}; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{\max}$.

This chain is defined as follows: $I^* \xrightarrow{f} S \xrightarrow{g} S' \xrightarrow{h} S''$ (The schedule S' is a feasible solution for the$\{\overline{P}, G\}|\text{prec}; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{max}$ problem.) , where f is the Munier-König algorithm [10], g the dilatation algorithm (see [11] for details or Appendix A) and h the folding algorithm (see [12] for details or Appendix B).

Subsequently, we will consider the three following scheduling problems:

(i) $\overline{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$,

(ii) $\overline{P}|\text{prec}; c_{ij} \geq 2; p_i = 1|C_{\max}$,

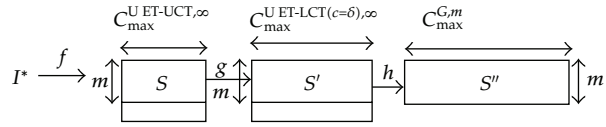
(iii) and finally $(P, G^*)|\text{prec}; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{\max}$.

The principal steps of the algorithm are described below.

An approximation algorithm uses three steps. In each step we apply an algorithm for a specified scheduling problem [10–12]. In the two first steps, a schedule is produced (these schedules are not feasible for our problem).

(i) In the first step of an algorithm, a schedule (denoted S on an unbounded number of processors), for the scheduling problem $\overline{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ is produced. For this problem, Munier and König [10] presented a (4/3)-approximation algorithm that is based on an integer linear programming formulation. They use the following procedure: an integrity constraint is relaxed, and a feasible schedule is produced by rounding.

(ii) The second step of an algorithm produces a schedule (denoted S', also on an unbounded number of processors) from S by applying the dilatation principle



It is clear that $C_{\max}^{\text{UET-UCT},\infty} \leq C_{\max}^{\text{UET-LCT}(c=\delta)\infty} \leq C_{\max}^{G,m}$

Figure 9: Description of chain of polynomial-time transformations.

proposed by [11] for the problem $\overline{P}|\text{prec}; c_{ij} \geq 2; p_i = 1|C_{\max}$ (this algorithm produces a feasible schedule for the large communication delay problem from unitary communication delay. We therefore have $S' = g(S)$ where g is the dilatation algorithm.

(iii) The third step produces a schedule S'' (feasible for the $(P, G^*)|\text{prec}, c_{ij} = d(\pi^k, \pi^l) = 1, p_i = 1|C_{\max}$ problem) on the G topology from S' using the folding principle [12]. The folding procedure constructs a feasible schedule on restricted number of processors from a feasible schedule on an unbounded number of processors. Thus, $S'' = h(S')$ with h being the folding algorithm.

Note that the length of schedule S is less than S', which is less than S''. The three steps are summarized in Figure 9. The notation description is given in the proof of Theorem 4.2.

Theorem 4.1. *The previous algorithm leads a feasible schedule for the problem* $(P, G^*|prec; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{\max}$.

Proof. Proof is clear from the previous discussion concerning the description of an algorithm. Indeed, the communication delay is preserved and the precedence constraint is respected. Moreover, at most m tasks are executed at any time. □

4.2. Relative Performance Analysis

Theorem 4.2. *The problem* $(P, G^*)|prec; c_{ij} = d(\pi^k, \pi^l) = 1; p_i = 1|C_{\max}$ *may be approximable within a factor of* $((\delta + 1)^2/3) + 1$ *using the previous algorithm.*

Proof. We denote using $C_{\max}^{x,y,z}$ with $x \in \{\text{opt}, \emptyset\}$, $y \in \{\text{UET-UCT}, \text{UET-LCT(c} = \delta), G^*\}$, and $z \in \{m, \infty\}$ the length of the schedule. Moreover $\rho^{G^*,m}$ (resp., $\rho^{G^*,\infty}$) designates the performance ratio on a G processor network model with a bounded (resp., unbounded) number of processors.

Now let us examine the relative performance of this algorithm.

(i) According to an algorithm, the first step deals with the problem $\overline{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$.

First of all the *Schedule* (UET-UCT,∞) is not optimal. Using the algorithm from [10] gives us a 4/3 relative performance. And so, by [10], we know that

$$C_{\max}^{\text{UET-UCT},\infty} \leq \frac{4}{3} C_{\max}^{\text{opt,UET-UCT},\infty}. \tag{4.1}$$

(ii) In the second step, a feasible solution for a large communication delay $c = \delta$ (recall that δ stands for the diameter of processors network) is created. This solution comes from using the dilatation algorithm. Then, the expansion coefficient is $(\delta + 1)/2$ ([11]). And so,

$$C_{\max}^{\text{UET-LCT}(c=\delta),\infty} \leq \frac{\delta + 1}{2} \cdot \frac{4}{3} C_{\max}^{\text{opt,UET-LCT(c=}\delta),\infty}, \tag{4.2}$$

$$C_{\max}^{\text{UET-LCT}(c=\delta),\infty} \leq \frac{2(\delta + 1)}{3} C_{\max}^{\text{opt,UET-LCT(c=}\delta),\infty}. \tag{4.3}$$

Thus, we have a schedule on a UET-LCT task system with a communication delay equal to δ and an infinite number of processors.

By definition it is obvious that

$$C_{\max}^{G^*,\infty} \leq C_{\max}^{\text{UET-LCT}(c=\delta),\infty}, \tag{4.4}$$

$$C_{\max}^{\text{opt,UET-UCT},\infty} \leq C_{\max}^{\text{opt,UET-LCT(c=}\delta),\infty}. \tag{4.5}$$

It is necessary to evaluate the gap between the optimal length for the schedule on a fully connected processor graph and a processor graph with a diameter of length K. For this, we consider unitary tasks subject to precedence constraints and an unbounded number of processors. □

Lemma 4.3. *The gap between a schedule on a fully connected graph of processors with a large communication delay c, for all pairs of tasks, and a schedule on a graph of processors with a diameter of length* $K \in \mathbb{N}$, *is at most* $(c + 1)/2$.

Proof. We need to compare first the relative performance of this schedule on our model with network processor. The relative performance for the UET-LCT task system is not valid for our model. We need to compute a new bound for this schedule on our model.

Let $p = \{x_1, x_2, \ldots, x_n\}$ be a critical path of the schedule (i.e., a path that gives the length of the schedule). Suppose that there is a communication delay between each pair of tasks (x_i, x_{i+1}) with $1 \leq i < n$. In the UET-LCT task system (with a communication delay equal to c for all pair of tasks) the length of the schedule would be $(1 + c)n - c$ units of time. In the graph of processors with a diameter of length k, the same path allows a length of $(k/2)(n-1) + n$ units of time. The worst case of the length for this path is $n + (n-1)k$ and the best case is $2n - 1$. So, the ratio is $(n(1 + c) - c)/(2n - 1)$. For the large n, we obtain the desired result. □

By applying Lemma 4.3, which is valid for all schedules, and in particular for the optimum, with $c = \delta$, we obtain

$$C_{\max}^{\text{opt,UET-LCT}(c=\delta),\infty} \leq \frac{(\delta+1)}{2} C_{\max}^{\text{opt},G^*,\infty} \tag{4.6}$$

and so

$$C_{\max}^{G^*,\infty} \leq C_{\max}^{\text{UET-LCT}(c=\delta),\infty} \quad \text{by (4.4)} \tag{4.7}$$

$$C_{\max}^{G^*,\infty} \leq \frac{2(\delta+1)}{3} C_{\max}^{\text{opt,UET-LCT}(c=\delta),\infty} \quad \text{using (4.3)} \tag{4.8}$$

$$C_{\max}^{G^*,\infty} \leq \frac{(\delta+1)^2}{3} C_{\max}^{\text{opt},G,\infty} \quad \text{using (4.6)} \tag{4.9}$$

$$\rho^{G^*,\infty} \leq \frac{(\delta+1)^2}{3}. \tag{4.10}$$

Now we have to transform this schedule using an infinite number of processors into a schedule with a bounded number of processors. This can be done easily using the method from [12]. The new worst-case relative performance is just increased by one. Thus we have

$$\rho^{G^*,m} \leq \rho^{G^*,\infty} + 1 \leq \frac{(\delta+1)^2}{3} + 1. \tag{4.11}$$

Remark 4.4. Note that the order of the operations may be modified. Nevertheless, the ratio becomes $7/6 \times (\delta + 1)^2$. Indeed, the folding principle may be used just after the solution given by an algorithm proposed by Munier and König [10]. We then obtain a schedule on m processors. Afterwards, we apply the dilation principle. This order yields a polynomial-time approximation algorithm with a ratio bounded by $7/6 \times (\delta + 1)^2$.

Remark 4.5. we may recall two classic results in scheduling problems for which the performance ratio increases by one between the unbounded and bounded versions.

(1) When the number of processors is unlimited, the problem of scheduling a set of n tasks under precedence constraints with noncommunication delay is polynomial. It is sufficient to use the classical algorithm given by Bellman [13] as well as the two techniques widely used in project management: CPM (Critical Path Method) and PERT (Project/Program Evaluation and Review Technique). In contrast, when the number of processors is limited, the problem becomes $\mathcal{NP}$-complete and a $(2 - 1/m)$-approximation is developed by Graham, see [14], where m designates the number of processors based on a list scheduling in which no order on tasks is specified.

(2) The second illustration is given by the transition to UET-UCT on unrestricted version to the restricted variant. In [10], we know the existence of a 4/3-approximation algorithm. Using the previous result Munier and Hanen in [15] design a 7/3-approximation for the restricted version.

5. Conclusion

We have sharpened the demarcation line between the polynomially solvable and $\mathcal{NP}$-hard case of the central scheduling problem (UET-UCT) on a structured processor network by showing that its decision is polynomially solvable for $C_{\max} \leq 2$ while it is $\mathcal{NP}$-complete for $C_{\max} \geq 3$. This result is given for a large class of graph with a nonconstant diameter. This result implies there is no ρ-approximation algorithm with $\rho < 4/3$. These results are extended to the case of precedence graph is a bipartite graph.

Lastly, we complete our complexity results by developing a polynomial-time approximation algorithm for $(P, G^*)|\text{prec}, c_{ij} = d(\pi^k, \pi^l) = 1, p_i = 1|C_{\max}$ with a worst-case relative performance of $(\delta + 1)^2/3 + 1$, where δ designates the diameter of the graph. An interesting question for further research is to find a polynomial-time approximation algorithm with performance guarantee ρ with $\rho \in \mathbb{R}$.

Appendices

A.

This section describes the dilatation principle. This principle has been studied in [11], and used for designing a new polynomial-time approximation algorithm with a nontrivial performance guarantee for the problem $\overline{P}|\text{prec}; c_{ij} = c \geq 2; p_i = 1|C_{\max}$. For the latter problem, the authors propose a $c + 1/2$-approximation algorithm (the best ratio as far as we know).

A.1. Introduction, Notation, and Description of the Method

Notation 1. We use σ^∞ to denote the UET-UCT schedule, and by σ_c^∞ the UET-LCT schedule. Moreover, we use t_i (resp., t_i^c) to denote the starting time of the task i in schedule σ^∞ (resp., in schedule σ_c^∞).

Principle

The tasks in σ_c^∞ allow the same assignment as the feasible schedule σ^∞ on an unbounded number of processors. We proceed to an expansion of the makespan, while preserving the communication delay ($t_j^c \geq t_i^c + 1 + c$) for two tasks i and j, with $(i, j) \in E$, processing on two different processors. For this, the starting time t_i^c is translated by a factor d.

In the following section, we will justify and determine the coefficient d.

More formally, let $G = (V, \text{E})$ be a precedence graph. We determine a feasible schedule σ^∞, for the model UET-UCT, using the (4/3)-approximation algorithm proposed by Munier and König [10]. The result of this algorithm gives a couple of values (t_i, π), $\forall i \in V$ on the schedule σ^∞ with t_i being the starting time of the task i for the schedule σ^∞ and π the processor on which the task i will be processed at t_i.

From this solution, we will derive a solution for the problem with large communication delays. For this, we will propose a new couple of values (t_i^c, π'), $\forall i \in V$ derived from couple (t_i, π). The computation of this set of new couples is obtained in the following ways: the start time $t_i^c = d \times t_i = ((c + 1)/2)t_i$ and, $\pi = \pi'$. In other words, all tasks in the schedule σ_c^∞ are allotted on the same processor as the schedule σ^∞, and the starting time of a task i undergoes a translation with a factor $(c + 1)/2$. The justification of the expansion coefficient is given below. An illustration of the expansion is given in Figure 10.

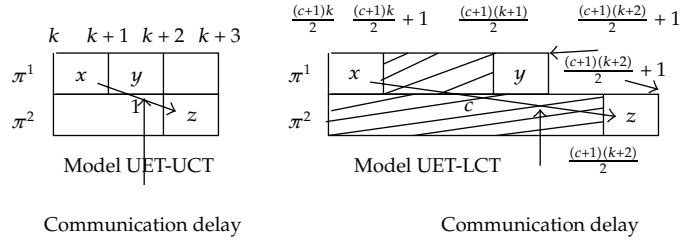


Figure 10: Illustration of the notion of expansion.

A.2. Feasibility, Analysis of the Method, and Computation of the Ratio

Afterwards, we will justify the existence of the coefficient d. Moreover, we prove the correctness of the feasible schedule for $\overline{P}|\text{prec}; c_{ij} = c \geq 2; p_i = 1|C_{\max}$ problem. Lastly, we propose a worst-case analysis for the algorithm.

Lemma A.1. *The coefficient of an expansion is* $d = (c + 1)/2$.

Proof. Let there be two tasks i and j such that $(i, j) \in E$, which are processed on two different processors in the feasible schedule σ^{∞}. We are interested in obtaining a coefficient d such that $t_i^c = d \times t_i$ and $t_j^c = d \times t_j$. After expansion, in order to respect the precedence constraints and communication delay, we must have $t_j^c \geq t_i^c + 1 + c$, and so $d \times t_i - d \times t_j \geq c + 1, d \geq (c + 1)/(t_i - t_j), d \geq c + 1/2$. It is sufficient to choose $d = (c + 1)/2$. □

Lemma A.2. *An expansion algorithm gives a feasible schedule for the* $\overline{P}|prec; c_{ij} = c \geq 2; p_i = 1|C_{\max}$ *problem in* $O(n)$.

Proof. It is sufficient to check that the solution given by an expansion algorithm produces a feasible schedule for the UET-LCT model. Let i and j be two tasks such that $(i, j) \in E$. We use π^i (resp., π^j) to denote the processor on which task i (resp., the task j) is executed in schedule σ^{∞}. Moreover, we use π'^i (resp., π'^j) to denote the processor on which task i (resp., the task j) is executed in schedule σ_c^{∞}. Thus,

(i) if $\pi^i = \pi^j$ then $\pi'^i = \pi'^j$. Since the solution given by Munier and König [10] gives a feasible schedule on the model UET-UCT, we have $t_i + 1 \leq t_j$, $(2/(c + 1))t_i^c + 1 \leq (2/(c + 1))t_j^c$; $t_i^c + 1 \leq t_i^c + (c + 1)/2 \leq t_j^c$;

(ii) if $\pi^i \neq \pi^j$ then $\pi'^i \neq \pi'^j$. We have $t_i + 1 + 1 \leq t_j, (2/(c + 1))t_i^c + 2 \leq (2/(c + 1))t_j^c$; $t_i^c + (c + 1) \leq t_j^c$.

□

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

Proof. We use $C_{\max}^{h,\text{UET-UCT},\infty}$ (resp., $C_{\max}^{\text{opt,UET-UCT},\infty}$) to denote the makespan of the schedule computed by Munier and König (resp., the optimal value of a schedule σ^{∞}). In the same way, we use $C_{\max}^{h^*,\text{UET-LCT},\infty}$ (resp., $C_{\max}^{\text{opt,UET-LCT},\infty}$) to denote the makespan of the schedule computed by an algorithm (resp., the optimal value of a schedule σ_c^{∞}).

We know that

$$C_{\max}^{h,\text{UET-UCT},\infty} \leq \frac{4}{3} C_{\max}^{\text{opt,UET-UCT},\infty}. \tag{A.1}$$

Thus, we obtain

$$\begin{aligned} \frac{C_{\max}^{h^*,\text{UET-LCT},\infty}}{C_{\max}^{\text{opt,UET}-\text{LCT},\infty}} &= \frac{((c+1)/2) C_{\max}^{h,\text{UET-UCT},\infty}}{C_{\max}^{\text{opt,UET-LCT},\infty}} \leq \frac{((c+1)/2) C_{\max}^{h,\text{UET-UCT},\infty}}{C_{\max}^{\text{opt,UET-UCT},\infty}} \\ &\leq \frac{(c+1/2)(4/3) C_{\max}^{\text{opt,UET-UCT},\infty}}{C_{\max}^{\text{opt,UET-UCT},\infty}} \leq \frac{2(c+1)}{3}. \end{aligned} \tag{A.2}$$

□

B.

In this section, we present a simple algorithm which gives a schedule σ^m on m machines from a schedule σ^∞ on an unbounded number of processors for $\overline{P}|\text{prec}, c_{ij} = 1, p_i = 1|C_{\max}$. Let X_i be the set of tasks executed at t_i in σ^∞ using a heuristic h^*. The X_i tasks are executed in $\lceil |X_i|/m \rceil$ units of time in the schedule σ^m. We apply this procedure for all $i = 0, \ldots, C_{\max}^{h^*,\text{UET-UCT},\infty} - 1$. 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 $t_i + 1$ (called Brent's lemma, see [12]).

Theorem B.1. *From any polynomial time algorithm h^* with performance guarantee ρ^∞ (i.e., $C_{\max}^{h^*,UET\text{-}UCT,\infty} \leq \rho^\infty C_{\max}^{opt,UET\text{-}UCT,\infty}$) for the problem $\overline{P}|prec, c_{ij} = 1, p_i = 1|C_{\max}$, we may obtain a polynomial-time algorithm h with performance guarantee $\rho^m = (1 + \rho^\infty)$ for the problem $P|prec, c_{ij} = 1, p_i = 1|C_{\max}$.*

Proof. Let $C_{\max}^{h^*,\text{UET-UCT},\infty}$ (resp., $C_{\max}^{h,\text{UET-UCT},m}$) be the length of the schedule given by h^* (resp., by h). In the same way, let $C_{\max}^{\text{opt,UET-UCT},\infty}$ (resp., $C_{\max}^{\text{opt,UET-UCT},m}$) be the optimal length of the schedule on an unbounded number of processors (resp., in a restricted number of processors). We denote by n the number of tasks in the schedule.
Clearly, this gives us $C_{\max}^{\text{opt,UET-UCT},\infty} \leq C_{\max}^{\text{opt,UET-UCT},m}$ and $C_{\max}^{h^*,\text{UET-UCT},\infty} \leq \rho C_{\max}^{h,\text{UET-UCT},\infty}$. So,

$$\begin{aligned} C_{\max}^{h,\text{UET-UCT},m} &\leq \sum_{i=0}^{C_{\max}^{h^*,\text{UET-UCT},\infty}-1} \left\lceil \frac{|X_i|}{m} \right\rceil \leq \sum_{i=0}^{C_{\max}^{h^*,\text{UET-UCT},\infty}-1} \left(\left\lfloor \frac{|X_i|}{m} \right\rfloor + 1 \right), \\ C_{\max}^{h,\text{UET-UCT},m} &\leq \sum_{i=0}^{C_{\max}^{h^*,\text{UET-UCT},\infty}-1} \left(\frac{|X_i|}{m} \right) + C_{\max}^{h^*,\text{UET-UCT},\infty}, \\ C_{\max}^{h,\text{UET-UCT},m} &\leq C_{\max}^{\text{opt,UET-UCT},m} + C_{\max}^{h^*,\text{UET-UCT},\infty}, \\ C_{\max}^{h,\text{UET-UCT},m} &\leq C_{\max}^{\text{opt,UET-UCT},m} + \rho C_{\max}^{\text{opt,UET-UCT},m}, \\ \rho^m &\leq (1 + \rho^\infty). \end{aligned} \tag{B.1}$$

This concludes proof of Theorem B.1. □

References

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[2] C. Picouleau, "UET-UCT schedules on arbitrary networks," Tech. Rep., LITP, Blaise Pascal, Université Paris VI, 1994.

[3] R. Giroudeau, J. C. König, and B. Valéry, "Scheduling uet-tasks on a star network: complexity and approximation," *4OR A Quarterly Journal of Operations Research*, vol. 9, no. 1, pp. 29–48, 2011.

[4] V. Boudet, Y. Cohen, R. Giroudeau, and J. C. Konig, "Complexity results for scheduling problem with non trivial topology of processors," Tech. Rep. 06050, LIRMM, 2006, submitted to Rairo-RO.

[5] C. Lahlou, "Scheduling with unit processing and communication times on a ring network: approximation results," in *Proceedings of Europar*, pp. 539–542, Springer, New York, NY, USA, 1996.

[6] V. J. Rayward-Smith, "UET scheduling with unit interprocessor communication delays," *Discrete Applied Mathematics*, vol. 18, no. 1, pp. 55–71, 1987.

[7] J. J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, "Scheduling precedence graphs in systems with interprocessor communication times," *SIAM Journal on Computing*, vol. 18, no. 2, pp. 244–257, 1989.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, A Series of Books in the Mathematical Science, W. H. Freeman, San Francisco, Calif, USA, 1979.

[9] P. Chrétienne and C. Picouleau, *Scheduling Theory and Its Applications*, Scheduling with Communication Delays: A Survey, chapter 4, John Wiley & Sons, Chichester, UK, 1995.

[10] A. Munier and J. C. König, "A heuristic for a scheduling problem with communication delays," *Operations Research*, vol. 45, no. 1, pp. 145–148, 1997.

[11] R. Giroudeau, J.-C. Konig, F. K. Moulai, and J. Palaysi, "Complexity and approximation for precedence constrained scheduling problems with large communication delays," *Theoretical Computer Science*, vol. 401, no. 1–3, pp. 107–119, 2008.

[12] R. P. Brent, "The parallel evaluation of general arithmetic expressions," *Journal of the Association for Computing Machinery*, vol. 21, pp. 201–206, 1974.

[13] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[14] R. Graham, "Bounds for certain multiprocessing anomalies," *Bell System Technical Journal*, vol. 45, pp. 1563–1581, 1966.

[15] A. Munier and C. Hanen, "An approximation algorithm for scheduling unitary tasks on m processors with communication delays," private communication, 1996.