Complexity of Task Graph Scheduling with Fixed Communication Capacity

Lucian Finta, Zhen Liu

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Complexity of Task Graph Scheduling with Fixed Communication Capacity

Lucian FINTA Zhen LIU

N° 2959

August 1996

THÈME 1

ISSN 0249-6399

Complexity of Task Graph Scheduling with Fixed Communication Capacity

Lucian FINTA Zhen LIU

Thème 1 — Réseaux et systèmes

Projet MISTRAL

Rapport de recherche n ° 2959 — August 1996 — 46 pages

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Abstract: Consider a scheduling problem of parallel computations in multiprocessor systems. Let a parallel program be modeled by a task graph, where vertices represent tasks and arcs the communications between tasks. An interprocessor communication time incurs when two tasks assigned to two different processors have to communicate. Such a scheduling problem has recently been studied in the literature, mostly for the case where interprocessor communication times are fully determined. In this paper, we consider the scheduling problem with communication resource constraints. More specifically, we consider the case where all interprocessor communications take place on a network (communication medium) of bounded capacity. We consider two variants of the problem: communications with independent-data semantics or common-data semantics. We show that even for very specific subproblems, viz. scheduling of general graphs on two processors and scheduling of binary trees on infinite number of processors, the minimization of the makespan of parallel programs in such a multiprocessor system is strongly $\mathcal{NP}$-hard. We first establish the results for the case of capacity 1, referred to as the single-bus system. We then extend the results to the more general case of fixed communication capacities. As a consequence, the general scheduling problem of parallel programs with communication resource constraints is strongly $\mathcal{NP}$-hard. These results are to be contrasted with the corresponding scheduling problems without constraint on the communication capacity, where the two-processor case has unknown time complexity and the infinite-processor case is polynomial. Our results are also extended to the case of broadcasting communications, and can be applied to multiprocessor systems with shared memory.

Key-words: Scheduling, Makespan, Precedence Constraint, Bounded Communication Capacity, Resource Constraint, $\mathcal{NP}$-completeness.

(Résumé : tsvp)

Correspondence: Zhen LIU, INRIA, Centre Sophia Antipolis, 2004 route des Lucioles, B.P. 93, 06902 Sophia-Antipolis, France. e-mail: liu@sophia.inria.fr

Complexité des problèmes d'ordonnancement des programmes parallèles avec capacité de communication fixe

Résumé : Nous considérons un problème d'ordonnancement pour les calculs parallèles dans un système multiprocesseur. Un programme parallèle est représenté par un graphe de tâches, où les sommets représentent les tâches et les arcs les communications entre les tâches. Un tel problème a été récemment étudié dans la littérature dans le cas où les temps de communication ne dépendent pas de l'état du système. Dans cet article, nous considérons le problème d'ordonnancement avec contrainte de ressources. Plus précisement, nous étudions le cas où toutes les communications ont lieu sur un réseau de communication avec capacité bornée. Nous analysons deux variantes du problème : les communications avec sémantique de données indépendantes ou sémantique de données communes. Nous prouvons que, même pour des sous-problèmes spécifiques, c'est-à-dire, l'ordonnancement des graphes généraux sur deux processeurs et l'ordonnancement des arborescences binaires sur une infinité de processeurs, la minimisation de la durée d'ordonnancement pour les programmes parallèles dans un tel système multiprocesseur est $\mathcal{NP}$-difficile. Nous établissons ces résultats d'abord pour le cas de capacité 1 (le cas du système avec bus unique). Puis nous les généralisons au cas général de capacité fixe. Par conséquent, le problème général d'ordonnancement pour les programmes parallèles avec contrainte de ressources de communication est $\mathcal{NP}$-difficile. Nos résultats s'étendent aux cas des communications avec diffusion, et peuvent s'appliquer aux systèmes multiprocesseurs à mémoire partagée.

Mots-clé : ordonnancement, durée d'ordonnancement, contraintes de précédence, capacité de communication bornée, complexité, $\mathcal{NP}$-complétude.

1 Introduction

Scheduling of parallel programs in multiprocessor systems is one of the important issues in parallel computing. Parallel programs are usually represented by task graphs, where vertices represent tasks and arcs the communications between the tasks. An interprocessor communication time incurs when two tasks assigned to two different processors have to communicate. A task can start execution only when all its predecessor tasks have completed execution and the communications between its predecessor tasks and the task have taken place. Our goal is to minimize the makespan, or the schedule length, of a program, i.e. the maximum task completion time.

Such a scheduling problem has been receiving growing interest in the literature recently. Most of the studies have been focused on the case where interprocessor communication times are fully determined, i.e., interprocessor communication times are constants which are possibly dependent of the message length and distance between processors, but are otherwise independent of the status of system (in particular, the congestion of the communication network).

Rayward-Smith [26] first analyzed the case where tasks have unit-execution-time (UET) and interprocessor communications have unit-communication-time (UCT). He showed that the minimization of makespan is strongly $\mathcal{NP}$-hard. Picouleau [24] and Hoogeveen et al. [14] analyzed the complexity (polynomial or $\mathcal{NP}$-complete) of the decision problem whether the minimum makespan of a program can be smaller than a threshold. Various heuristics were proposed in [13, 17, 21, 28], and a general worst-case analysis of greedy heuristics was given in Liu [20]. For the case of two processors, when the task graph has an in-tree structure with UET-UCT, polynomial optimal algorithms were recently proposed in Varvarigou et al. [29], Lawler [16], Lenstra et al. [19], Guinand and Trystram [12]. These polynomial-time-complexity results are further extended to the UET-UCT series-parallel graphs by Finta et al. [8] who proposed a polynomial optimal algorithm for the scheduling on two processors.

There are also studies on the case of infinite number of processors. Chretienne [3] provided a polynomial algorithm for the case of trees with integer execution time and

short communication times (shorter than execution time). Picouleau [24] obtained a polynomial algorithm for the case of series-parallel graphs. Picouleau [25] and Hoogeveen et al. [14] showed that the problem is strongly $\mathcal{NP}$-hard for general task graphs with UET-UCT. There are also studies for the case when task duplication is allowed, see e.g. Papadimitriou and Yannakakis [22] and Colin and Chretienne [6].

Another related problem is the minimization of the total amount of communication, which was analyzed in Prastein [23] and Afrati et al. [1].

The reader is referred to Veltman et al. [31] for a classification of the variants of the scheduling problem, and to Chretienne and Picouleau [4] for a survey on the complexity results, optimal and approximate algorithms.

In this paper, we consider the scheduling problem with communication resource constraints. More specifically, we consider the case where all interprocessor communications take place on a network (communication medium) of bounded capacity (bandwidth). We consider two variants of the problem: communications with independent-data semantics or common-data semantics. We show that even for very specific subproblems, viz. scheduling of general graphs on two processors and scheduling of binary trees on infinite number of processors, the minimization of the makespan of parallel programs in such a multiprocessor system is strongly $\mathcal{NP}$-hard. We first establish the results for the case of capacity 1, referred to as the single-bus system. We then extend the results to the more general case of fixed communication capacities. As a consequence, the general scheduling problem of parallel programs with communication resource constraints is strongly $\mathcal{NP}$-hard. These results are to be contrasted with the corresponding scheduling problems without constraint on the communication capacity, where the two-processor case has unknown time complexity and the infinite-processor case is polynomial. Our results are also extended to the case of broadcasting communications, and can be applied to multiprocessor systems with shared memory.

We consider several variants of the problem. Communications between tasks can have independent-data or common-data semantics. The distinction is due to the type

of data dependency we consider for two tasks in direct precedence relation. Such a distinction of data semantics was also made in [1, 23]. A more general discussion can be found in [31].

The paper is organized as follows. In Section 2, we describe the scheduling problem and its variants in detail. We will first consider single-bus systems, where the communication capacity is bounded by one. In Section 3 we prove the strong $\mathcal{NP}$-hardness of problems when communications between tasks have independent-data semantics. In Section 4 we prove the strong $\mathcal{NP}$-hardness of problems when communications between tasks have common-data semantics. In Section 5, we extend these results to the more general case where the communication capacity is a fixed integer. Finally, in Section 6, we conclude with remarks on further complexity results for the cases with task preallocation, trees on two processors and blocking communication mechanism.

2 Problem Description

The multiprocessor system under consideration contains m parallel processors connected to communication network. All communications between processors take place on the network. The capacity, or bandwidth, of the network is denoted by $B \geq 1$. At any time, at most B processors can send a message over the network. A particular example of the network is the single-bus system, where the capacity $B = 1$.

Two *communication mechanisms* will be considered. The first one is *one-to-one communication*, where only one of the processors can be the receiver of a message. The second one is *broadcasting communication*, where several processors can be receivers of a message, provided they are the destinations.

When a processor is sending or receiving a message, it can stop processing a task until the communication is finished. This communication mechanism is called communication with *blocking*. In what follows we shall consider nonblocking commu-

nications. For the case of blocking communications, the interested reader is referred to [7], where less strong results were obtained.

A parallel program is represented by a directed acyclic graph $G = (V, E)$, referred to as *task graph*, where vertices in V represent tasks of the parallel program, arcs of E represent data dependence or communication relations between tasks. If $(i, j) \in E$, then task j has to wait for results of task i before starting its own execution.

Task running times on processors are assumed to be known constants. Throughout this paper we consider tasks with UET.

The time for data transfer between tasks allocated to the same processor is assumed to be negligible. However, data transfer between two tasks allocated to different processors are carried out through message passing. The set of data to be transferred from one task to another can be specified as weight of an arc in the task graph. In general, as discussed in Veltman et al. [31], if two immediate successors u, v of task i are assigned to the same processor and if task u starts execution prior to task v, then only those data that are useful to v but not to u need to be transferred from task i to task v. In order to simplify discussions, we shall consider two (extreme) cases of the *communication semantics*: *independent data* and *common data*. In the case of independent-data communication semantics, the sets of data to be transferred from a task to its immediate successors are assumed to be different. In the case of common-data communication semantics, however, the sets of data to be transferred from a task to its immediate successors are assumed to be the same. Only in this case one can use broadcasting communication mechanisms.

In both cases, we can simply specify the communications between tasks by the amount of data to be transferred, or even simpler, the communication time for the data transfer if the communication takes place in the network. In this paper, we assume that these communication times are specified (e.g. as the weights of arcs in the task graph). We will consider the simplest case, i.e. UCT, where all communications in the network take unit time.

Note that in the literature, most of the results (see e.g. [4]) on scheduling of parallel programs with communication are on the independent-data communication semantics.

A *schedule* of the parallel program in the multiprocessor system defines for each task the processor the task is assigned to, and the time epoch when the task starts execution. The schedule defines also for each communication the time instant when the message is sent over the network. The schedule is feasible if at any time only one task is running on a processor and at most B communications occur in the network, and if a task starts its execution on a processor only after all data needed from its predecessors are ready on the processor. No *task duplication* is allowed, i.e., a task can be run only on one processor.

Given a schedule σ, let $C_i(\sigma)$ denote the completion time of the task i in the schedule σ. The goal of our study is to minimize the schedule length or the *makespan*, i.e. the maximum of task completion times: $C_{\max}(\sigma) = \max_{i \in V} C_i(\sigma)$.

This scheduling problem can be defined in a formal way as follows. Let there be m identical processors. Let $G = (V, E)$ be the task graph with p_i, $i \in V$, being the processing time of task i, and $c(i,j)$, $(i,j) \in E$, being the communication time if tasks i and j are assigned to different processors. Each task $i \in V$ can be run on one of the processors of the subset $A_i \subseteq \{1, 2, \cdots, m\}$. A schedule is a pair of functions (π, σ), where $\pi(i)$ specifies the processor to which task $i \in V$ is assigned, $\sigma(i)$ is the starting time of task $i \in V$, and $\sigma(i,j)$ is the starting time of communication $(i,j) \in E$ in the network, provided $\pi(i) \neq \pi(j)$.

The assignment is constrained by $\pi(i) \in A_i$, $i \in V$. In this paper we shall consider the case $A_i = \{1, 2, \ldots, m\}$ for all $i \in V$, which we referred to as *tasks without preallocation*. The case of *tasks with preallocation*, which corresponds to the case where A_i is singleton for all $i \in V$, will be discussed briefly in Section 6.

The time schedule should satisfy the precedence constraints:

$$\sigma(i,j) \geq \sigma(i) + p_i, \qquad (i,j) \in E, \quad \pi(i) \neq \pi(j), \tag{1}$$

and

$$\sigma(j) \geq (\sigma(i) + p_i)\,\mathbf{1}_{\pi(i)=\pi(j)} + (\sigma(i,j) + c(i,j))\,\mathbf{1}_{\pi(i)\neq\pi(j)}, \qquad (i,j) \in E, \quad (2)$$

where $\mathbf{1}_\bullet$ is the indicator function. Moreover, at any time, at most one task is running on a processor:

$$\forall i,j \in V: \quad \text{if } \pi(i) = \pi(j) \quad \text{then} \quad \sigma(j) \geq \sigma(i) + p_i \quad \text{or} \quad \sigma(i) \geq \sigma(j) + p_j, \quad (3)$$

and at most B communications are taking place in the network: $\forall t > 0$,

$$\sum_{(i_1,i_2)\in E,\ \pi(i_1)\neq\pi(i_2)} \mathbf{1}_{t\in[\sigma(i_1,i_2),\ \sigma(i_1,i_2)+c(i_1,i_2))} \leq B. \quad (4)$$

When common-data semantics is under consideration, we have the constraints that $c(i,j) = c(i)$ for all $(i,j) \in E$ and that

$$\sigma(i,j) = \sigma(i,j')\mathbf{1}_{\pi(i)\neq\pi(j),\ \pi(j)=\pi(j')} \qquad (i,j),(i,j') \in E. \quad (5)$$

If, moreover, the broadcasting communication mechanism is implemented,

$$\sigma(i,j) = \sigma(i,j')\mathbf{1}_{\pi(i)\neq\pi(j),\ \pi(i)\neq\pi(j')} \qquad (i,j),(i,j') \in E. \quad (6)$$

In this paper we consider the special case: UET-UCT, i.e. $p_i = 1$ for $i \in V$ and $c(i,j) = 1$ for $(i,j) \in E$. We obtain "negative" results of the scheduling problem, i.e., we prove the strong $\mathcal{NP}$-hardness of the makespan minimization problems. In order to do this, we consider the associated *decision problem*:

> *Given a task graph $G = (V,E)$ with UET-UCT and independent-data (resp. common-data) communication semantics, m processors with a communication network of capacity B, and a time limit T, is there a feasible schedule (π,σ) satisfying constraints (1)–(4) (resp. (1)–(5)) such that $C_{\max}(\sigma) \leq T$?*

When broadcasting communication is allowed, constraint (5) is replaced by (6) in the above definition.

According to the three-field notation scheme introduced by Graham, Lawler, Lenstra and Rinnooy Kan [11] and the extension by Veltman, Lageweg and Lenstra [31] for scheduling problems with communication delays, our problem can be denoted as $P,\ B \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$, where $c = 1\ (i.d.)$ denotes that any communication delay is unit with independent-data semantics. When there is *unbounded number* of processors, say $m \geq |V|$, the problem is denoted as $\overline{P},\ B \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$. When there is *fixed number* m of processors, the problem is denoted as $Pm,\ B \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$. When the communication network has *fixed capacity* b, the problem is denoted as $P,\ Bb \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$.

If common-data semantics is under consideration, the problems will be denoted as $P,\ B \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ for arbitrary number of processors, $\overline{P},\ B \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ for unbounded number of processors, $Pm,\ B \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ for fixed number of processors. and $P,\ Bb \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ for fixed communication capacity.

We will show that these decision problems are strongly $\mathcal{NP}$-complete so that the corresponding optimization problems are strongly $\mathcal{NP}$-hard. Since the source problems of the polynomial transform in our proofs are concerned with non-number problems, the strong $\mathcal{NP}$-completeness follows from the $\mathcal{NP}$-completeness. Thus in the sequel we shall not stress this and omit the term "strongly" in the statements.

3 Scheduling with Independent-Data Communication Semantics

In this section we consider scheduling problems with independent-data communication semantics. We shall analyze the single-bus system ($B = 1$). The more general case will be considered in Section 5. We prove that the problem is $\mathcal{NP}$-hard for bi-

nary trees (or, more precisely, binary forests) with unbounded number of processors, and for general task graphs with two processors.

We polynomially transform the well-known *3SAT* problem to $\overline{P},\ B1 \mid \mathit{binary\ tree},\ c = 1\ (\mathit{i.d.}),\ p_j = 1 \mid C_{\max}$. We recall the definition of 3SAT.

> *3SAT problem*: Given a set of variables U, a collection of clauses C over U such that each clause has three literals, is there a satisfying truth assignment for C?

Lemma 3.1 *3SAT problem polynomially transforms to* $\overline{P},\ B1 \mid \mathit{binary\ tree},\ c = 1\ (\mathit{i.d.}),\ p_j = 1 \mid C_{\max}$.

Proof. Given an instance of the above 3SAT problem, we construct the following instance of the scheduling problem, such that there exists a feasible schedule σ if and only if the 3SAT problem has a solution. The construction is inspired by that of the proof of Theorem 1 of Lenstra et al. [19] who obtained the strong $\mathcal{NP}$-hardness of $P \mid \mathit{tree},\ c = 1,\ p_j = 1 \mid C_{\max}$.

Let $n = |U|$ be the number of variables and $m = |C|$ the number of clauses in the 3SAT problem. The time limit for the scheduling function is $T = 3m + n + 6$. The task graph $G = (V, E)$ is constructed as follows:

- For each variable u_i, $1 \leq i \leq n$, we introduce a task x_{u_i}, a V-chain $\hat{u}_i$ of $i + 1$ tasks and two L-chains of $n - i + 3m + 4$ tasks each; one of these two chains corresponds to the literal u_i and the other to the literal $\bar{u}_i$. The x-task is preceded by the first task of the V-chain. Each L-chain is preceded by the V-chain, i.e., there is an arc from the last task of the V-chain $\hat{u}_i$ to the first task of L-chain u_i (resp. $\bar{u}_i$).
- For each clause c_j, $1 \leq j \leq m$, we introduce three C-chains of $3j + 1$ tasks each. There is an one-to-one correspondence between those chains and the

literals that constitute c_j. If variable u_i occurs in clause c_j, the corresponding C-chain is denoted by c_{j,u_i}. The first task of the C-chain is preceded by the $(n-i+3(m-j)+1)$-st task of L-chain u_i (resp. $\bar{u}_i$) provided that literal u_i (resp. $\bar{u}_i$) occurs in clause c_j.

- Finally, we introduce a T-chain of T tasks. Attached to this chain, there are four chains of $3m+n+4$, 3, 2 and 1 task, respectively, which are successors of the 1-st, $(T-4)$-th, $(T-3)$-rd and $(T-2)$-nd task of the T-chain.

We thus obtain a binary forest with $n+1$ binary trees. The trees other than that connected with the T-chain are referred to as V-trees.

In any feasible schedule, it is clear the T-chain is to be processed on one processor and the communications from it to the other four chains are critical, and must take place on the bus in time slots 2, $T-3$, $T-2$ and $T-1$.

Figure 1 illustrates the task graph that we construct for an instance of 3SAT problem. The clauses are: $C_1 = u+\bar{v}+w$ and $C_2 = u+v+\bar{w}$. The time limit is 15.

Given a satisfying truth assignment for 3SAT, we can construct a feasible schedule of length T as follows. Each V-chain is executed without interruption on the same processor. Different V-chains are executed on different processors. If variable u_i is true (resp. false), the L-chain u_i (resp. L-chain $\bar{u}_i$) is executed immediately after the V-chain on the same processor without interruption. The L-chain $\bar{u}_i$ (resp. L-chain u_i) is executed on another processor after one communication. The x-tasks are executed in the last time slot on the same processors as their predecessors.

Each C-chain is executed on a distinct processor, i.e. the arc connecting a C-chain to an L-chain corresponds to an interprocessor communication. There is a group of three communications for each clause. Each such group uses the bus for three units of time. The first communication of such a group is feasible since at least one literal is true in each clause so that the corresponding L-chain is executed immediately after the V-chain on the same processor. The other two communications are feasible as well since they are on paths of length $T-3$.

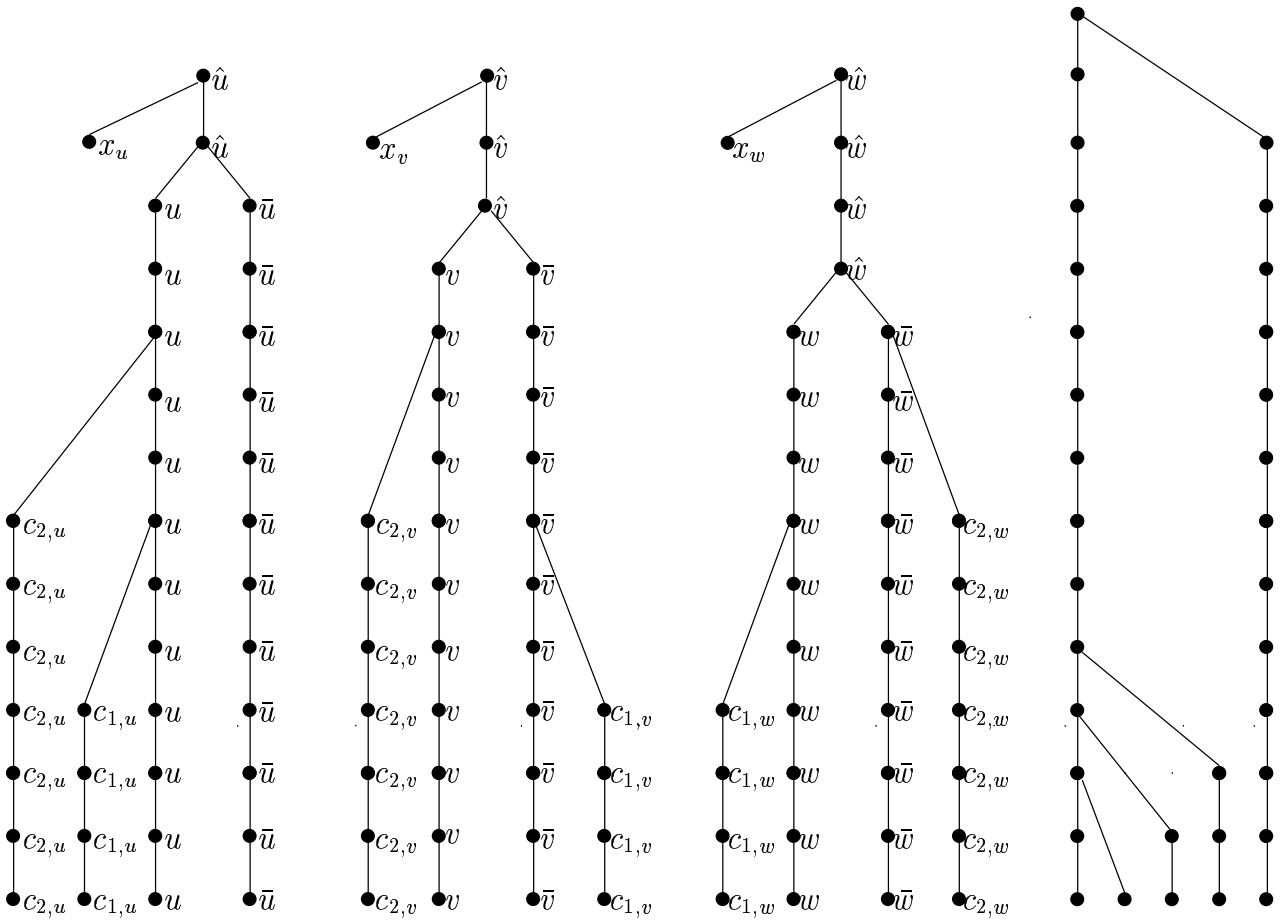


Figure 1: Task graph instance of $\overline{P},\ B1 \mid \textit{binary tree},\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$ corresponding to the instance of 3SAT problem.

We now show that for any feasible schedule there is a solution to the corresponding instance of 3SAT.

Consider a V-tree corresponding to some variable u. There are two paths of $T-1$ tasks from the root of the tree ending with L-chains. These paths will be denoted as T_1-paths. Suppose that the number of C-chains in the tree is k. Then there are k paths of length $T-3$ in the tree. Such paths are denoted as T_3-paths.

The following claims present some useful properties of T_1-paths and T_3-paths of a V-tree. For simplicity, in what follows, interprocessor communication is referred to as communication.

Claim 1: *There is at least one communication in the two T_1-paths of the same V-tree.* Since the total number of tasks of the two T_1-paths is greater than T, one cannot execute all those tasks on only one processor.

Claim 2: *There is at most one communication in each T_1-path.* Since each T_1-path has $T-1$ tasks, two or more communications would make the schedule infeasible.

Claim 3: *There is no communication between tasks of V-chains.* Otherwise, all tasks of the two T_1-paths of the V-tree after the communication have to be executed on only one processor. This is impossible since only one of the T_1-paths can be executed entirely on a processor.

Claim 4: *Neither a T_1-path together with a T_3-path, nor two T_3-paths can be executed on the same processor.* This is because such paths have at least $T-3$ tasks and any C-chain has at least 4 tasks.

Claim 5: *There are at least $k+1$ communications for each V-tree. If there are 2 communications in the two T_1-paths of a V-tree, the first such communication corresponds to the arc between the V-chain and an L-chain.*

In order to prove the above claim, we first assume that there are 2 communications in the two T_1-paths of a V-tree, i.e. one L-chain u and the other on L-chain $\bar{u}$. Thus, there is one communication in each T_1-path. Both T_1-paths are now critical. Moreover, it is easy to see that the processor executing the root of the tree has to execute tasks of only one T_1-path until the second communication. Therefore, the first communication must be between the V-chain and an L-chain, say L-chain u, otherwise the second communication would occur too late.

This L-chain u is critical in the sense that all its tasks have to be executed on the same processor continuously. Thus all C-chains connected to L-chain u correspond to communications as they have to be executed on other processors. The other L-chain,

i.e. L-chain $\bar{u}$, starts execution immediately after the V-chain on the same processor. Let y be the task that initiates the communication on the L-chain $\bar{u}$. Then, all C-chains connected to L-chain $\bar{u}$ through the predecessors of y or y itself correspond to communications as they have to be executed on other processors, cf. Claim 4 (except at most one, if any). Moreover, all C-chains connected to L-chain $\bar{u}$ through the successors of y have to be executed on other processors (as the L-chain becomes critical). Thus, at least $k-1$ C-chains have to be executed on different processors than their predecessors. We conclude that there are at least $k+1$ communications for each tree, i.e. $k-1$ for C-chains and two communications for the T_1-paths.

Assume now that one of the T_1-paths contains no communication. On the processor executing the T_1-path without communication, no other T_3-path (with a C-chain) can be executed (cf. Claim 4). Therefore, by similar arguments as in the previous case, we obtain that there are $k+1$ communications for the V-tree, i.e. k for the C-chains and one for one of the T_1-paths.

Claim 6: *In any feasible schedule there are exactly $T-2$ communications.* It follows from Claim 5 that for the n V-trees there must be at least $3m+n$ communications. If we add now the four critical communications of the T-chain, and since there are only $T-2$ slots on the bus (any communication must be between two tasks), we can conclude that in any feasible schedule there must be exactly $T-2$ communications on the bus (recall that $T = 3m+n+6$).

Claim 7: *There is no communication on arcs connecting V-chains to x-tasks.* It follows directly from the fact that $T-2$ communications are needed for the C-chains, T_1-paths and the T-chain.

Claim 8: *If there is no communication on one of the T_1-paths of the same V-tree, the communication on the other T_1-path is between the V-chain and an L-chain.* This is because the processor executing the root has to execute one T_1-path and the x-task.

Claim 9: *If there are two communications on the two T_1-paths of the same V-tree, there must be one T_3-path in the tree without any communication on it.* Otherwise,

some other tree has no enough time slots for communications in order to finish execution within T slots.

Claim 10: *Any schedule having two communications on the T_1-paths of the same V-tree can be transformed to a schedule with only one communication in the two T_1-paths without increasing the makespan.* Suppose that we have a feasible schedule that makes two communications for some pair of T_1-paths belonging to the same V-tree. Let t be the time slot where the second communication occurs and y, z be the sender and receiver tasks of the communication. Let P_y be the processor executing tasks of the T_1-path between the root and y, and P_z be the processor executing tasks between z and the final task of the same path. According to Claims 7 and 9, task x connected to the root and a C-chain are executed on P_y after the execution of y. We construct a new schedule by executing the tasks of the C-chain on P_z, and executing on P_y the last $T - t$ tasks of the T_1-path (previously on P_z) and task x in the last slot. The communication between the T_1-path and the C-chain occur in the same time slot t. This schedule is feasible since the C-chain has at most $T - t$ tasks. Iterating this transformation will yield the desired schedule.

We are now in a position to show how to find a solution to the 3SAT instance from a feasible schedule. Without loss of generality we consider only schedules that always put tasks on a new (previously unused) processor after each communication. Moreover, we consider feasible schedules satisfying the property of Claim 10. Observe that in such a schedule, each arc connecting a C-chain to an L-chain represents a communication.

In our construction of the task graph, each clause is associated with three equal length C-chains and, in any feasible schedule, associated with a group of three communications. All these three communications are on T_3-paths so that they should take place in three consecutive time slots. Consider the path (among the three T_3-paths) which uses the first slot (among the three consecutive time slots) for a communication. Since this communication occurs in the first time slot, it is the only communication on that T_3-path, so that there is no communication on the T_1-path to

which the C-chain is connected. Therefore, from any feasible schedule, for any V-tree associated with variable u, if L-chain u (resp. $\bar{u}$) is assigned to the same processor as V-chain $\hat{u}$, then the truth assignment to variable u is "true" (resp. "false"). Such an assignment yields a solution to the corresponding 3SAT problem.

To conclude, a feasible schedule exists if and only if there is a satisfying truth assignment for 3SAT. ∎

As a consequence of Lemma 3.1, we have:

Theorem 3.1 *Problem* $\overline{P},\ B1 \mid$ *binary tree,* $c = 1$ *(i.d.),* $p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

Similar to the $\mathcal{NP}$-completeness of $P \mid tree,\ c = 1,\ p_j = 1 \mid C_{\max}$ by Lenstra et al. [19], we obtain, as a consequence of the above theorem,

Corollary 3.1 *Problem* $P,\ B1 \mid$ *binary tree,* $c = 1$ *(i.d.),* $p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

It is interesting to note that when there is no bus constraint, the UET-UCT scheduling problem on infinite number of processors is $\mathcal{NP}$-hard for general task graphs (cf. [14, 25]). However, when the task graph is a tree, a series-parallel graph or a bipartite graph, the problem becomes polynomial (cf. [3, 24]). The result of Theorem 3.1 indicates that the bus constraint makes the scheduling problem more difficult.

We now consider problem $P2,\ B1 \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$. We show that the makespan minimization of a task graph with UET-UCT and independent-data communication semantics is $\mathcal{NP}$-hard in the strong sense even if there are only two processors. To this end, we polynomially transform the well-known *ONE-IN-THREE 3SAT* problem (which is $\mathcal{NP}$-complete [10]) to $P2,\ B1 \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$. Recall the definition of ONE-IN-THREE 3SAT problem.

ONE-IN-THREE 3SAT problem: Given a set of variables U, a collection of clauses C over U such that each clause has three literals, is there a truth assignment for U such that each clause in C has exactly one true literal?

Lemma 3.2 *ONE-IN-THREE 3SAT problem polynomially transforms to* $P2,\ B1 \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$.

Before presenting this transformation, we prove two technical lemmas.

Lemma 3.3 *Consider task graph* $\mathcal{V}$ *in Figure 2. In any feasible schedule of length 8 of the graph, tasks* a *and* h *are executed on the same processor (say P1) in time slots* 1 *and* 6. *Tasks* s_0 *and* t_1 *are executed on the other processor (say P2) in the slots* 1 *and* 8. *Either tasks* b, c *or* d, e *are executed on P2 in the slots* $3, 4$. *Moreover, the bus is occupied in slots* $2, 3, \ldots, 7$.

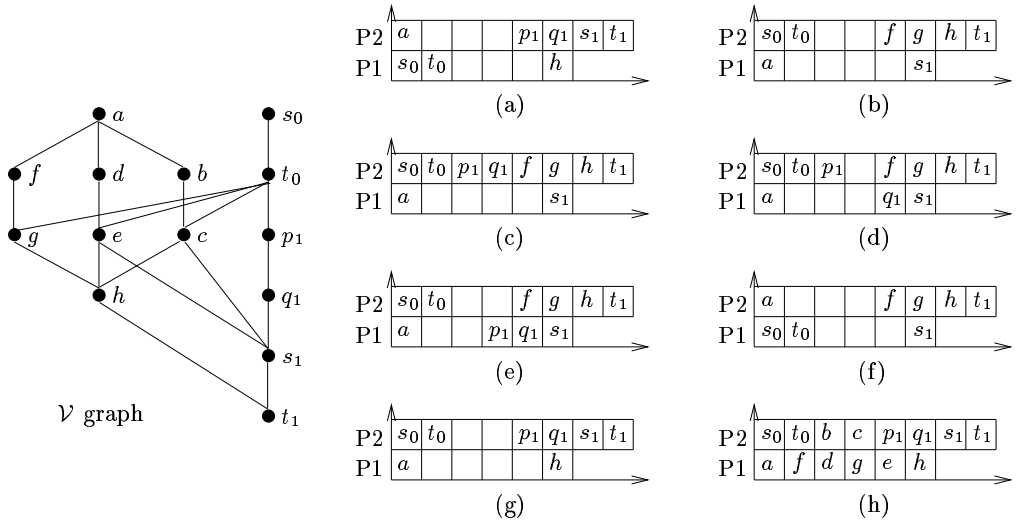


Figure 2: Task graph $\mathcal{V}$ and partial schedules for it.

Proof. Consider the case where t_1 is executed by P2 (the other case is symmetric). In any feasible schedule of length 8, processor P2 should never idle and P1 idles only in time slots 7 and 8, as t_1 has 13 predecessors.

If we now look from the end of the schedule and we schedule backwards, clearly in time slot 7 either task h or task s_1 should be executed on P2. Thus, on P1 in slot 6, either task s_1 or h should be processed. Suppose h (s_1) on P1 in time slot 6, no predecessors of h (s_1) can be executed on P2 in time slots 5, 6, tasks p_1, q_1 (f, g) become the only choice, see Figure 2-(a),(b).

In the first time slot only tasks a and s_0 are available for execution, they have to be executed on two different processors in slot 1 in order to avoid idling. Moreover, t_0 should be executed on the same processor as s_0, in order to avoid idle in slot 2 on this processor. We analyze four possible cases, of which only one case results in feasible schedules.

Case 1: tasks s_0, t_0, h are on P1 and a, s_1 on P2 (Figure 2-(a)).

Tasks b, c, d, e, f, g are to be filled in the remaining time slots, i.e. slots 2, 3, 4 on P2 and slots 3, 4, 5 on P1. It is easy to see that exactly one task among b, d, f is on P1 (in slot 3), while the other two are to be processed on P2, otherwise either P1 or P2 will idle in some time slot. Hence, tasks c, e, g are to be executed in slots 4, 5 on P1 and in slot 4 on P2 (since the task executed in slot 4 on P2 must be an *immediate* predecessor of h).

Since there are three disjoint paths from a to h, and tasks a and h are executed on different processors, there are at least three communications in time slots 2, 3, 4, 5. Similarly, since there are three disjoint paths between tasks t_0 and s_1, and that these three paths do not have common arcs with the previous three paths, there are at least three more communications to carry out on the bus during time slots 3, 4, 5, 6. Therefore, Case 1 is infeasible due to bus constraint.

Case 2: tasks s_0, t_0, h are on P2 and a, s_1 on P1 (Figure 2-(b)).

This case is also infeasible as will be shown below in the analysis of three subcases.

Case 2.a : tasks p_1, q_1 are on P2 in slots $3, 4$ (Figure 2-(c)).

Tasks b, c, d, e are to be filled in the remaining time slots, i.e. slots $2, 3, 4, 5$ on P1. Given that c, e are successors of t_0, they are to be processed in slots $4, 5$. Case 2.a is thus infeasible since there are three communications to be dealt with by the bus in slots $5, 6$, i.e. $q_1 \rightarrow s_1$, $c \rightarrow h$ and $e \rightarrow h$.

Case 2.b : task p_1 is on P2 in slot 3 and q_1 on P1 in slot 5 (Figure 2-(d)).

Since on P2 in slot 4 there can be neither b nor d, there should be either c or e, otherwise an idle occurs on P1 or P2. Suppose this task is c (the other case is symmetric). Then, on P1 we have tasks b, d, e in slots $2, 3, 4$. Thus, this subcase is again infeasible since communications $t_0 \rightarrow e$ and $b \rightarrow c$ occur concurrently in slot 3.

Case 2.c : tasks p_1, q_1 are on P1 in slots $4, 5$ (Figure 2-(e)).

It is easy to see that an immediate successor of a, i.e. one task among b, d, is to be processed on P2 in slot 3. Let this task be b. Thus, tasks d, e are processed on P1 in slots $2, 3$. This yields an infeasible schedule since both tasks c, e are successors of t_0.

Case 3: tasks s_0, t_0, s_1 are on P1 and a, h on P2 (Figure 2-(f)).

Tasks p_1 and q_1 are to be executed on P1 (since they are inbetween t_0 and s_1). Only one task among tasks b, c, d, e, f, g is executed on P1 in one time slot among slots $3, 4, 5$. Focus now on the subgraph defined by tasks t_0, p_1, q_1, c, e, s_1. Clearly either c or e is to be executed on P2 in slot 4, so that e or c is on P1. Suppose task c is on P1 and e on P2 (the other case is symmetrical). Then, Case 3 is infeasible due to bus capacity constraint: the communications $t_0 \rightarrow e, e \rightarrow s_1, t_0 \rightarrow g, b \rightarrow c, c \rightarrow h$ are to be done in time slots $3, 4, 5, 6$.

Case 4: tasks s_0, t_0, s_1 are on P2 and a, h on P1 (Figure 2-(g)).

Tasks b, c, d, e, f, g are to be filled in the remaining time slots, i.e. slots $3, 4$ on P2 and slots $2, 3, 4, 5$ on P1. Using similar arguments as in the Case 1, we can see that

the task on P2 in slot 3 should be among b, d, f and the one in slot 4 be among c, e, g. The two tasks on P2 must be in precedence relation, otherwise there are too many communications on the bus in slots $3, 4$ which would make the schedule infeasible. Moreover, those two tasks cannot be f, g, otherwise e and c have to be executed on processor P1 in slots $4, 5$, so that there are 3 communications to be dealt with by the bus among time slots $5, 6$, i.e. $g \rightarrow h, c \rightarrow s_1, e \rightarrow s_1$. In Figure 2-(h) we give an example of feasible schedule.

We can therefore conclude that a schedule is feasible if and only if either b, c or d, e are on processor P2 in slots $3, 4$. Moreover, the bus is occupied in slots $2, 3, \ldots, 7$. ∎

Lemma 3.4 *Consider task graph $\mathcal{C}$ in Figure 3. In any feasible schedule of length 9 of the graph, tasks a, h, k are executed on the same processor (say P1) in time slots $1, 6, 7$. Tasks s_0 and t_1 are executed on the other processor (say P2) in the slots 1 and 9. Either tasks b, c or d, e or f, g are executed on P2 in the slots $3, 4$. Moreover, the bus is occupied in slots $2, 3, \ldots, 8$.*

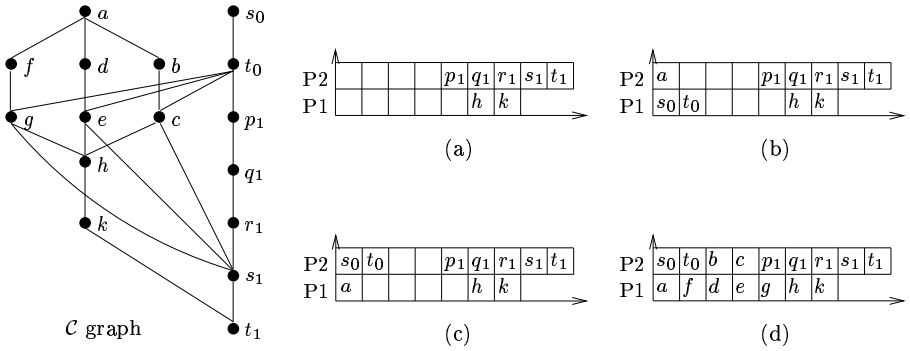


Figure 3: Task graph $\mathcal{C}$ and partial schedules for it.

Proof. The proof is similar to that of Lemma 3.3. We assume again that t_1 is executed by P2. Given the number of predecessors of s_1 (resp. t_1), in any feasible schedule of length 9, task s_1 (resp. t_1) is to be executed on P2 in slot 8 (resp. 9).

Processor P1 is always busy except the last two time slots. P2 never idles. If we look backwards from the end of the schedule, only an *immediate* predecessor of t_1, that is task k, can be executed in time slot 7 on P1. As processor P2 never idles, it has to execute tasks q_1, r_1 in slots $6, 7$. Similarly, task h must be processed on P1 in time slot 6 and task p_1 on P2 in time slot 5, see Figure 3-(a).

Since only tasks a and s_0 have no predecessor, two cases are to be analyzed:

Case 1: tasks s_0, t_0 are on P1 and a on P2 (Figure 3-(b)).

Given that tasks a and h are executed on different processors and there are three disjoint paths from a to h, there are at least three communications during time slots $2, 3, 4, 5$. There are four disjoint paths from task t_0 to s_1 (and these paths have no common arcs with the previous three). Thus, there are four more communications to be dealt with on the bus during time slots $3, 4, 5, 6, 7$. Consequently, Case 1 is infeasible since there cannot be 7 communications during time slots $2, 3, 4, 5, 6, 7$.

Case 2: tasks s_0, t_0 are on P2 and a on P1 (Figure 3-(c)).

Clearly, tasks b, c, d, e, f, g are to be executed on P1 in time slots $2, 3, 4, 5$ and on P2 in time slots $3, 4$. Task in slot 3 on P2 is among b, d, f (immediate successor of a) and the one in slot 4 is among c, e, g (immediate predecessor of h). Moreover, the two tasks on P2 must be in precedence relation, otherwise there are too many communications to occur on the bus in slots $3, 4$ which will make the schedule infeasible. In Figure 3-(d) we give an example of feasible schedule.

Therefore, Case 2 is feasible if and only if either b, c or d, e or f, g are on processor P2 in slots $3, 4$. Moreover, the bus is occupied in slots $2, 3, \ldots, 8$. ∎

Proof of Lemma 3.2. Given an instance of ONE-IN-THREE 3SAT, we construct the following instance for the scheduling problem, such that there exists a feasible schedule σ if and only if the ONE-IN-THREE 3SAT problem has a solution.

Let $U = \{u_1, \cdots, u_n\}$ be the set of variables and $C = \{C_1, \cdots, C_m\}$ the set of clauses in the instance of ONE-IN-THREE 3SAT problem.

We construct now a task graph $G = (V, E)$ of $12n + 14m + 2$ tasks for the associated scheduling problem. Figure 4-(a) illustrates the global form of the graph, and Figure 4-(b),(c) the intermediate stages corresponding to $\mathcal{V}$-graphs and $\mathcal{C}$-graphs, respectively.

The tasks in $G = (V, E)$ are

$$\begin{aligned} V &= \{s_0, t_0\} \bigcup \{a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i, p_i, q_i, s_i, t_i,\ 1 \leq i \leq n+m\} \\ &\quad \bigcup \{k_i, r_i,\ n+1 \leq i \leq n+m\}, \end{aligned}$$

and the precedence relations are:

- $s_0 \rightarrow t_0$;
- for $1 \leq i \leq n+m$, $a_i \rightarrow b_i \rightarrow c_i \rightarrow h_i$, $a_i \rightarrow d_i \rightarrow e_i \rightarrow h_i$, $a_i \rightarrow f_i \rightarrow g_i \rightarrow h_i$, $t_{i-1} \rightarrow c_i \rightarrow s_i$, $t_{i-1} \rightarrow e_i \rightarrow s_i$, $t_{i-1} \rightarrow g_i$, $t_{i-1} \rightarrow p_i \rightarrow q_i$, $s_i \rightarrow t_i$;
- $h_i \rightarrow t_i$, $q_i \rightarrow s_i$, for $1 \leq i \leq n$;
- $h_i \rightarrow k_i \rightarrow t_i$, $q_i \rightarrow r_i \rightarrow s_i$, $g_i \rightarrow s_i$, for $n+1 \leq i \leq n+m$;
- Consider clause C_j, $1 \leq j \leq m$. Suppose the occurrence of variable u_i in C_j is unnegated and u_i is the first (resp. the second or the third) literal in C_i. There is an arc $b_i \rightarrow b_{n+j}$ (resp. $b_i \rightarrow d_{n+j}$ or $b_i \rightarrow f_{n+j}$). If the occurrence of u_i in C_j is negated, the arc is from task d_i, i.e. $d_i \rightarrow b_{n+j}$ (resp. $d_i \rightarrow d_{n+j}$ or $d_i \rightarrow f_{n+j}$).

The time limit for the schedule of G is $T = 6n + 7m + 2$.

Figure 4-(d) illustrates the task graph that we construct for an instance of ONE-IN-THREE 3SAT problem. The clauses are: $C_1 = u + \bar{v} + w$ and $C_2 = \bar{u} + v + \bar{y}$. The time limit is 40.

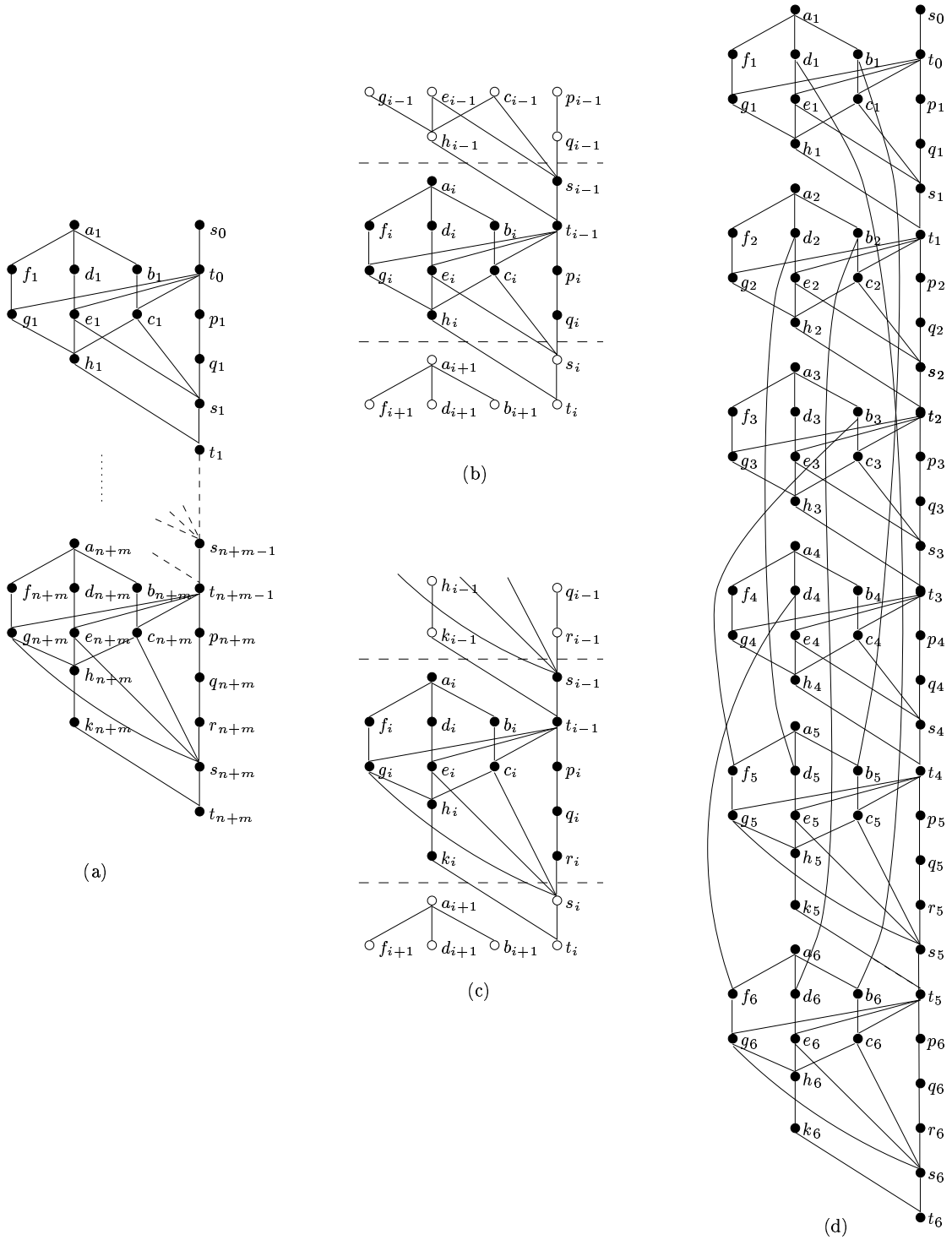


Figure 4: Task graph instance of $P2,\ B1 \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{max}$ corresponding to the instance of ONE-IN-THREE 3SAT problem.

Observe that task t_{n+m} is the common successor of all the other tasks, so that the soonest we can execute t_{n+m} is time slot T. Moreover, the processor executing this task never idles and the other idles only during the last two time slots.

In what follows, we first show that for any solution of the problem ONE-IN-THREE 3SAT there is a feasible solution for the scheduling problem. Using Lemmas 3.3 and 3.4 we construct a feasible schedule for the graph G by simply concatenating n schedules of $\mathcal{V}$ graphs (c.f. Figure 4-(b)) with m schedules of $\mathcal{C}$ graphs (c.f. Figure 4-(c)) in such a way that tasks b_i, c_i (resp. d_i, e_i) are executed on processor P2 if u_i is true (resp. false), $1 \leq i \leq n$, and tasks b_{n+j}, c_{n+j} (resp. d_{n+j}, e_{n+j} or f_{n+j}, g_{n+j}) are executed on P2 if the first (resp. second or third) literal is the true one in clause C_j, $1 \leq j \leq m$. Since there is exactly one true literal in each clause, there are no communications associated with arcs connecting $\mathcal{V}$ graphs to $\mathcal{C}$ graphs (i.e. the extreme vertices of these arcs are assigned to the same processors). Thus the simple concatenation of individual schedules of $\mathcal{V}$ graphs to $\mathcal{C}$ graphs yields a feasible schedule of the task graph.

We now show that for any feasible schedule there is a satisfying truth assignment for the corresponding ONE-IN-THREE 3SAT instance.

In any feasible schedule, task t_{n+m} is executed at time T. Given the number of predecessors of t_{n+m-1}, we cannot execute this task *before* time slot $T-7$. Consider the subgraph consisting of tasks inbetween t_{n+m-1} and t_{n+m} (i.e. t_{n+m-1} and t_{n+m} and those which are both successors of t_{n+m-1} and predecessors of t_{n+m}). Suppose that tasks t_{n+m-1} and t_{n+m} are executed in time slots α and β, respectively. Then, due to communication delay, at most one task can be processed in time slot $\alpha+1$ and at most one in slot $\beta-1$. Thus, at most $2(\beta-\alpha-3)+2$ tasks can be processed inbetween α and β. Since the number of tasks which are both successors of t_{n+m-1} and predecessors of t_{n+m} is 9, $\beta-\alpha$ should be *at least* 7. Thus, in any feasible schedule task t_{n+m-1} is executed exactly 7 time slots before task t_{n+m}.

Iterating backwards in the same manner, we conclude that tasks t_{n+i} should be executed in slot $6n+7i+2$ in any feasible schedule, $0 \leq i \leq m$.

Similarly, we can show that tasks t_{n-1} is executed exactly 6 time slots before task t_n. Iterating backwards in the same way, we obtain that tasks t_i should be executed at slot $6i+2$ in any feasible schedule, $0 \leq i \leq n-1$.

It is easy to see now, using Lemma 3.3, that the predecessors of t_1 must be executed as in Lemma 3.3, i.e. tasks $s_0, t_0, p_1, q_1, s_1, t_1$ and either b_1, c_1 or d_1, e_1 on the same processor (say P2), and all other tasks on P1. If we look at the schedule starting from slot 7, we can easily recognize the same scheduling problem for predecessors of t_2, i.e. tasks s_1, t_1, p_2, q_2, s_2 and $a_2, b_2, c_2, d_2, e_2, f_2, g_2, h_2$, with tasks s_1, t_1 already scheduled on P2 in slots $7, 8$. Repeating these arguments we see that in any feasible schedule either b_i, c_i or d_i, e_i are on P2, $1 \leq i \leq n$.

Similarly, using Lemma 3.4, we can conclude that in any feasible schedule, either b_i, c_i or d_i, e_i or f_i, g_i are on P2, $n+1 \leq i \leq n+m$.

Thus, the truth assignment for the variables is such that if b_i, c_i are assigned to P2, then u_i is "true", otherwise u_i is "false", $1 \leq i \leq n$.

According to Lemmas 3.3 and 3.4, the bus is already occupied from slot 2 to slot $T-1$ in any feasible schedule. Thus, there are no communications associated with arcs connecting $\mathcal{V}$ graphs to $\mathcal{C}$ graphs. Thus extreme vertices of these arcs are assigned to the same processors. Therefore, such a truth assignment yields a solution to the ONE-IN-THREE 3SAT problem. ∎

Theorem 3.2 *Problem* $P2,\ B1 \mid prec,\ c=1\ (i.d.),\ p_j=1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

And, for any fixed number of processors, it is sufficient to add some critical chains in the task graph to obtain

Corollary 3.2 *Problem* $Pm,\ B1 \mid prec,\ c=1\ (i.d.),\ p_j=1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

It is worthwhile noticing that scheduling of general task graphs with fixed number of processors is a notoriously difficult problem. In the case of no interprocessor communications, the two-processor case is polynomial owing to Coffman and Graham [5]. When there are three or more processors, the problem is still open, regardless important research effort (see e.g. [2]). In the case of UET-UCT, the complexity is unknown even for two processors. The result of Theorem 3.2 indicates again that the bus constraint makes the scheduling problem more difficult.

4 Scheduling with Common-Data Communication Semantics

In this section we consider scheduling problems with common-data communication semantics. Again, we shall analyze the single-bus system ($B = 1$). The more general case will be considered in Section 5. The results are similar to those of Section 3. We first consider the case of non-broadcasting communication.

Note that when there are unlimited number of processors, the makespan minimization problem was shown to be $\mathcal{NP}$-hard for binary trees and independent-data communication semantics. This result remains valid in the case of common-data semantics, as no two communications are required between a task and two immediate successors of the task. Thus, we have

Theorem 4.1 *Problem* $\overline{P},\ B1 \mid$ *binary tree,* $c = 1$ *(c.d.),* $p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

Corollary 4.1 *Problem* $P,\ B1 \mid$ *binary tree,* $c = 1$ *(c.d.),* $p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

When we have two processors and a general task graph, the makespan minimization problem was shown to be $\mathcal{NP}$-hard for independent-data communication

semantics. This result still holds for common-data semantics. However, the proof has to be modified.

Lemma 4.1 *Consider task graph $\mathcal{V}$ in Figure 5 with common-data communication semantics. In any feasible schedule of length 8 of the graph, tasks b and c are executed together on one processor, and tasks d and e are on the other. Moreover, the bus is occupied in slots $2, 3, \ldots, 7$.*

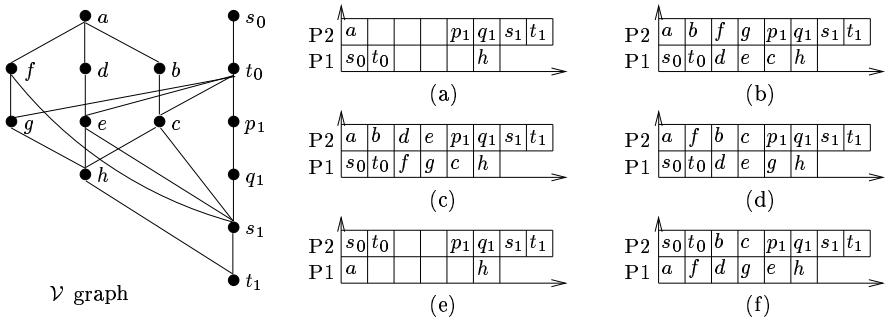


Figure 5: Task graph $\mathcal{V}$ and partial schedules for it.

Proof. Consider the case where t_1 is executed by P2 (the other case is symmetric). In any feasible schedule of length 8, processor P2 never idles and P1 idles only in time slots 7 and 8, since t_1 has 13 predecessors.

If we look from the end of the schedule and we schedule backwards, clearly in time slot 7 task s_1 should be executed on P2 (it has 10 predecessors). Thus, task h is to be processed on P1 in slot 6. No predecessors of h can be executed on P2 in time slots $5, 6$, tasks p_1, q_1 become the only choice.

In the first time slot only tasks a and s_0 are available for execution, they have to be executed on two different processors in slot 1 in order to avoid idling. Moreover, t_0 should be executed on the same processor as s_0, otherwise this processor would be idle in time slot 2. Therefore, two cases are to be analyzed:

Case 1: tasks s_0, t_0, h are on P1 and a, s_1 on P2 (Figure 5-(a)).

Tasks b, c, d, e, f, g are to be filled in the remaining time slots, i.e. slots $2, 3, 4$ on P2 and slots $3, 4, 5$ on P1. It is easy to see that exactly one task among b, d, f is on P1 (in slot 3), while the other two are to be processed on P2, otherwise either P1 or P2 will idle in some time slot. Hence, tasks c, e, g are to be executed in slots $4, 5$ on P1 and in slot 4 on P2 (since the task executed in slot 4 on P2 must be an *immediate* predecessor of h).

Focus now on tasks f, g. Those two tasks are either together on some processor, or separate, i.e. f on P2 and g on P1.

Case 1.a : tasks f, g are on P2 (Figure 5-(b)).

Clearly tasks c, e are on P1 in slots $4, 5$. In this case, there are three communications to be dealt with by bus in slots $5, 6$, i.e. $c \rightarrow s_1$, $e \rightarrow s_1$ and $g \rightarrow h$. Thus, there is no feasible schedule.

Case 1.b : tasks f, g are on P1 (Figure 5-(c)).

Suppose c is the task among c, e which is assigned to P1 (the other case is similar). Thus, tasks b, d, e are on P2. This results in five communications $b \rightarrow c$, $t_0 \rightarrow e$, $c \rightarrow s_1$, $f \rightarrow s_1$ and $e \rightarrow h$ to be done in slots $3, 4, 5, 6$. Hence this case is again infeasible due to the bus constraint.

Case 1.c : tasks f is on P2 and g is on P1 (Figure 5-(d)).

Suppose the task on P1 in slot 3 is d. It is easy to see that e is to be processed on P1 also. Thus, b, c and d, e are never on the same processor in this subcase. In Figure 5-(d) we provide an example of feasible schedule for the Case 1.c.

Case 2: tasks s_0, t_0, s_1 are on P2 and a, h on P1 (Figure 5-(e)).

This case is similar to Case 4 of the proof of Lemma 3.3. However, since in the case of independent-data semantics there are two communication from t_0 to the two tasks among c, e, g on P1, here, one of those communications does not occur anymore, and is replaced by the communication $f \rightarrow s_1$. The schedule in Figure 5-(f) is an example of feasible schedule.

We can therefore conclude that a schedule is feasible if and only if tasks b and c are executed together on one processor, and tasks d and e are on the other. Moreover, the bus is occupied in slots $2, 3, \ldots, 7$. ∎

Lemma 4.2 *Consider task graph $\mathcal{C}$ in Figure 6 with common-data communication semantics. In any feasible schedule of length 8 of the graph, two tasks among b, d, f are executed on the same processor in time slots $2, 3$ and the third one is on the other processor in slot 3. Moreover, the bus is occupied in slots $2, 3, \ldots, 7$.*

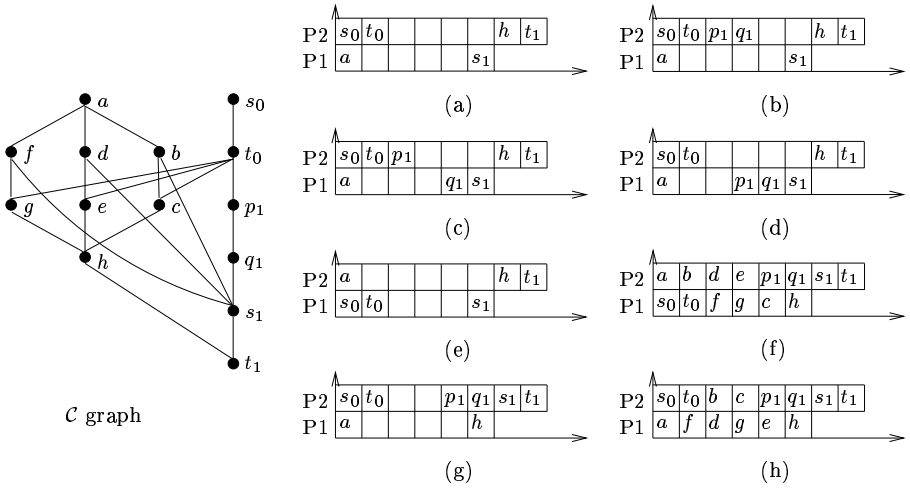


Figure 6: Task graph $\mathcal{C}$ and partial schedules for it.

Proof. We assume that t_1 is executed on P2 (the other case is symmetric). Since t_1 has 13 predecessors, any feasible schedule of length 8 has both processors always busy except for P1 which idles only in the last two time slots. It is easy to see that in time slot 7 either task h or task s_1 should be executed on P2. Thus on P1 in slot 6 either task s_1 or h should be processed.

There are only two initial tasks in the graph, thus in the first slot both tasks a and s_0 are processed. It is clear that only t_0 can be executed in the second slot by the processor that executes s_0. Hence, four cases are to be analyzed:

Case 1: tasks s_0, t_0, h are on P2 in slots $1, 2, 7$ and a, s_1 on P1 in slots $1, 6$ (Figure 6-(a)).

We consider three subcases to show that this configuration results in no feasible solution.

Case 1.a : tasks p_1, q_1 are on P2 in slots $3, 4$ (Figure 6-(b)).

Since on P2 in slots $5, 6$ there cannot be any of tasks b, d, f, there should be two tasks among c, e, g on this processor (the third one has to be on P1). There are two communications from two tasks on P1 among b, d, f to their two successors on P2 in slots $5, 6$. Moreover, there is also a communication from t_0 to the third task (the one on P1) among c, e, g. In slot 5 there is a communication from q_1 to s_1. Therefore, Case 1.a is infeasible since there cannot be 4 communications in time slots $3, 4, 5$.

Case 1.b : task p_1 is on P2 in slot 3 and q_1 on P1 in slot 5 (Figure 6-(c)).

Either tasks b, d, f or only two of those tasks and one among c, e, g are executed on P1. In the former situation, there are three communications from tasks b, d, f to c, e, g in time slots $3, 4, 5$, and in slot 4 there should be a communication from p_1 to q_1. Thus this situation yields no feasible schedule. In the latter situation, let b, d, e be on P1 and f, c, g on P2 (other assignments can be analyzed in the same way). The communications are: $a \rightarrow f$, $b \rightarrow c$, $p_1 \rightarrow q_1$, $t_0 \rightarrow e$ and $f \rightarrow s_1$. This situation again yields an infeasible schedule since the five communications should take place on the bus within four time slots: $2, 3, 4, 5$. Therefore, Case 1.b is infeasible.

Case 1.c : tasks p_1 and q_1 are on P1 in slots $4, 5$ (Figure 6-(d)).

In slot 3 there is a communication from t_0 to p_1. No task among c, e, g can be on P1 in slots $2, 3$. Suppose b, d are on P1. Since there are at least five communications in slots $2, 3, 4, 5$, i.e. $a \rightarrow f$, $t_0 \rightarrow p_1$, $b \rightarrow c$, $d \rightarrow e$ and $f \rightarrow s_1$, Case 1.c is infeasible.

Case 2: s_0, t_0, s_1 are on P1 and a, h on P2 (Figure 6-(e)).

Tasks p_1 and q_1 are to be executed on P1, otherwise the communication delay is at least 2 slots in the path $t_0 \rightarrow p_1 \rightarrow q_1 \rightarrow s_1$. Only one task among tasks b, c, d, e, f, g is executed on P1 in one of the slots $3, 4, 5$. Hence, there are at least one successor of t_0 and two predecessors of s_1 which are on P2, so that there are at least three communications (one from t_0 and two to s_1) to deal with on the bus in slots $3, 4, 5$. Suppose now task b (resp. c) is on P1 (the other cases are symmetric). Then the communication $b \rightarrow c$ should also take place in slots $3, 4, 5$ due to the fact that c should be executed in slot 6 or earlier (resp. b should be executed in slot 2 or later) on P2. Case 2 is therefore infeasible due to bus capacity constraint.

Case 3: tasks s_0, t_0, h are on P1 and a, s_1 on P2 (Figure 6-(f)).

It is easy to see that only a task among b, d, f can be executed on P1 in slot 3, and only a task among c, e, g can be executed on P2 in slot 4. Thus, one can immediately conclude that tasks b, d, f are to be executed on P1 in slot 3 and on P2 in slots $2, 3$. An example of feasible schedule for this case is illustrated in Figure 6-(f).

Case 4: tasks s_0, t_0, s_1 are on P2 and a, h on P1 (Figure 6-(g)).

As in Case 3, tasks b, d, f should be executed in the slots $2, 3$, i.e. on P1 in slots $2, 3$ and on P2 in slot 3. The schedule is feasible in Case 4 if and only if the two tasks on P2 among b, c, d, e, f, g are in precedence relation, i.e. they should be b, c or d, e or f, g. An example of such a schedule is provided in Figure 6-(h).

Thus, a schedule of length 8 is feasible if and only if two tasks among b, d, f are on the same processor (slots $2, 3$) and the third one on the other processor (slot 3). Moreover, the bus is occupied in slots $2, 3, \ldots, 7$. ∎

We are now in a position to present a polynomial transformation from the well-known *NOT-ALL-EQUAL 3SAT* problem (which is $\mathcal{NP}$-complete cf. [10]) to $\overline{P}$, $B1 \mid \textit{binary tree}, \; c = 1 \; (c.d.), \; p_j = 1 \mid C_{\max}$. We recall the definition of NOT-ALL-EQUAL 3SAT problem.

NOT-ALL-EQUAL 3SAT problem: Given a set of variables U, a collection of clauses C over U such that each clause has three literals, is there a truth assignment for U such that each clause in C has at least one true literal and at least one false literal?

Lemma 4.3 *The NOT-ALL-EQUAL 3SAT problem polynomially transforms to* $P2$, $B1 \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$.

Proof. The proof similar to that of Lemma 3.2. We provide here a sketch of the proof. The interested reader can complete it by mimicking the arguments of the proof of Lemma 3.2.

Let $U = \{u_1, \cdots, u_n\}$ be the set of variables and $C = \{C_1, \cdots, C_m\}$ the set of clauses in the instance of ONE-IN-THREE 3SAT problem.

The task graph $G = (V, E)$ constructed from a given instance of NOT-ALL-EQUAL 3SAT problem is obtained by concatenating n $\mathcal{V}$-graphs (see Figure 7-(b)) with m $\mathcal{C}$-graphs (see Figure 7-(c)). The global form of the graph is illustrated in Figure 7-(a). Figure 7-(d) illustrates the task graph that we construct for an example of NOT-ALL-EQUAL 3SAT problem where the clauses are: $C_1 = u + \bar{v} + w$ and $C_2 = \bar{u} + v + \bar{y}$.

In any feasible schedule of $\mathcal{C}$-graphs, two tasks among b, d, f are on one processor and the other task on the other processor. This motivates the choice of NOT-ALL-EQUAL 3SAT as source problem in our proof. Indeed, as we will see later on in the proof, the entering arcs (from $\mathcal{V}$-graphs) to tasks b, d, f of $\mathcal{C}$-graphs do not correspond to any communication in a feasible schedule. Thus, we can define truth assignment in such a way that the clause has two true literals if two of the tasks b, d, f are on P2 and one true literal if only one of them is on P2.

The formal definition of the task graph is given as follows. Graph $G = (V, E)$ has $12(n + m) + 2$ tasks. They are

$$V = \{s_0, t_0\} \bigcup \{a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i, p_i, q_i, s_i, t_i,\ 1 \leq i \leq n + m\}.$$

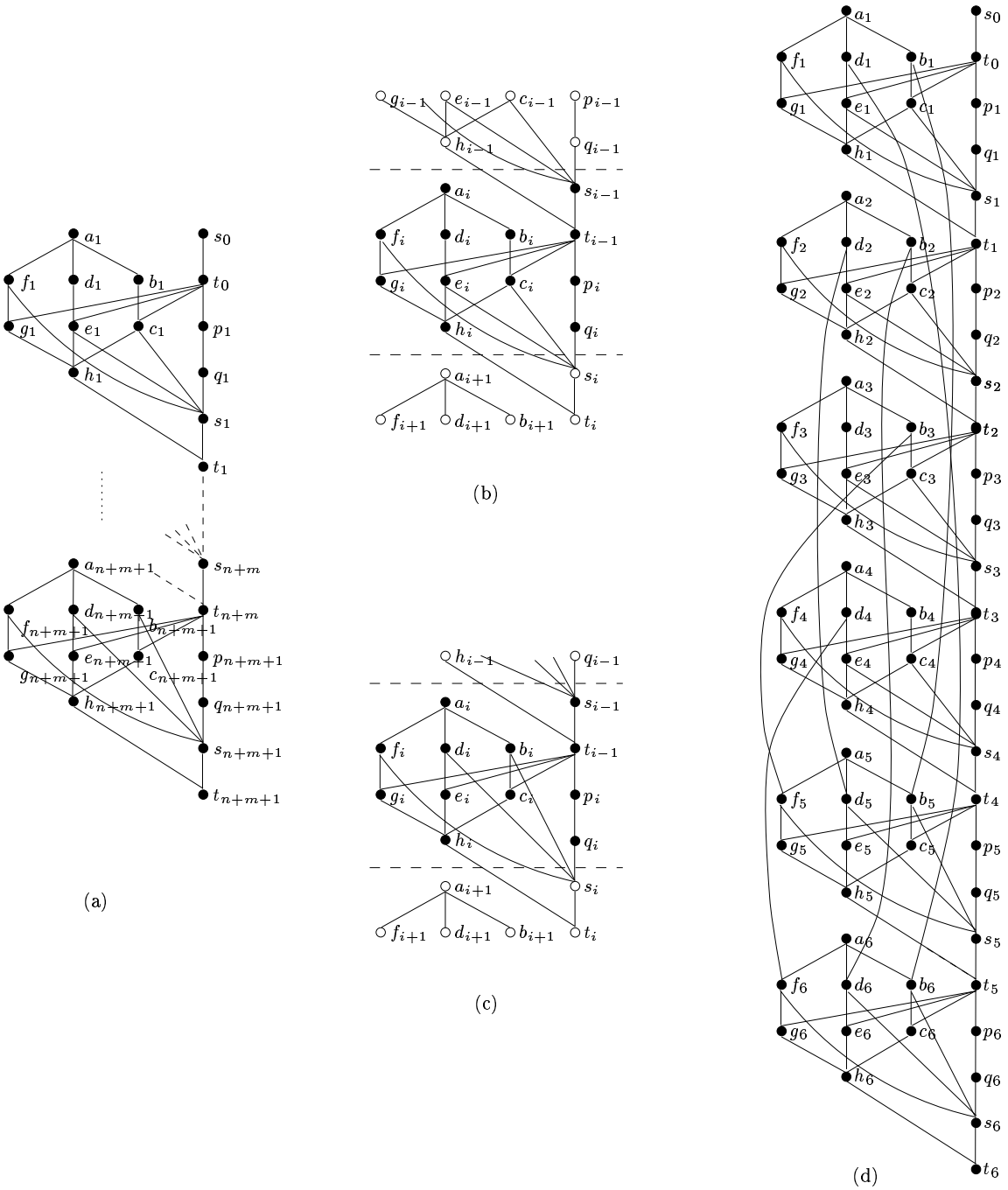


Figure 7: Task graph instance of $P2,\ B1 \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ corresponding to the instance of NOT-ALL-EQUAL 3SAT problem.

The precedence relations are:

- $s_0 \rightarrow t_0$;
- for $1 \leq i \leq n+m$, $a_i \rightarrow b_i \rightarrow c_i \rightarrow h_i$, $a_i \rightarrow d_i \rightarrow e_i \rightarrow h_i$, $a_i \rightarrow f_i \rightarrow g_i \rightarrow h_i$, $t_{i-1} \rightarrow c_i$, $t_{i-1} \rightarrow e_i$, $t_{i-1} \rightarrow g_i$, $f_i \rightarrow s_i$, $h_i \rightarrow t_i$, $t_{i-1} \rightarrow p_i \rightarrow q_i \rightarrow s_i \rightarrow t_i$;
- $c_i \rightarrow s_i$, $e_i \rightarrow s_i$, for $1 \leq i \leq n$;
- $b_i \rightarrow s_i$, $d_i \rightarrow s_i$, for $n+1 \leq i \leq n+m$;
- Consider clause C_j, $1 \leq j \leq m$. If the occurrence of variable u_i in C_j is unnegated and u_i is the first (resp. the second or the third) literal in C_j, there is an arc $b_i \rightarrow b_{n+j}$ (resp. $b_i \rightarrow d_{n+j}$ or $b_i \rightarrow f_{n+j}$). Otherwise, if the occurrence of u_i in C_j is negated and u_i is the first (resp. the second or the third) literal in C_j, then the arc is from task d_i (i.e. $d_i \rightarrow b_{n+j}$ (resp. $d_i \rightarrow d_{n+j}$ or $d_i \rightarrow f_{n+j}$)).

The time limit for the schedule of G is $T = 6(n+m)+2$. The time limit for the example of Figure 7-(d) is 38.

Given a solution of an instance of NOT-ALL-EQUAL 3SAT problem, we construct a feasible schedule in the following way. We concatenate n schedules for the graph $\mathcal{V}$ (Cases 1.c or 2 in the proof of Lemma 4.1) with m schedules of graph $\mathcal{C}$ (Cases 3 and 4 in the proof of Lemma 4.2). For all $1 \leq i \leq n$, tasks b_i, c_i are assigned to P2 (resp. P1) if variable u_i is true (resp. false).

Consider now clause C_j, $n \leq j \leq n+m$. If there are two true literals in C_j, then we execute tasks b_{n+j} and d_{n+j} (resp. b_{n+j} and f_{n+j} or d_{n+j} and f_{n+j}) on P2 provided that the first two literals (resp. the first and the third, or, the second and the third) in C_j are true. If, however, there is only one true literal in C_j, we execute task b_{n+j} (resp. d_{n+j} or f_{n+j}) on P2 provided that the first (resp. the second or the third) literal is the true one.

With such a task assignment, one can easily see that the extreme tasks of arcs connecting $\mathcal{V}$-graphs to $\mathcal{C}$-graphs are assigned to the same processors. Thus, there is no communication associated with such arcs so that the simple concatenation of individual schedules of $\mathcal{V}$-graphs and $\mathcal{C}$-graphs results in a feasible schedule.

We now prove that for any feasible schedule there is a satisfying truth assignment for the corresponding instance of NOT-ALL-EQUAL 3SAT problem.

Analogously to the proof of Lemma 3.2, we can show that tasks t_i has to be executed at time $6i + 2$, $0 \leq i \leq m + n$, and that partial schedules for the $\mathcal{V}$-subgraphs and $\mathcal{C}$-subgraphs have the properties stated in Lemmas 4.1 and 4.2.

For all $1 \leq i \leq n$, variable u_i is assigned a "true" value if task b_i is on P2 and "false" otherwise. Due to the fact that in the feasible schedule there is no communication associated with arcs connecting $\mathcal{V}$-graphs to $\mathcal{C}$-graphs, such a truth assignment guarantees that in each clause C_j, $1 \leq j \leq m$, there is at least one "true" (resp. "false") literal (corresponding to one of the tasks b_i and d_i, $1 \leq i \leq n$, which is assigned to P2 (resp. P1) and which has an arc to the j-th $\mathcal{C}$-graph).

Hence we obtain a truth assignment for the corresponding instance of NOT-ALL-EQUAL 3SAT problem.

Note that changing all truth values for all variables to the opposite value yields another solution to the 3SAT problem. ∎

As a consequence, we obtain

Theorem 4.2 *Problem* $P2,\ B1 \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

Corollary 4.2 *Problem* $Pm,\ B1 \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ *is* $\mathcal{NP}$*-complete.*

When broadcasting communication is under consideration, all the results of this section still hold. Indeed, in the case of unlimited processors, the task graph is a binary tree and there is at most one communication from a task to its immediate successors. In the case of 2 processors, broadcasting or nonbroadcasting makes no difference.

5 Extensions to the General Communication Capacity Constraint

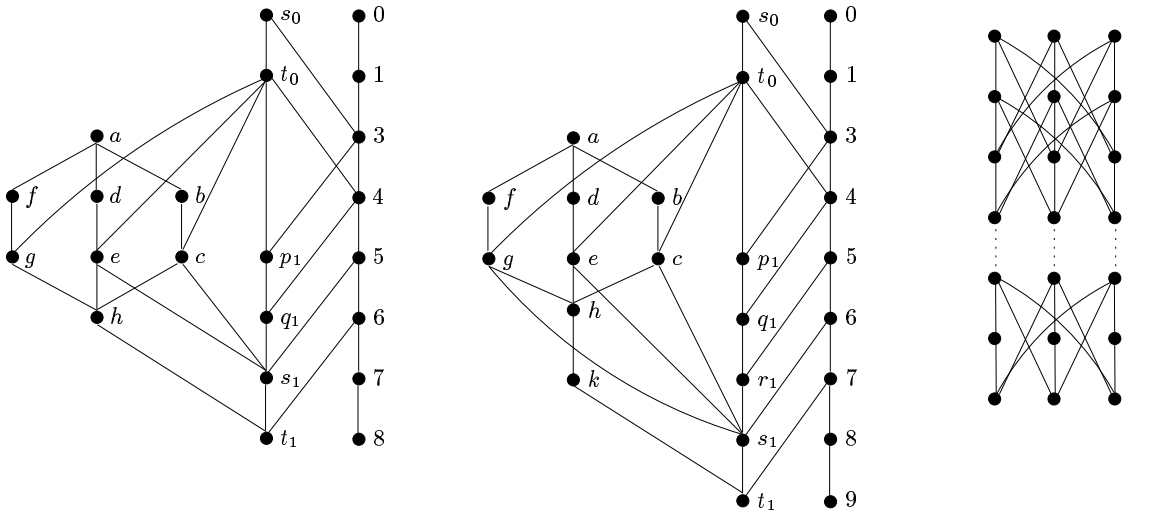
In this section we consider the general case of fixed communication capacity constraint $b \geq 1$, i.e. the number of messages that can be handled in each timeslot is at most b.

We prove that for unlimited number of processors the results showed in the previous sections are still valid when the communication capacity is b for both communication semantics, i.e. common-data or independent-data communication semantics.

Theorem 5.1 *For any fixed $b \geq 1$, problems $\overline{P},\ Bb \mid binary\ tree,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$ and $\overline{P},\ Bb \mid binary\ tree,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ are $\mathcal{NP}$-complete.*

Proof. The construction is identical to the one in the proof of lemma 3.1 to which we add for each unit of additional communication capacity a binary tree containing $T-1$ critical chains and $T-2$ critical communications (similar to the rightmost tree in figure 1). More precisely, one chain has length T (i.e. T tasks). Attached to this chain there are $T-2$ chains of lengths $T-2, T-3, \ldots, 1$ which are successors of the 1st, 2nd, ..., $T-2$nd task, respectively, of the first chain. Therefore, those $b-1$ new trees will occupy the extra communication capacity in slots $1, \cdots, T-2$ with critical communications. ∎

Corollary 5.1 *For any fixed $b \geq 1$, problems $P,\ Bb \mid binary\ tree,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$ and $P,\ Bb \mid binary\ tree,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ are $\mathcal{NP}$-complete.*



$\mathcal{V}$ graph (a) $\mathcal{C}$ graph (b) critical chains (c)

Figure 8: Task graph instance of $Pm,\ Bb \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$. (a) $\mathcal{V}$-graph. (b) $\mathcal{C}$-graph. (c) $m-2$ critical chains.

When the number of processors is m fixed *a priori*, the problem is still $\mathcal{NP}$-complete for any fixed communication capacity $b \leq (m-2)^2 + 1$ as follows:

Theorem 5.2 *For any fixed $m \geq 1$ and $1 \leq b \leq (m-2)^2+1$, problems $Pm,\ Bb \mid prec,\ c = 1\ (i.d.),\ p_j = 1 \mid C_{\max}$ and $Pm,\ Bb \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ are $\mathcal{NP}$-complete.*

Proof. The construction is similar to that of the two-processor case (see previous sections), to which we add $m-2$ critical chains of length T (cf. Figure 8-(c)) which are to be executed on the $m-2$ remaining processors. The original task graph of the two-processor case will use 1 unit of communication capacity.

For each additional unit of communication capacity, up to $(m-2)(m-3)$, we take a pair of critical chains and we add a communication arc from each task i,

$0 \leq i \leq T-3$ of one chain to task $i+2$ of the other chain (cf. Figure 8-(c)). It is clear that all these $T-2$ communications are critical.

If $(m-2)(m-3)+1 \leq b \leq (m-2)^2+1$, then we can add some more critical arcs between the tasks of critical chains introduced above and the tasks of the chain $s_0, t_0, \cdots, s_{|U|+|C|+1}, t_{|U|+|C|+1}$ in the original task graph, see Figure 8-(a),(b). Note that in this figure, we illustrate the construction for the independent-data communication semantics. The case of common-data is analogous. Again, one can easily see that these arcs correspond to critical communications. ∎

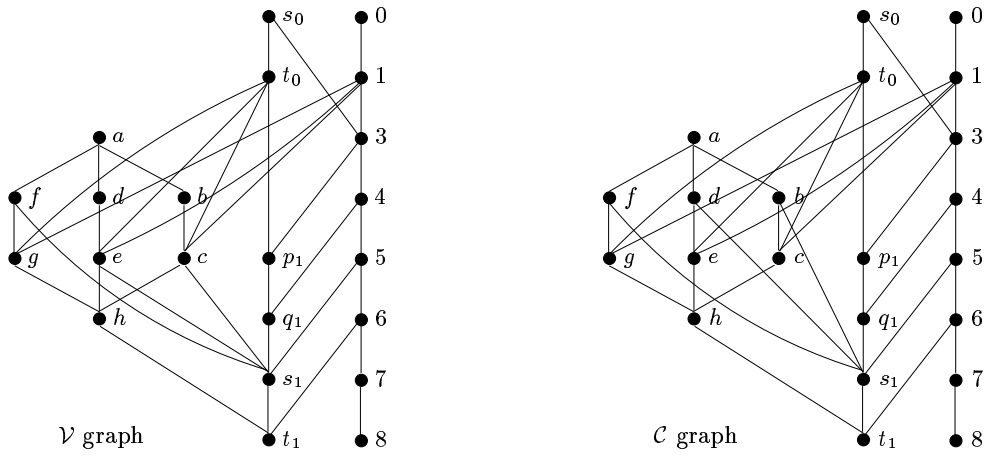
Observe that for common-data semantics, if the communication capacity b exceeds $m(m-1)$, the problem reduces to the classical UET-UCT scheduling problem with no communication capacity constraint. Thus the complexity is an open question.

Consider now broadcasting communications. Results of Theorem 5.1 and Corollary 5.1 are still valid due to the fact that only binary trees are involved so that there is at most one communication from each task. For the case of fixed number of processors, we have

Theorem 5.3 *For any fixed $m \geq 1$ and $1 \leq b \leq m-1$, problem $Pm,\ Bb-broadcast \mid prec,\ c=1\ (c.d.),\ p_j=1 \mid C_{\max}$ is $\mathcal{NP}$-complete.*

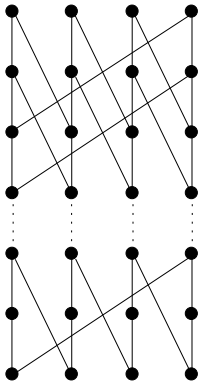
Proof. Consider first the case $m=3$. If $b=1$ the problem reduces to the case of $m=2$ by adding a critical chain in the task graph. Consider the case $b=2$. We add a critical chain as previously. Moreover, we add communications inbetween the critical chain and the original 2-processor task graph, cf. Figure 9.

For the case $m \geq 4$, we use again the task graph of the 2-processor problem and add $m-2$ critical chains, see Figure 10. The original 2-processor task graph will occupy one unit of communication capacity. For each additional unit of communication capacity, up to $m-2$, we take critical chains k and $l = (k \bmod m-2)+1$,



(a) (b)

Figure 9: Task graph instance of $P3,\ B2-broadcast \mid prec,\ c=1\ (c.d.),\ p_j=1 \mid C_{\max}$. (a) $\mathcal{V}$-graph. (b) $\mathcal{C}$-graph.



critical chains

Figure 10: Subgraph of critical chains for $Pm,\ Bb-broadcast \mid prec,\ c=1\ (c.d.),\ p_j=1 \mid C_{\max}$.

$1 \leq k \leq m-2$, and we add a communication arc from each task i, $0 \leq i \leq T-3$ of chain k to task $i+2$ of chain l (cf. Figure 10). It is clear that all these $T-2$ communications are critical. ∎

Observe that when $b \geq m$, the problem $Pm,\ Bb - broadcast \mid prec,\ c = 1\ (c.d.),\ p_j = 1 \mid C_{\max}$ reduces to the classical UET-UCT scheduling problem with no communication capacity constraint.

6 Concluding Remarks

In this paper, we have analyzed scheduling problems in a multiprocessor system with bounded communication capacity. The goal of these scheduling problems is to minimize the makespan when parallel programs are represented by a UET-UCT task graph. We have shown that these problems are strongly $\mathcal{NP}$-hard for binary trees with unbounded number of processors and for general graphs with two processors. Thus, the general scheduling problem of parallel programs with communication resource constraints is strongly $\mathcal{NP}$-hard.

We have considered several variants of the problem: communications with independent-data semantics and with common-data semantics. Our results have been extended to the case of broadcasting communications. Thus, the $\mathcal{NP}$-hardness of analyzed problems extends trivially to scheduling problems with general communication semantics and communication mechanisms. It is also easy to see that our results apply also to multiprocessor systems with shared memory.

In some scheduling problems (cf. [9, 27, 30, 31]), tasks are assumed to be preallocated to specific processors. Using the arguments of Goyal [9] and Veltman et al. [15, 30] who established the $\mathcal{NP}$-hardness of scheduling problem of chains on two processors with task preallocation and no communication delay, we can show that scheduling UET-UCT chains on two processors with bus constraint is strongly $\mathcal{NP}$-hard. The interested reader is referred to Finta and Liu [7].

In some parallel systems, communications are blocking in the sense that the processor sending and/or receiving a message cannot execute tasks until the communication is completed. The complexity issues of the scheduling problem with bus constraint and blocking communications are discussed in [7].

In general, adding constraints in scheduling problems can make problems easier or more difficult. The results of this paper clearly suggest that the constraint on communication capacity makes the UET-UCT scheduling problem more difficult.

This difficulty is also encountered in the search of polynomial solutions for specific cases. When there are two processors and the task graph is a tree, the polynomial algorithm proposed by Guinand and Trystram [12] for trees with UET-UCT still works, as the algorithm yields a schedule in which all communications are from processor 2 to processor 1, and each task communicates with at most one of its successors. Thus, the constraint on communication capacity has no effect on the schedule. For other more complex graphs, such as series-parallel graphs, existing polynomial algorithms (see Finta et al. [8]) do not extend easily to the case with bus constrain (counterexample can be constructed with a simple graph with 1 source task, 8 parallel tasks and 1 final task). We did not succeed in finding optimal polynomial algorithms for special classes of task graphs other than trees.

Finally, we remark that according to the impossibility theorem of Lenstra and Shmoys [18], if the problem of deciding whether there is a feasible schedule with length at most c is $\mathcal{NP}$-complete, then there is no polynomial time algorithm with performance guarantee ratio $(c+1)/c$, unless $\mathcal{P} = \mathcal{NP}$. In the case of traditional UET-UCT scheduling problems, various results are reported (see e.g. Hoogeveen [14]) showing that there are no efficient polynomial approximations. This kind of results do not seem to be easy in the case of scheduling with constraint on communication capacity.

Acknowledgements: The authors are grateful to the referees for pointing out errors in the first draft of the paper, and for their stimulating comments on the revised versions.

References

[1] F. Afrati, C. H. Papadimitriou, G. Papageorgiou, "Scheduling Dags to Minimize Time and Communication", in *Proc. 3rd Conf. VLSI Algorithms and Architectures AWOC*, Lecture Notes in Computer Science, **319** (1988), pp. 134-138, Springer-Verlag, Berlin.

[2] M. Bartusch, R.H. Mohring, and F.J. Radermacher, "M-Machine Unit Time Scheduling: A Report on Ongoing Research", *Lecture notes in economics and mathematical systems*, **304** (1988), pp 165-212, Springer, Berlin.

[3] P. Chretienne, "A Polynomial Algorithm to Optimally Schedule Tasks over a Virtual Distributed System under Tree-like Precedence Constraints", *Euro. J. Oper. Res.*, **43** (1989), pp. 225-230.

[4] P. Chretienne, C. Picouleau, "Scheduling with Communication Delays: a Survey", in *Scheduling Theory and Its Applications*, pp. 65–90, (Eds. P. Chretienne et al.), J. Wiley, 1995.

[5] E. G. Coffman, Jr. and R. L. Graham, "Optimal Scheduling for Two-Processor Systems", *Acta Informatica*, **1** (1972), pp 200–213.

[6] J. Y. Colin, P. Chretienne, "CPM Scheduling with Small Communication Delays", *Oper. Res.*, **39** (1991), pp. 680–684.

[7] L. Finta, Z. Liu, "Scheduling of Parallel Programs in Single-Bus Multiprocessor Systems", Rapport de Recherche INRIA, No. 2302, 1994.

[8] L. Finta, Z. Liu, I. Milis, E. Bampis, "Scheduling UET-UCT Series-Parallel Graphs on Two Processors", *Theoretical Computer Science*, **162** (1996).

[9] D. K. Goyal, "Scheduling Processor Bound Systems", Report No. CS-76-036, Comp. Sci. Dep., Washington State Univ., Pullman, WA. 1976.

[10] M. R. Garey, D.R. Johnson, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*, W. H. Freeman, San Francisco, 1979.

[11] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, "Optimization and Approximation in Deterministic Scheduling: A Survey", *Ann. Disc. Math.*, **5** (1979), pp. 287-326.

[12] F. Guinand, D. Trystram, "Optimal Scheduling of UECT Trees on Two Processors". Technical Report APACHE RR-93-03, IMAG, Grenoble, 1993.

[13] J. J. Hwang, *Deterministic Scheduling in Systems with Interprocessor Communication Times*, Ph.D. Dissertation, Computer and Information Sciences Department, Univ. of Florida, 1987.

[14] J. A. Hoogeveen, J. K. Lenstra, B. Veltman, "Three, Four, Five, Six, or the Complexity of Scheduling with Communication Delays", *Oper. Res. Lett.*, **16** (1994), pp. 129-137.

[15] J. A. Hoogeveen, S. L. van de Velde, B. Veltman, "Complexity of Scheduling Multiprocessor Tasks with Prespecified Processor Allocations", Technical Report of CWI, BS-R9211, 1992.

[16] E. L. Lawler, "Scheduling Trees on Multiprocessors with Unit Communication Delays". In *Proc. Workshop on Models and Algorithms for Planning and Scheduling Problems*, Villa Vigoni, Lake Como, Italy, June 14-18, 1993.

[17] C. Y. Lee, J. J. Hwang, Y. C. Chow, F. D. Anger, "Multiprocessor Scheduling with Interprocessor Delays", *Oper. Res. Letters*, **7** (1988), pp. 141-147.

[18] J. K. Lenstra, D. B. Shmoys, "Computing Near-Optimal Schedules", in *Scheduling Theory and Its Applications*, pp. 1-14, (Eds. P. Chretienne et al.), J. Wiley, 1995.

[19] J. K. Lenstra, M. Veldhorst, B. Veltman, "The Complexity of Scheduling Trees with Communication Delays". *J. of Algorithms*, **20** (1996), pp. 157-173.

[20] Z. Liu, "A Note on Graham's Bound." *Information Processing Letters*, **36** (1990), pp. 1-5.

[21] Z. Liu, J. Labetoulle, "A Heuristic Method for Loading and Scheduling Flexible Manufacturing Systems", *Proc. of the Intern. Conf. Control 88*, London, IEE Conference Publication No.285, pp.195-200, 1988.

[22] C. H. Papadimitriou, M. Yannakakis, "Towards an Architecture-Independent Analysis of Parallel Algorithms", *SIAM J. Comput.*, **19** (1990), pp. 322-338.

[23] M. L. Prastein, "Precedence-Constrained Scheduling with Minimum Time and Communication", Technical Report UILU-ENG-87-2207 , ACT-75, Coordinated Science Lab., Dep. of Computer Science, Univ. of Illinois at Urbana-Champaign, IL. USA, 1987.

[24] C. Picouleau, *Etude de problèmes d'optimisation dans les systèmes distribués*, Thèse de doctorat, Université Pierre et Marie Curie, France, 1992.

[25] C. Picouleau, "New Complexity Results on Scheduling with Small Communication Delays", *Disc. Appl. Math.*, **60** (1995), pp. 331-342.

[26] V. J. Rayward-Smith, "UET Scheduling with Unit Interprocessor Communication Delays", *Disc. Appl. Math.*, **18** (1987), pp. 55-71.

[27] V. J. Rayward-Smith, F. Warren Burton, G. J. Janacek, "Scheduling Parallel Programs Assuming Prelocation", in *Scheduling Theory and Its Applications*, pp. 145–166, (Eds. P. Chretienne et al.), J. Wiley, 1995.

[28] G. C. Sih, E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures", *IEEE Trans. on Parallel and Distributed Systems*, **4** (1993), pp. 175-187.

[29] T. Varvarigou, V. P. Roychowdhury, T. Kailath. "Scheduling In and Out Forests in the Presence of Communication Delays". In *Proc. Intern. Parallel Processing Symposium* (IPPS'93), Newport Beach, CA, April 1993, pp. 222-229.

[30] B. Veltman, *Multiprocessor Scheduling with Communication Delays*, Phd Thesis, CWI-Amsterdam, 1993.

[31] B. Veltman, B. J. Lageweg, J. K. Lenstra, "Multiprocessor Scheduling with Communication Delays", *Parallel Computing*, **16** (1990), pp. 173-182.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399