# Scheduling with communication delays

Rachid Saad*
21, rue Ziane Said, la Scala
El Biar, Alger
Algeria

ABSTRACT. In this paper, scheduling problems with communication delays are considered. Formally, we are given a partial order relation $\prec$ on a set of tasks $T$, a set of processors $P$ and a deadline $d$. Supposing that a unit communication delay between two tasks $a$ and $b$ such that $a \prec b$ occurs whenever $a$ and $b$ are scheduled on different processors, the question is: Can the tasks of $T$ be scheduled on $P$ within time $d$? It is shown here that the problem is NP-complete even if $d = 4$. Also for unlimited number of processors, C. Picouleau has shown that for $d = 8$ the problem is NP-complete. Here it is shown that it remains NP-complete for $d \geq 6$ but is polynomially solvable for $d < 6$, which closes the gap between $P$ and $NP$ for this problem, as regards the deadline.

## 1. Introduction

Multiple-machine scheduling theory is the study of constructing schedules of machine processing for a set of jobs in order to ensure the execution of all jobs in the set in a reasonable amount of time. Such problems arise in the parallelisation of algorithms, a model of parallel computation being represented by a network of processors communicating through a main memory and a set of tasks ("a program") to be executed as soon as possible. But what we observe then is that, the practical performance (the execution time in practice) of a computation does not perfectly match the gain (in time) that one would expect from a (theoretically) effective parallelisation. Anomalies of this kind account for the communication delays

*This work was done while the author was at the LRI, Bat 490 universite de Paris XI, 91405 Orsay cedex France.

JCMCC 18 (1995), pp. 214-224

involved in the communication between the processors. A realistic evaluation of a performance of an algorithm should then take into account such communication delays.

Formally, we are given a partial order relation $\prec$ on a set of tasks $T$, a set of processors $P$ and a deadline $d$. Supposing that a unit communication delay between two tasks $a$ and $b$ such that $a \prec b$ occurs whenever $a$ and $b$ are scheduled on different processors, the question is: Can the tasks of $T$ be scheduled on $P$ within time $d$? Previous work has been obtained in the area (see [?] and [?]). The complexities of some scheduling problems with communication delays have been studied by Picouleau [?] and Rayward Smith [?].

We propose here to strengthen some known NP-completeness results in order to obtain lower bounds on the polynomial approximation ratios and to locate more precisely the boundary between P and NP for these problems.

### 1.1. Definitions and notations

Let $T$ be a set of tasks endowed with a partial order relation $\prec$, $P$ be a set of $p$ processors and $d$ be an integer.

Throughout, each task will be supposed to have unit execution time. Moreover, for every two tasks $a$ and $b$ such that $a \prec b$, if $a$ and $b$ are scheduled on two different processors, then a unit communication delay will occur before the execution of $b$. The problem of deciding on the existence of a schedule of $T$ on $P$ within time $t \leq d$ is denoted by $SC(P, d)$. Moreover, if the number of processors is not limited (so $p \geq \mid T \mid$), the corresponding subproblem is denoted by $SC(\bar{P}, d)$. If in addition, the duplication of tasks is allowed, the corresponding decision problem will be denoted by $SC(P, d, dup)$. (Notice that the duplication of tasks can be usefull to decrease communication time). The optimisation versions of these problems are denoted by $SC(P)$ and $SC(\bar{P})$ respectively.

Let us add a few words on the encoding of a partial order relation in our decision problems. A precedence graph, as a graph of a partial order relation, is an acyclic directed graph and its vertices are called tasks. Conversely, every directed acyclic graph defines a partial order relation which is the transitive closure of the relation of the graph. As the computation of the transitive closure of a graph is easy, our partial order relations will be defined (encoded) simply as directed acyclic graphs in all our decision problems.

Examples of schedules with communication delays:
Let $T = \{a, b, c\}$ be a set of three tasks such that $a$ and $b$ precede $c$, and $P = \{P_1, P_2\}$ be a set of two processors. Then the earliest execution time of $T$ is $t = 3$, because the processor to which the task $c$ is assigned needs both "results" from $a$ and $b$, so either $a$ and $b$ are executed by the same processor,

or a communication delay will occur. Hence, $c$ cannot be executed at time $t = 2$. Similarly, if $a$ precedes both $b$ and $c$, the earliest execution time of $T$ is $t = 3$, because either $T$ is executed by a unique processor, or one of the tasks $b$ and $c$ is executed by a processor distinct from the one to which $a$ is assigned, which produces a communication delay. However, if duplication of tasks is allowed in this latter case, we could execute $T$ in time $t = 2$ by duplicating $a$.

Let $G$ be a partial order relation graph and $L_1$ be the set of vertices of $G$ that have no predecessors. Let us denote by $U$ the graph obtained from $G$ by adding a source node $x_0$ which precedes all the vertices of $L_1$.

For a vertex $x$ of $G$, we call *level of* $x$ and we denote $s(x)$ the integer defined by $s(x) = d_U(x_0, x)$, where $d_U(x, y)$ denotes the distance from $x$ to $y$ in $U$. In other words, $s(x)$ is the breadth first search level (search from $x_0$) of vertex $x$.

We denote by $L_i(G)$ (or simply $L_i$ if $G$ is obvious from the context) the set of vertices of level $i$ in $G$. For a subset $A$ of tasks, we denote by $\Gamma^+(A)$ (resp.$\Gamma^-(A)$) the set of all successors (resp. predecessors) in $G$ of the tasks of $A$. Simillarly, we define $\Gamma^k(A)$ (resp. $\Gamma^{-k}(A)$) as: $\Gamma^k(A) = \Gamma^+(\Gamma^{k-1}(A))$ (resp. $\Gamma^{-k}(A) = \Gamma^-(\Gamma^{-(k-1)}(A))$). For a task $x$, we denote by $\delta^+(x)$ (resp. $\delta^-(x)$) the out-degree (resp. in-degree) of $x$ as a vertex of $G$: $\delta^+(x) = \mid \Gamma^+(x) \mid$ (resp. $\delta^-(x) = \mid \Gamma^-(x) \mid$). For two subsets $A$ and $B$ of tasks, we denote by $\Gamma^+(A, B)$ the set of all the successors of the tasks of $A$ in $B$, i.e: $\Gamma^+(A, B) = \Gamma^+(A) \cap B$ and we denote by $\delta^+(A, B)$ the cardinality of $\Gamma^+(A, B)$.

## 2. Complexity of the scheduling problems with communication delays

We are going to establish in this section the NP-completeness of: $SC(P, 4)$, $SC(P, 4, dup)$ and $SC(\bar{P}, 6)$.

### 2.1. Case of a limited number of processors

The aim of this subection is to establish the NP-completeness of $SC(P, 4)$ and $SC(P, 4, dup)$.

Let us consider the following decision problem (denoted by $SE$):

**Instance:** A (undirected) balanced bipartite graph B, with bipartition $(X, Y)$ and an integer $k$.

**Question:** Does there exist in $B$ an independent set with $k$ vertices in $X$ and $k$ vertices in $Y$? Such a subgraph is called *balanced independent set of order* $k$.

We know that $SE$ is NP-complete (see [?], where $SE$ is called "balanced bipartite complete graph problem"). The subproblem of $SE$ corresponding to the case $k = n/2$ (where $n$ is the cardinality of $X$) will be denoted $SE'$.

**Lemma.** *$SE'$ is NP-complete.*

**Proof:** Let $(B = (X, Y), k)$ be an instance of $SE$.

We construct a balanced bipartite graph $B'$ instance of $SE'$ as follows: If $k < n/2$, we add to $B$ an independent set of order $2(n-2k)$, vertex-disjoint from $B$. Thus, $B$ admits a balanced independent set of order $k$ if and only if $B'$ admits a balanced independent set of order $n/2$. □

**Theorem 1.** *$SC(P, 4)$ and $SC(P, 4, dup)$ are NP-complete.*

**Proof:** We are going to reduce $SE'$ to $SC(P, 4, dup)$. Let $B = (X, Y)$ be a balanced bipartite graph instance of $SE'$.
Consider the following bipartite precedence graph denoted by $G$:

1. $V(G) = X \cup Y \cup Z \cup U \cup V \cup W \cup \{a_1, a_2, a_3, a_4\} \cup \{b_1, b_2, b_3, b_4\}$.
2. $X \cup Y$ induces the bipartite graph $B$ oriented from $X$ to $Y$.
3. $\mid W \mid = \mid Z \mid = \mid V \mid = \mid U \mid = n$.
4. $U \cup V$ induces a perfect matching oriented from $U$ to $V$. Similarly, $V \cup Y$ induces a perfect matching oriented from $V$ to $Y$ and $W \cup Z$ induces a perfect matching oriented from $W$ to $Z$. Moreover, $a_1$ and $b_1$ precede $W$ (i.e., there are two arcs from $a_1$ and $b_1$ to every vertex of $W$), $U$ precedes $a_3$ and $b_3$, V precedes $a_4$ and $b_4$ and finally $a_i$ (resp. $b_i$) precedes $a_{i+1}$ (resp. $b_{i+1}$) for every $i < 4$.
5. $X \cup Z$ induces the complete bipartite graph of bipartion $X, Z$ oriented from $X$ to $Z$. This completes the description of the precedence graph $G$ (see figure below). Let us take $p = n + \frac{n}{2} + 2$ as the number of processors.

We claim that the so-constructed instance of $SC(P, 4, dup)$ is "schedulable" in time $t = 4$ if and only if the bipartite graph $B$ admits a balanced independent set of order $n/2$.

**Proof of the claim:** Suppose first that $B$ admits a balanced independent set of order $n/2$ denoted by $(X_1, Y_1)$, with $X_1 \subset X$, $Y_1 \subset Y$ and $\mid X_1 \mid = \mid Y_1 \mid = n/2$. Consider then the following schedule of the tasks (vertices) of $G$:

$t = 1$ : execute $U$, $X - X_1$, $a_1$ and $b_1$.

$t = 2$ : execute $V$, $X_1$, $a_2$ and $b_2$.

$t = 3$ : execute $W$, $Y_1$, $a_3$ and $b_3$.

$t = 4$ : execute $Z$, $(Y - Y_1), a_4$ and $b_4$.

It is easy to see that this schedule is realisable (the assignment of the tasks to the processors is omitted as it is easy here to find one from the schedule).

Conversely, suppose that we have a schedule of the tasks of $G$ on $p$ processors within time $t \leq 4$. Then we have the following facts:

1. No task of $G$ is duplicated: Otherwise, the number of tasks would exceed $4p$ and so we would have $t > 4$) and at each time, $p$ tasks are processed (no idle time is allowed for such a schedule).

2. Every task of $X$ is executed either at $t = 1$ or at $t = 2$: Otherwise, there would be at least one task from $Z$ that could not be executed before $t = 5$. We denote by $X_1$ and $X_2$ respectively, the sets of tasks of $X$ executed at time $t = 1$ and $t = 2$.

3. No task of $Y$ can be executed before time $t = 3$: Since every task $y$ of $Y$ is at distance two from a task of $U$, it can not be executed before time $t = 3$. We denote by $Y_3$ and $Y_4$ respectively the sets of $Y$ executed at time $t = 3$ and $t = 4$. Clearly, $X_2 \cup Y_3$ forms an independent set (as each task $y$ of $Y_3$ is preceded by a task of $V$ which cannot be executed before t=2, then due to communication delays, $y$ cannot be preceded by a task of $X_2$).

4. For every $i \leq 4$, $a_i$ and $b_i$ are executed at time $t = i$:
Because the $a_i$'s (resp. the $b_i$'s) form critical paths (that is, paths of length 3 here) and, as a consequence, all the tasks of $X$ are executed at time $t = 1$. Hence, $| X_1 | \leq p-(n+2) = n/2$ and $| X_2 | \geq n/2$.

5. Every task $w$ of $W$ must be executed at time $t = 3$:
Indeed, it cannot be executed earlier due to the communication delays between the processors assigned to the tasks $w$, $a_1$ and $b_1$, and it cannot be executed later as it precedes a task from $Z$). Therefore all tasks in $Z$ are executed in time 4. This leaves $n/2$ slots for time 4. On the other hand, all the tasks of $V$ must be executed at time $t = 2$ (otherwise, $a_4$ cannot be scheduled before $t = 5$).

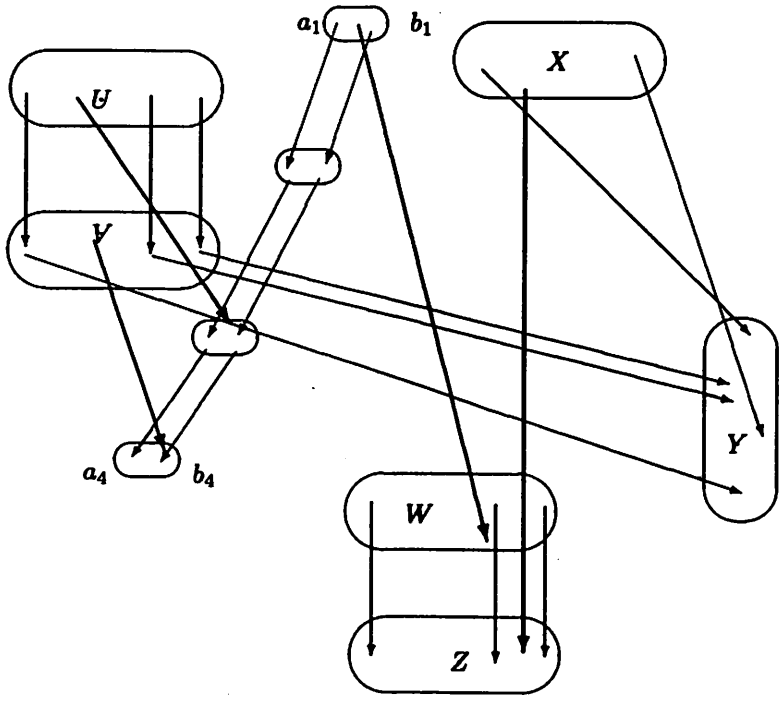


**Figure 1:** The precendence graph $G$.
Bold lines denote the occurrence of all arcs between two sets of vertices.

Thus, the $n/2$ slots are filled by half of the tasks in $Y$, that is, $| Y_3 | = n/2$.

Thus, there exists in $B$ a balanced independent set of order $n/2$ given by any balanced subset of $X_2 \cup Y_3$ of cardinality $n/2$. Moreover, Fact (1) above allows us to conclude that $SC(P, 4)$ is also NP-complete. □

As a consequence, $SC(P)$ has no polynomial $\epsilon$-approximation with $\epsilon < 5/4$ unless P=NP. On the other hand, picouleau has shown that $SC(P, 3)$ is polynomial.

## 2.2. Case of an unlimited number of processors

The remaining part of this section is devoted to scheduling with an unlimited number of processors. C. Picouleau has shown that $SC(\bar{P}, 8)$ is NP-complete. We prove that it remains NP-complete for $d = 6$ even if the precedence graph is constrained to be of fan-in 4 and depth 4 (the fan-in of

$G$ is the maximum in-degree of a vertex in $G$ and the depth is the number of levels minus one) and it becomes polynomial when $d < 6$.

**Theorem 2.** *$SC(\bar{P}, 6)$ is NP-complete.*

**Proof:** We are going to reduce 3-SAT to $SC(\bar{P}, 6)$. Let $f$ be a formula instance of 3-SAT. Let us label the litterals by $x_1, x_2, ..., x_n$. Represent each litteral $x_i$ by three vertices $i, x_i, \bar{x}_i$ and put an arc from $i$ to $x_i$ and from $i$ to $\bar{x}_i$. Represent each clause $c$ by the component $H_c$ of figure 2 which is the directed graph constructed as follows:

1. $V(H_c) = A \cup B \cup B' \cup C \cup R \cup R' \cup R"$; each of $A, B, B', C$ is of cardinality three and is labelled from 1 to 3. For example, the first element of $A$ is labelled $a_1$, the second element of $B'$ is labelled $b'_2$ ...etc.

2. For every $1 \leq i \leq 3$, join $r_i$ to $r'_i$ by an arc and join $r'_i$ to $r"_i$ by an arc and finally join $r"_i$ to $c_i$ by an arc.

3. For every $1 \leq i \leq 3$, join $a_i$ to $b_i$ (resp. $b'_i$) by an arc. Join $b_i$ to $c_{i+1}$ by an arc (the subcripts are taken here modulo 3) and $b'_i$ to $c_{i-1}$ by an arc.

Notice that, if we consider $H_c$ as a precedence relation on tasks, then all its tasks can be scheduled within time $t = 5$ but then only one task $c$ from $C$ at most is scheduled at time $t = 4$ (moreover, $c$ can be chosen to be any of the three tasks of $C$). This describes $H_c$.

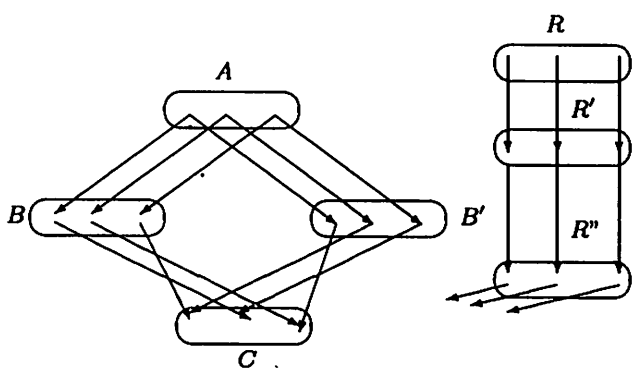


**Figure 2:** Representation of $H_c$

For every clause $c$ of $f$, if variable $x_i$ is the first (resp. second, third) variable appearing in $c$, then join the vertex corresponding to $\bar{x}_i$ to the first (resp. second, third) vertex of the subset $C$ of $H_c$.

1. For every clause $c$ construct the component $H'_c$ which is the graph of Figure 3.

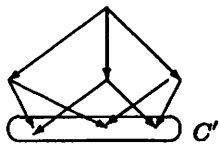


**Figure 3:** The graph $H'_c$

Observe here that if we consider $H'_c$ as a precedence relation on tasks, then all its tasks can be scheduled within time $t = 5$ but then only two tasks from $C'$ (which is of cardinality 3) at most are scheduled at time $t = 4$ (moreover, these two tasks can be chosen to be any pair from the three tasks of $C'$).

6) For every clause $c$ construct a set $D$ of three vertices and for every $1 \leq i \leq 3$, join $c_i$ (resp.$c'_i$) of $C \subset V(H_c)$ (resp. $C' \subset V(H'_c)$) to $d_i$ of $D$. This completes the description of our instance of $SC(\bar{P}, 6)$ (see next figure in which the formula $C_1.C_2$ is represented, where $C_1 = x_1 + \bar{x}_2 + x_3$ and $C_2 = \bar{x}_1 + x_2 + x_3$).

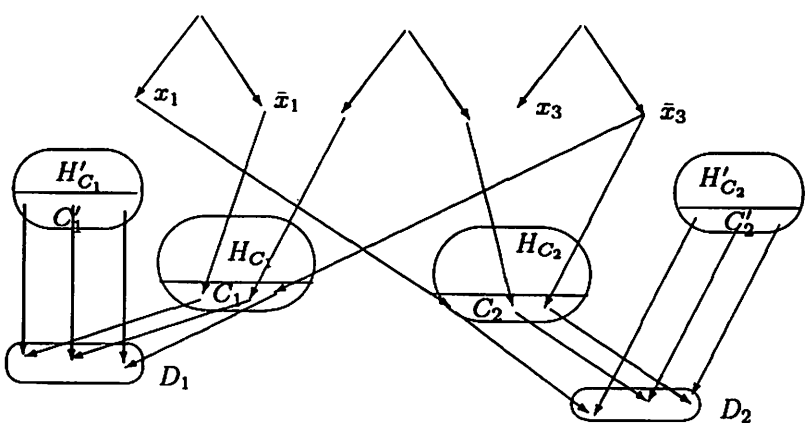


**Figure 4.**

We claim that the instance of $SC(\bar{P}, 6)$ so associated to $f$ admits the scheduling time 6 if and only if $f$ is satisfiable.

**Proof of our claim::** Suppose that $f$ is satisfiable. To every clause $c$, let us associate a variable $x_c$ satisfying $f$ and consider the task $c_v$ of $C \subset H_c$ joined to $\bar{x}_c$. Then we have the following schedule $\tau$:

At $t = 1$: execute all the tasks $i$ preceding the variables $x_i$, $\bar{x}_i$ (for every $i \leq n$) and all the tasks of $A \cup R \subset H_c$ for every clause $c$.

At time $t = 2$: execute all the false variables. At the same time, execute for every $c$, $b_{v-1}$ and $b_v$ (of $B \subset H_c$) and $b'_{v+1}$ (of $B' \subset H_c$). Execute also all the tasks of $R' \subset H_c$ for every $c$.

At time $t = 3$: execute all the true variables and all the remaining tasks of $B \cup B' \cup R'' \subset H_c$ for every $c$.

At time $t = 4$: execute $c_v$ for every clause $c$. On the other hand, for every clause $c$, consider a schedule of $H'_c$ within time $t = 5$ such that $c'_{v+1}$ and $c'_{v-1}$ are scheduled at time $t = 4$.

At time $t = 6$: execute $D_c$ for every $c$.

Clearly, $\tau$ is a schedule within time $t = 6$ of our instance because of the mentionned simple properties of the $H_c$'s and the $H'_c$'s. Now conversely, suppose that our instance admits a schedule within time 6. Set all the variables scheduled at time $t \geq 3$ to "true". Then $f$ is satisfied by such an assignment. Indeed, as for every clause $c$, $C' \subset H'_c$ has at least one task that is executed at time $t = 5$, $C \subset H_c$ must contain a task scheduled at time $t = 4$. Let us call it $c_v$ and suppose that $\bar{x}_i$ the variable linked to $c_v$ in our construction (so that $x_i$ appears in the clause $c$). Then $\bar{x}_i$ must have been executed at time $t = 2$ and consequently $c$ is true for our assignment, which proves the claim and the theorem. □

As a consequence, we see that $SC(\bar{P})$ does not admit a polynomial $\epsilon$-approximation with $\epsilon < 7/6$, unless P=NP. We conclude this section by the following theorem:

**Theorem 3.** *$SC(\bar{P}, 5)$ is polynomial.*

**Proof:** We describe here a simple reduction of $SC(\bar{P}, 5)$ to a maximum flow problem in a capacitated graph. Let $G$ be a directed acyclic graph instance of $SC(\bar{P}, 5)$. Let us call here a feasible schedule of the tasks of $G$ every schedule within time 5 of $G$. We say that a schedule is "active" if at any time, no processor is idle while it can execute some task. Clearly, if $G$ admits a feasible schedule, then it admits a feasible active schedule.

First determine the levels $L_1, L_2, ...L_k$ of the precedence graph. If $k > 5$ then clearly, the answer is "no", a schedule within time 5 of the tasks of $G$ is not possible. So, let us suppose, without loss of generality, that $k = 5$ ($L_5$ may be empty in all the arguments below). A task $x$ of $L_i$, for $i \leq k$, is said to be "critical" if for every schedule within time 5, $x$ is necessarily executed at time $i$. For every given feasible schedule of $G$ and for every $i \leq k$, a task $x \in L_i$ is said to be "late" if its execution time $t(x)$ is larger

than or equal to $i+1$. We say that a subset of tasks $B$ satisfies property $\pi$ if:

1. For every $i$, every task $b$ of $B$ in level $i > 1$ has only one predecessor $a$ in level $i-1$ and $a$ is in $B$.

2. Every task $b$ of $B$ in level $i$ has at most one successor in $B$ at level $i+1$.

Observe that every subset of critical tasks satisfies property $\pi$ since an unlimited number of processors is assumed. Consider now the subset $U$ of $L_3$ such that: $x \in U$ if and only if $x \in \Gamma^{-2}(L_5) \cap L_3$ or $x$ has at least two sucessors in $L_4$. Clearly, $U$ is a critical subset of $L_3$. Similarly, $L_5$ and $\Gamma^-(L_5)$ are critical subsets of tasks. To simplify the writing, let us put: $U' := \Gamma^-(U) \cap L_2$ and $U" := \Gamma^{-2}(U)$. From the preceding observation, $U'$ and $U"$ are also critical subsets hence the whole set $C = L_5 \cup \Gamma^-(L_5) \cup U \cup U' \cup U"$ is a critical set and satisfies therefore property $\pi$. We shall suppose in the sequel that $C$ satisfies property $\pi$ (otherwise, there is no feasible schedule for $G$).

Let us consider $W$ to be the set of tasks: $W = \{x \in L_2,\ \delta^-(x) \geq 2\} \cup \Gamma^+(U", L_2)$. For every feasible schedule of $G$, every task of $\Gamma^+(L_3 - U)$ has at most one late predecessor in $L_3 - U$; on the other hand, every task $x$ of $L_1 - U"$ has $\delta^+(x, L_2 - U' - W) - 1$ late successors because the processor to which the task $s$ has been assigned at time $t = 1$ can only execute one task from $\Gamma^+(x, L_2 - U' - W)$ at time $t = 2$ and our schedule is active. Let us then add a source $x_0$ and a sink $y_0$ to our graph $G$ so that:

1. To every vertex $x$ of $L_1 - U"$, $x_0$ is linked to $x$ by the arc $x_0x$ of capacity $\delta^+(x, L_2 - U' - W) - 1$ and to every vertex $y$ of $W$, $x_0$ is linked to $y$ by the arc $x_0y$ of capacity 1.

2. To every vertex $z$ of $\Gamma^+(L_3 - U)$, $y_0$ is linked to $z$ by the arc $zy_0$ of capacity 1.

3. All the arcs of $G$ have capacity 1.

Then $G$ admits a feasible schedule if and only if the so constructed capacitated graph $G'$ admits an $x_0y_0$-flow saturating the capacities of all the arcs $x_0x$ and $x_0y$. The "if" part of our claim is clear from the arguments above. To see the "only if" part, observe that, for a given flow on $G'$, the meaning of the flow on a given arc $xx'$ of $G$ being 1 (resp 0) is that if $x$ is executed at some time $t$ on a processor $p$, then $x'$ is executed at time $t+2$ (resp. $t+1$) on a different processor (resp. the same processor). This defines without ambiguity a feasible schedule when our condition is satisfied. □

## 3. Conclusions

Communication delays make scheduling problems difficult even for aproximation. In this paper, it is shown that even if the acyclic graph describing the precedence relation is bipartite of depth 4 and fan-in 4 and an unlimited number of processors is assumed, finding a schedule within time 6 is NP-complete. Hence no polynomial approximation ratio less than 7/6 can be found for $SC$ unless $P = NP$. In fact, by modifying appropriately the components $H_c$ in the above proof of Theorem 2, the same result can be established for acyclic graphs with depth 3 and fan-in 2 while it is polynomial to decide if a schedule with completion time 5 is possible as shown in Theorem 3. In the case with a polynmial number of processors, it is even NP-complete to decide if there exists a schedule within time 4; hence no polynomial 5/4-approximation ratio can be found here unless $P = NP$.


## Acknowledgements

I wish to thank Pr. Jean Claude Koenig and the unknown referee for helpful comments on writing.


## References


[1] Colin et al.*CPM Scheduling with small communication delays and task duplication* Op. Res., pp. 680-684, 1990.

[2] Hwang et al.*Scheduling precedence graphs in systems with interprocessor communication times* SIAM J. on Computing, vol 18 pp. 244-257, 1989.

[3] M.R. Garey and D.S. Johnson, *Computers and Intractability, a guide to the theory of NP-completness*, ed. Freeman, 1979.

[4] C. Picouleau, Ordonnancement avec coûts de communication en temps constant Rapport interne, Université Paris VI, MASI, 1992 (unpublished).

[5] V.J. Rayward Smith *UET scheduling with Unit Interprocessor Communication Delays,* Discr.App.Math., 18, 5571,1987.