RAIRO-Oper. Res. 47 (2013) 73–87
DOI: 10.1051/ro/2013028

RAIRO Operations Research
www.rairo-ro.org

SCHEDULING AN INTERVAL ORDERED PRECEDENCE GRAPH WITH COMMUNICATION DELAYS AND A LIMITED NUMBER OF PROCESSORS

Alix Munier Kordon[1], Fadi Kacem[2],
Benoît Dupont de Dinechin[3] and Lucian Finta[4]

Abstract. We consider the scheduling of an interval order precedence graph of unit execution time tasks with communication delays, release dates and deadlines. Tasks must be executed by a set of processors partitioned into K classes; each task requires one processor from a fixed class. The aim of this paper is to study the extension of the Leung–Palem–Pnueli (in short LPP) algorithm to this problem. The main result is to prove that the LPP algorithm can be extended to dedicated processors and monotone communication delays. It is also proved that the problem is NP–complete for two dedicated processors if communication delays are non monotone. Lastly, we show that list scheduling algorithm cannot provide a solution for identical processors.

Keywords. Approximation and complexity, combinatorial optimization, scheduling.

Mathematics Subject Classification. 90B35.

Received September 1, 2009. Accepted January 24, 2013.

[1] LIP6 – Université Pierre et Marie Curie, 4 place Jussieu, 75252 Paris Cedex 05, France. alix.munier@lip6.fr

[2] Laboratoire IBISC, 523 place des terrasses, 91000 Evry, France

[3] Kalray, 445 rue Lavoisier, 38330 Montbonnot Saint Martin, France

[4] LIPN, Laboratoire d'Informatique de l'Université Paris-Nord, 99 avenue Jean-Baptiste Clment, 93430 Villetaneuse, France

1. Introduction

Minimizing the makespan for an interval ordered precedence graph $G = (T, A)$ on a limited number of processors and unit execution time tasks was intensively studied. To the best of our knowledge, this class of graphs was introduced by Papadimitriou and Yannakakis in [10] who developed a polynomial time algorithm for m identical processors. More recently, Jansen [7] extended this result to typed tasks systems: processors are partitioned into K classes $M_1, \ldots, M_K$ and each task $i \in T$ must be executed by a processor from a fixed class $m_i \in \{M_1, \ldots, M_K\}$. Note that typed tasks systems include both identical processors (only one class of processors) and dedicated processors (only one processor per class).

In computer science applications, tasks may model instructions executed on processors and precedence constraints correspond to data transfers through processor registers. For any precedence constraint between two tasks i and j, a minimum additional delay $\ell_{ij} \geq 0$ between the end of i and the beginning of j is usually considered. If t_i and t_j denotes respectively the starting times of tasks i and j, the corresponding constraint is $t_i + p_i + \ell_{ij} \leq t_j$ where p_i is the processing time of the task i.

Several studies were devoted to interval ordered precedence graphs in this context: each task is associated to an interval, and there exists a precedence relation (i, j) if the interval of i ends before the begining of the one of j. An interval ordered precedence graph is said monotone if delays ℓ_{ij} are monotonically increasing with the distance between the intervals corresponding to the tasks. This natural limitation of delays was first introduced by Palem and Simons in [9], who presented a polynomial–time algorithm to solve the determination of a feasible schedule for a monotone interval ordered precedence graph with deadlines and m identical processors. Their main idea is to use *backward scheduling* in order to strengthen the deadlines, which are then used as priorities in a list scheduling algorithm. Leung *et al.* [8] further developed this idea for solving this problem with additional release dates. Lastly, Dupont de Dinechin [3] considered a similar approach to solve the determination of a feasible schedule for a monotone interval ordered precedence graph with release dates, deadlines and typed tasks system.

Another way to model data transfers is to conseider a communication delay only when two adjacent tasks i and j are executed on two different processors. More formally, if $c_{ij} \in \mathbb{N}$ denotes the minimum communication delay between tasks i and j, the precedence constraint is $t_i + p_i + x_{ij}c_{ij} \leq t_j$ with x_{ij} a boolean variable equal to 0 if and only if i and j are performed by the same processor. Surveys on scheduling problems with communication delays can be found in [2, 11].

Communications delays are said large if they are possibly greater than execution times. Scheduling problems under this assumption are known to be very difficult to be solved exactly. For instance, for unit execution time tasks and delay equal to $c \geq 3$, Giroudeau *et al.* in [4] proved that the existence of a schedule of length smaller that $c + 4$ is NP-complete for a general precedence graph with no resource limitation. Note that monotone communication delays include constant delays.

In this context, most of the authors limit their studies to unit execution time tasks and unit communication delays. Ali and Rewini [1] developed a polynomial–time algorithm for the minimization of the makespan for an interval ordered precedence graph, m identical processors and unit communication delays. Verriet [13] developed a polynomial algorithm to solve the determination of a feasible schedule for an interval ordered precedence graph with release dates, deadlines, unit communication delays and m identical processors. Verriet developed another polynomial algorithm [12] to solve the determination of a feasible schedule for an interval ordered precedence graph with communication delays on a typed tasks system.

We consider the determination of a feasible schedule with large communications delays for an interval ordered precedence graphs with release dates, deadlines for a typed tasks system. Problem and notations are presented in Section 2. We prove in Section 3 that the problem is NP–complete in the strong sense for 2 dedicated processors and general communication delays. In the rest of the paper, communication delays are supposed to be monotone. Section 4 describes the extension of the algorithm presented in [8] by Leung, Palem and Pnueli (in short LPP algorithm) to interval ordered precedence graphs with release dates, deadlines, monotone communication delays and dedicated processors. The deadline modification scheme presented in [8] and proofs are also simplified. In Section 5, we show with an example that for m identical processors and large communication delays, there is no optimal list-based scheduling algorithm, so the LPP algorithm cannot be extended to solve this problem.

2. Model formulation

2.1. General problem

Let T be a set of n unit execution time tasks with arbitrary release dates $r_i \in \mathbb{N}$ and deadlines $d_i \in \mathbb{N}$, $i \in T$. As mentioned above, our study is limited to an interval ordered precedence graph $G = (T, A)$: each task $i \in T$ can be associated to a non empty interval $I_i =]a_i, b_i[$ with $(a_i, b_i) \in \mathbb{R}^2$, $a_i < b_i$. The arc $(i, j) \in A$ if $b_i \leq a_j$. Integer communication delays $c_{ij} \in \mathbb{N}$ are associated with every arc $(i, j) \in A$.

Each class of machines M_p, $p \in \{1, \dots, K\}$ may be composed by τ_p identical processors. A schedule is a couple of vector $\sigma = (t, \pi)$ where t_i, $i \in T$ denotes start time of i and $\pi_i \in \{1, \dots, \tau_{m_i}\}$ denotes the machine of type m_i that executes i. A schedule is feasible if:

1. Release dates and deadlines are fulfilled, *i.e.*$\forall i \in T$, $r_i \leq t_i < d_i$.
2. Precedence relations are also fulfilled, *i.e.*$\forall (i, j) \in A$, set $x_{ij} = 0$ if $m_i = m_j$ and $\pi_i = \pi_j$, $x_{ij} = 1$ otherwise. We get $t_i + 1 + x_{ij}c_{ij} \leq t_j$.
3. At each time t, there is at most one task executed by each machine.

The general problem considered here is building a feasible schedule for an interval ordered precedence graph with communication delays, release dates, deadlines and typed tasks system.

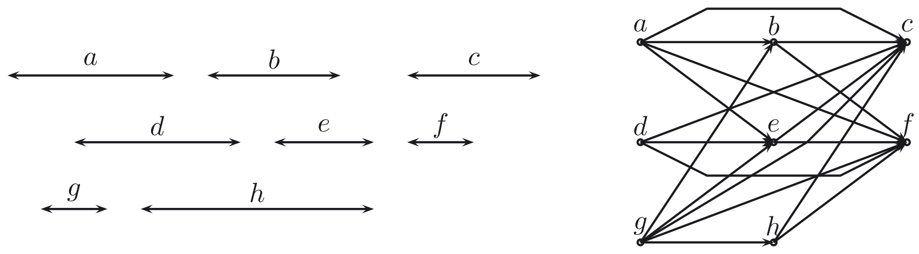


Figure 1. An interval ordered graph and the corresponding intervals.

Table 1. Monotone communication delays for the interval ordered precedence graph pictured by Figure 1.

c_{ij}	b	c	e	f	h
a	1	3	2	3	$\star$
b	$\star$	2	$\star$	2	$\star$
d	$\star$	2	1	2	$\star$
e	$\star$	1	$\star$	1	$\star$
g	2	3	2	3	2
h	$\star$	1	$\star$	1	$\star$

The precedence relation as defined before is transitive: if (i,j) and (j,k) are both in A, then (i,k) also belongs to A. Moreover, transitive arcs must not be removed, since any value c_{ik} can be larger than $1 + c_{ij} + c_{jk}$.

2.2. Monotone communication delays

For any task $i \in T$, $\Gamma^+(i)$ (*resp.*$\Gamma^-(i)$) denotes the immediate successors (*resp.*predecessors) of i in G. Property 2.1 comes from [10]:

Property 2.1. *Let $G = (T, A)$ be an interval ordered precedence graph. For any couple of tasks $(i,j) \in T^2$, $\Gamma^-(i) \subseteq \Gamma^-(j)$ or $\Gamma^-(j) \subseteq \Gamma^-(i)$.*

Definition 2.2. Let $G = (T, A)$ be an interval ordered precedence graph. Communication delays associated with G are monotone if

$$\forall((i,j),(i,k)) \in A^2, \Gamma^-(j) \subseteq \Gamma^-(k) \Rightarrow c_{ij} \leq c_{ik}.$$

$\Gamma^-(j) \subseteq \Gamma^-(k)$ is equivalent to $a_j \leq a_k$, and thus $a_j - b_i \leq a_k - b_i$. Roughly speaking, c_{ij} is increasing with the distance between the two disjunctive intervals I_i and I_j.

As example, let consider the set of intervals and the corresponding interval ordered graph pictured by Figure 1. Table 1 presents monotone communication delays. Release dates, deadlines and machines type are reported in Table 2.

Table 2. Release date, deadline and machine type for tasks in example of Figure 1.

$i \in T$	a	b	c	d	e	f	g	h
r_i	0	1	2	0	4	1	0	1
d_i	3	3	7	3	6	6	3	6
m_i	2	1	1	1	2	2	1	2

2.3. List schedules

A priority list is a bijection $L : T \to \{1, \ldots, n\}$. It is usually built using the precedence graph structure, release dates, deadlines, and resource constraints. The determination of a list schedule using an extension of the LPP algorithm [8] will be discussed in Section 4.

Let us suppose here that L is fixed. A list schedule with (large) communication delays may be built as described in [6]: consider task $i \in T$ having it's start time t_i not yet fixed at current time t. Task i is said to be ready to be executed at time t by processor π of type m_i if

1. $t \geq r_i$;
2. Every task $j \in \Gamma^-(i)$ was scheduled before t. Moreover, if j is not performed by π, the communication delay between j and i is fulfilled and allows i to be performed at time t;
3. π is idle at time t.

The goal of the algorithm is to find for every couple (t, π), $\pi \in \bigcup_{p=1}^{K} M_p$ the ready task i with the highest priority (*i.e.*with value $L(i)$ minimal among ready tasks) and set $t_i = t$ and $\pi_i = \pi$ if i exists. It starts with time $t = 0$ and increases t as soon as there is no more ready task or idle slot at time t.

One may observe that if time increment step equals 1, the previous algorithm may have a complexity equal to $\mathcal{O}(n^2 \max_{(i,j)\in A} c_{ij})$, which is not polynomial. However, it is proved in [6] that values of t may be limited to get an algorithm of time complexity following $\mathcal{O}(n^3)$.

3. Complexity for general communication delay

We prove here that the scheduling problem of a (general) communication delay interval ordered precedence graph and 2 dedicated processors is NP–complete in the strong sense. This result justifies the limitation of our study to monotone communication delays. For that purpose, let us consider Sequencing Bipartite Graph with Communication Delays (SBC) defined as follows:

SCHEDULING BIPARTITE GRAPH WITH COMM. DELAYS (SBC)
Input: Let $A = \{1, \ldots, k\}$ and $B = \{k+1, \ldots, 2k\}$, $k \in \mathbb{N}^\star$ two sets of k unit execution time tasks, $\forall (i,j) \in A \times B$, a delay $c_{ij} \in \mathbb{N}$, $\forall i \in A \cup B$, $m_i \in \{1, 2\}$ and an integer $D \geq 0$.

Question: is there a vector t_i, $i \in A \cup B$ such that, $\forall (i,j) \in (A \cup B)^2$, if $m_i = m_j$ then $t_i \neq t_j$, $\max_{i \in A \cup B} t_i < D$ and $\forall (i,j) \in A \times B$, $x_{ij} = 0$ if $m_i = m_j$, $x_{ij} = 1$ otherwise and $t_i + x_{ij} c_{ij} + 1 \leq t_j$?

Sequencing Couples with Latencies (SCL) defined as follows was prove to be NP–complete in the strong sense by Yu *et al.* [14]. In the following, we will show that SBC is NP-complete in the strong sense by reducing SCL to SBC.

SEQUENCING COUPLES WITH LATENCIES (SCL)
Input: Let $A = \{1, \ldots, k\}$ and $B = \{k+1, \ldots, 2k\}$, $k \in \mathbb{N}^\star$ be two sets of k unit execution time tasks, k values $\ell_i \in \mathbb{N}$, $i \in \{1, \ldots, k\}$ and an integer $D \geq 0$.
Question: is there a vector t_i, $i \in A \cup B$ such that, $\forall (i,j) \in (A \cup B)^2$, $t_i \neq t_j$, $\max_{i \in A \cup B} t_i < D$ and $\forall i \in \{1, \ldots, k\}$, $t_i + 1 + \ell_i \leq t_{i+k}$?

Theorem 3.1. *There exists a polynomial transformation from* SCL *to* SBC.

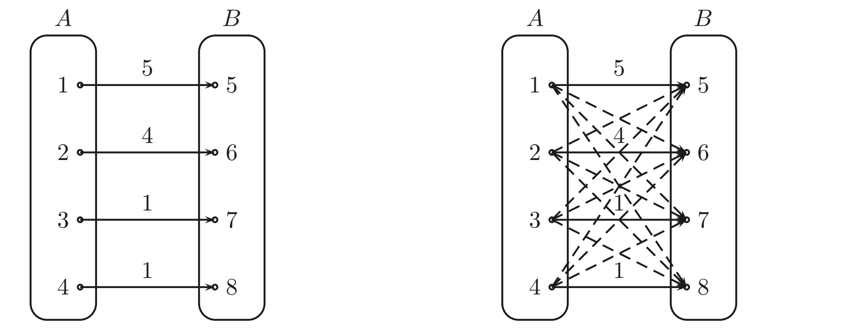
Proof. Let Π be an instance of SCL. An instance $f(\Pi)$ of SBC is built by setting, $\forall (i,j) \in A \times B$, $c_{ij} = \ell_i$ if $j = k + i$, $c_{ij} = 0$ otherwise. Moreover, $\forall i \in A$, $m_i = 1$ and $\forall i \in B$, $m_i = 2$. Figure 2 illustrates this transformation.

1. Let t_i, $i \in A \cup B$ be a solution to Π. We show that all tasks from A are completed without interruption in the time interval $[0, k]$:
    - Let suppose that there exists two successive tasks i and j with $i \in B$ and $j \in A$. Then, because of precedence relations, they can be exchanged without any influence on the other tasks. Thus, we can suppose that tasks from A are all performed before tasks from B.
    - Tasks from A have no predecessor. Thus they can be performed without any interruption.

    Thus t_i, $i \in A \cup B$ is also a solution to $f(\Pi)$.
2. Conversely, let us assume that t_i, $i \in A \cup B$ is a solution to $f(\Pi)$. By construction of the graph, the first task from B starts after the completion time of the last tasks from A. Thus, for every couple $(i,j) \in (A \cup B)^2$, $t_i \neq t_j$ and t_i, $i \in A \cup B$ is also a feasible schedule for Π. □

Now, a complete bipartite graph is clearly an interval order such that every task from A (*resp.*from B) is associated with an interval $I =]a, b[$ (*resp.*$I' =]a', b'[$) with $b \leq a'$. Moreover, SBC is clearly in NP. Since SCL is strongly NP–complete [14], we get the following corollary:

Corollary 3.2. *The existence of a schedule for the problem with due dates for a (general) communication delay interval ordered precedence graph and* 2 *dedicated processors is NP–complete in the strong sense.*



Instance Π of SEQUENCING COUPLES WITH LATENCIES

Instance $f(\Pi)$ of SCHEDULING BIPARTITE GRAPH WITH COMMUNICATION DELAYS

FIGURE 2. Transformation from SEQUENCING COUPLES WITH LATENCIES to SCHEDULING BIPARTITE GRAPH WITH COMMUNICATION DELAYS.

4. EXTENDED LPP ALGORITHM FOR DEDICATED PROCESSORS AND MONOTONE COMMUNICATION DELAYS

The aim of this section is to present an extension of the LPP algorithm [8] for an interval ordered precedence graph with monotone communication delays and dedicated processors. We first present a simple polynomial algorithm to compute an upper bound of the completion time of a task based on Jackson's rule, a process called *backward scheduling* by Leung et *et al.* [8]. Backward scheduling is applied iteratively in next subsection to modify the deadline of every task until a fix-point. It is lastly proved that, using a list algorithm based on an "earliest deadline first" priority list, modified deadlines lead to a feasible solution if it exists.

4.1. AN UPPER BOUND OF THE COMPLETION TIME OF A TASK

Let $k \in T$ and $S_k = \Gamma^+(k) \cup indep(k)$, where $indep(k)$ is the set of tasks from T having no precedence relation with k. Let us suppose that every task ℓ from $S_k \cup \{k\}$ has a release date r_ℓ and a deadline d_ℓ. Moreover, since there is exactly one machine per type (dedicated processors), we can set for every arc $(i, j) \in A$, $x_{ij} = 0$ if $m_i = m_j$, $x_{ij} = 1$ otherwise.

Let $t \in \{r_k + 1, \ldots, d_k\}$. Then, we consider the decision problem Existence$_k(t, r, d)$ defined as follows. Roughly speaking, it fixes the schedule of task k at instant $t-1$ and it returns true if a feasible schedule exists for tasks in $S_k \cup \{k\}$ (considering release dates, deadlines, resource constraints and precedence relations between k and its immediate successors).

TABLE 3. Effective delays for the interval ordered precedence graph pictured by Figure 1 considering the allocation of the tasks to 2 dedicated processors expressed by Table 2.

$x_{ij}c_{ij}$	b	c	e	f	h
a	1	3	0	0	$\star$
b	$\star$	0	$\star$	2	$\star$
d	$\star$	0	1	2	$\star$
e	$\star$	1	$\star$	0	$\star$
g	0	0	2	3	2
h	$\star$	1	$\star$	0	$\star$

TABLE 4. Release dates, deadlines and feasible start times for the decision problem Existence$_e(5,\, r,\, d)$ with answer true.

$i \in T'$	b	c	e	f	h
r_i	1	6	4	5	1
d_i	3	7	5	6	6
t_i	1	6	4	5	1

Existence$_k(t,\, r,\, d)$

Input: Set of tasks $S_k \cup \{k\}$ of unit execution time with deadlines $d'_k = t$ and $\forall j \in S_k$, $d'_j = d_j$. Every task $j \in S_k \cup \{k\}$ has to be executed by m_j. Release dates of tasks are computed as follows:

1. $r'_k = t - 1$; $\forall j \in \mathit{indep}(k), r'_j = r_j$;
2. $\forall j \in \Gamma^+(k)$, $r'_j = \max(r_j, t + x_{kj}c_{kj})$.

Question: Is there a schedule of tasks from $S_k \cup \{k\}$ meeting release dates r', deadlines d' and resource constraints?

Let consider the example defined previously by Figure 1, Table 1 and Table 2. The delays $x_{ij}c_{ij}$ for any arc $(i, j) \in A$ are reported by Table 3.

For this example, $S_e = \{c, f\} \cup \{b, h\}$. Thus Existence$_e(5,\, r,\, d)$ asks the question of the existence of a schedule for tasks $T' = \{b, c, e, f, h\}$ with releases dates, deadlines and resource constraints presented in Table 4. The answer to the question is here clearly positive.

If Existence$_k(t,\, r,\, d)$ is false, then k cannot be performed at time $t-1$ without violating a precedence or a resource constraint of the initial problem. This decision problem can be solved easily by an $\mathcal{O}(n \log n)$ algorithm using Jackson's rule (*i.e.*tasks are sorted following their deadline and are scheduled by a priority list algorithm using earliest deadline first) [5].

For any task $k \in T$, Existence can be used to tighten the upper bound of the completion time of k considering precedence and resources constraints. Indeed, the maximum value $t^\star$ in $\{r_k + 1, \ldots, d_k\}$ such that Existence$_k(t^\star,\, r,\, d)$ is true is an upper bound of the completion time of k. Let us denote by BS$_k(r,\, d)$ an algorithm that computes $t^\star$ if it exists. This algorithm is referred as the backward schedule algorithm, or BS algorithm in short.

For any fixed task $k \in T$, BS algorithm is based on successive calls of Existence$_k$ until $t^\star$ is reached. As explained in [3,8], two binary searches in the time interval $[r_k + 1, d_k]$ are needed, leading to an algorithm of time complexity bounded by $\mathcal{O}(\log(d_k - r_k) \times n \log n)$.

The following lemma describes an important property on BS algorithm:

Lemma 4.1. *Let d^1 and d^2 be two elements from $\mathbb{N}^n$ with $d^1 \leq d^2$ (*i.e.*$\forall i \in T, d^1_i \leq d^2_i$). For any vector $r \in \mathbb{N}^n$ such as $r \leq d^1$ and any task $i \in T$, $BS_i(r, d^1) \leq BS_i(r, d^2)$.*

Proof. For every integer value $t \in \{r_i + 1, \ldots, d^1_i\}$, if Existence$_i(t, r, d^1)$ is true, then Existence$_i(t, r, d^2)$ is. Now, as BS is a maximization algorithm, BS$_i(r, d^1) \leq$BS$_i(r, d^2)$, the result. $\square$

4.2. Deadline modification algorithm

The idea here is to modify deadlines of all the tasks by computing successively upper bounds of the completion times following Algorithm 1.

If a call of BS has no solution, Algorithm 1 stops. In the following, we suppose that $d^\star$ can be computed. For any couple $(i, k) \in \{1, \ldots, n\}^2$, let us note by $d^\star_k(i)$ the modified deadline of task k at the end of the ith iteration of the loop **for**. We also set $d^\star_k = d^\star_k(n)$ at the end of Algorithm 1.

Algorithm 1 Computation of the modified deadlines

```
Require: A precedence graph $G$ with communication delays, release dates $r$, deadlines
  $d$ and $K$ dedicated processors.
Ensure: Modified release dates $r^\star$ and deadlines $d^\star$.
  Compute $\forall i \in \{1, \ldots, n\}$, $r^\star_i = \max_{j \in \Gamma^-(i)}(r_i, r^\star_j + 1 + x_{ji}c_{ji})$ following a topological
  order;
  Renumber tasks such as $r^\star_1 \geq r^\star_2 \geq \ldots \geq r^\star_n$;
  Compute $\forall i \in \{1, \ldots, n\}$, $d^\star_i = \min_{j \in \Gamma^+(i)}(d_i, d^\star_j - 1 - x_{ij}c_{ij})$ following an inverse
  topological order;
  for $i = 1$ to $n$ do
    $d^\star_i = \mathrm{BS}_i(r^\star, d^\star)$;
    Compute $\forall i \in \{1, \ldots, n\}$, $d^\star_i = \min_{j \in \Gamma^+(i)}(d_i, d^\star_j - 1 - x_{ij}c_{ij})$ following an inverse
    topological order;
  end for
```

For instance, let's consider the execution of Algorithm 1 on the instance of the scheduling problem presented by Figure 1, Table 1 and Table 2. Table 5 shows the vector $r^\star$ computed by the algorithm and a possible associated renumbering of tasks. Values $d^\star_k(i)$, $(i, k) \in \{1, \ldots, n\}^2$ are reported by Table 6 (as well as the vector $d^\star(0)$ computed before the loop for).

TABLE 5. Modified release dates $r^\star$ and a corresponding possible renumbering of tasks.

T	a	b	c	d	e	f	g	h
$r^\star$	0	2	6	0	4	5	0	3
i	8	5	1	7	3	2	6	4

TABLE 6. Modified deadlines $d^\star_k(i)$, $(i,k) \in \{1,\dots,8\}^2$.

$k \in T$	1	2	3	4	5	6	7	8
$d^\star(0)$	**7**	**6**	**5**	**5**	**3**	**2**	**3**	**1**
$d^\star(1)$	**7**	**6**	**5**	**5**	**3**	**2**	**3**	**1**
$d^\star(2)$	**7**	**6**	**5**	**5**	**3**	**2**	**3**	**1**
$d^\star(3)$	**7**	**6**	**5**	**5**	**3**	**2**	**3**	**1**
$d^\star(4)$	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**1**
$d^\star(5)$	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**1**
$d^\star(6)$	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**1**
$d^\star(7)$	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**1**
$d^\star(8)$	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**1**

Following lemma characterizes vectors $d^\star(i)$, $i \in \{1,\dots,n\}$:

Lemma 4.2. *For any task $k \in T$, the function $i \to d^\star_k(i)$ is non increasing for $i \in \{0,\dots,n\}$. Moreover, for every value $i \in \{k,\dots,n\}$, $d^\star_k(i) = d^\star_k(k) =BS_k(r^\star, d^\star(k-1))$.*

Proof. By definition of the BS algorithm, $\text{BS}_k(r^\star, d^\star(k-1))$ is in $\{r^\star_k+1,\dots,d^\star_k(k-1)\}$ and thus $\text{BS}_k(r^\star, d^\star(k-1)) \le d^\star_k(k-1)$. Moreover, adjusting values of $d^\star$ following an inverse topological order also decreases $d^\star$. Thus, $d^\star$ may only decrease and the first part of lemma is proved.

Now, by Algorithm 1, for any $k \in \{1,\dots,n\}$, $d^\star_k(k) =\text{BS}_k(r^\star, d^\star(k-1))$. Since $r^\star_n \le r^\star_{n-1} \le \dots \le r^\star_k$, there is no successor of k in $\{k+1,\dots,n\}$. Thus, modifications of $d^\star_j$, $j \in \{k+1,\dots,n\}$ have no influence on $d^\star_k$ and $d^\star_k(i) = d^\star_k(k)$ for any value $i \in \{k,\dots,n\}$. □

Lemma 4.3. *For every arc $(k,j) \in A$ and every $i \in \{0,\dots,n\}$, $d^\star_k(i) < d^\star_j(i)$.*

Proof. This lemma comes from the adjustment procedure called at each step following an inverse topological order. □

The following lemma proves that at the end of the algorithm, values $d^\star_\ell, \ell \in T$ cannot be modified again using BS algorithm:

Lemma 4.4. *If a feasible schedule exists, then at the end of Algorithm 1, $\forall k \in \{1,\dots,n\}$, $d^\star_k = BS_k(r^\star, d^\star)$.*

Proof. By contradiction, let assume that there exists $k \in \{1, \dots, n\}$ such as $d^\star_k \neq$ BS$_k(r^\star,\ d^\star)$. By Lemma 4.2, $d^\star_k = d^\star_k(k) =BS_k(r^\star,\ d^\star(k-1))$. Since deadlines are decreasing following the iterations of Algorithm 1, $d^\star(k-1) \geq d^\star$. Thus, by Lemma 4.1, $d^\star_k =$BS$_k(r^\star,\ d^\star(k-1)) >$ BS$_k(r^\star,\ d^\star)$.

As a consequence, Existence$_k(d^\star_k,\ r^\star,\ d^\star)$ is false. Let's denote by r' and d' the release dates and dealines of this call of Existence. $\forall \ell \in S_k \cup \{k\}$, let t_ℓ be the starting time of ℓ obtained by using a list schedule governed by Jackson's rule. Let us consider the first instant t such that a task $j \in S_k$ misses its deadline d'_j, thus $t = d'_j$.

Let Δ be the minimum value in $\{-1, 0, \dots, d'_j - 1\}$ such that every slot in $[\Delta+1, d'_j]$ is filled on machine m_j by a task ℓ with $d'_\ell \leq d'_j$. Let also $B = \{j\} \cup \{\ell \in S_k \cup \{k\} | m_\ell = m_j$ and $\Delta < t_\ell < d'_j\}$.

Every slot on m_j during the time interval $[\Delta+1, d'_j]$ is filled by a task from B. Moreover, j is not performed during this interval. The consequence is that $|B| = 1 + (d'_j - \Delta)$, and thus there are no sufficient slots on m_j to perform tasks from B during the interval $[\Delta+1, d'_j]$.

Moreover every task $\ell \in B$ has a priority larger than the one (if any) performed at time Δ, thus $r'_\ell \geq \Delta + 1$.

Two cases are to be considered:

1. Let suppose first that, for every task $\ell \in B - \{k\}$, $\ell < k$. The, by Lemma 4.2, $d^\star_\ell(k-1) = d^\star_\ell = d'_\ell$. The consequence is that every task from B has to be performed during the interval $[\Delta+1, d'_j]$ in Existence$_k(d^\star_k,\ r^\star,\ d^\star(k-1))$, which is impossible. Thus, $d^\star_k \neq$BS$_k(r^\star,\ d^\star(k-1))$, a contradiction.
2. Otherwise, there exists a task $\ell \in B - \{k\}$ with $\ell > k$. We first show that, for any task $i \in B$, $r^\star_\ell \geq \Delta + 1$.
   - Let suppose that $\ell \in B$ with $\ell > k$. By definition of the numbering, $r^\star_\ell \leq r^\star_k$. By definition of $r^\star$, we get that $\ell \notin \Gamma^+(k)$, so $r'_\ell = r^\star_\ell$ and $r^\star_\ell \geq \Delta + 1$ by definition of B.
   - Let consider now a task $i \in B$ with $i \leq k$, then $r^\star_i \geq r^\star_k \geq r^\star_\ell$ with $\ell \in B$ and $\ell > k$. Thus, as seen just before, $r^\star_i \geq r^\star_\ell \geq \Delta + 1$.

   Moreover, for any task $i \in B$, $d'_i = d^\star_i \leq d'_j$. The consequence is that it is impossible to schedule tasks from B with release dates $r^\star$ and deadlines $d^\star$. The scheduling problem is thus not feasible. □

4.3. Optimality of the extended LPP algorithm

The extended LPP algorithm consists in computing the modified deadlines using Algorithm 1 and in building a priority-list based schedule with a priority list L such that, for any couple $(i, j) \in T^2$, $L(i) < L(j) \Rightarrow d^\star_i \leq d^\star_j$.

As an example, the schedule obtained for the example depicted in Figure 1 with modified deadlines from Table 6 is shown in Figure 3. The next theorem shows that, in case of dedicated processors and monotone communication delays, there exists a feasible schedule if and only if the schedule built by the extended LPP algorithm is feasible.

g	d	b				c
a		h		e	f	

FIGURE 3. LPP algorithm for the example from Figure 1 and the modified deadlines from Table 6.

Theorem 4.5. *The extended version of the LPP algorithm solves polynomially the determination of a feasible schedule for a unit execution time typed tasks system with arbitrary release dates and deadlines, monotone communication delays, precedence constraints ruled by an interval ordered precedence graph and a limited number of typed processors with one processor for each type.*

Proof. By contradiction, let us assume that the extended LPP algorithm provides a schedule $\sigma = (t_1, \dots, t_n)$, which is not feasible. Let i be the earliest task that misses its modified deadline $i.e. t_i \geqslant d_i^\star$. We show that there must be an earlier task k that also misses its modified deadline.

A task $j \in T$ is said *saturated* if $m_j = m_i$ and $d_j^\star \leqslant d_i^\star$. Define $\Delta < d_i^\star$ as the latest time slot that is not filled with a saturated task on the processor of type m_i; consider the task sets $\Sigma = \{\ell \text{ saturated} : \Delta < t_\ell < d_i^\star\} \cup \{i\}$ and $\Sigma' = \{\ell \in \Sigma : r_\ell^\star \leqslant \Delta\}$.

We first prove by contradiction that $\Sigma' \neq \emptyset$ and $\Delta \geq 0$. Indeed, if $\Sigma' = \emptyset$, then $\forall \ell \in \Sigma$, $r_\ell^\star > \Delta$. Now, since Σ is composed by i and all the tasks performed by m_i in the interval $[\Delta + 1, d_i^\star]$, we get $|\Sigma| = (d_i^\star - \Delta) + 1 > 0$. Moreover, there are exactly $d_i^\star - \Delta$ slots from $\Delta + 1$ to $d_i^\star$ available on processor m_i. Thus, the scheduling problem is unfeasible. The consequence is that $\Sigma' \neq \emptyset$ and thus $\Delta \geq 0$.

Since Σ' is a finite non empty set of tasks, there exists $j \in \Sigma'$ such that $|\Gamma^-(j)|$ is minimum in Σ'. As j is not scheduled at date Δ or earlier, there must be a constraining task $k \in \Gamma^-(j)$. Note that, $\forall \ell \in \Sigma'$, $k \in \Gamma^-(\ell)$: indeed, $\forall \ell \in \Sigma'$, $|\Gamma^-(j)| \leq |\Gamma^-(\ell)|$, thus by Property 2.1, $\Gamma^-(j) \subseteq \Gamma^-(\ell)$ and then $k \in \Gamma^-(\ell)$. Lastly, after the computation of $r^\star$, $r_k^\star < r_j^\star$. As $j \in \Sigma'$, $r_j^\star \leq \Delta$ and thus $r_k^\star < \Delta$.

Task k is either of type m_i or of a type different than m_i:

Case 1. Let us assume that $m_k = m_i$. In that case, there is no communication delay from k to tasks from Σ'. Besides, task k constrains task j to be executed after date Δ, thus $t_k \geq \Delta$.

We prove that $k \notin \Sigma$: since $k \in \Gamma^-(j)$, $\Gamma^-(k) \subset \Gamma^-(j)$ and thus $k \notin \Sigma'$. As $r_k^\star < \Delta$, we get that $k \notin \Sigma$.

Now, $d_k^\star < d_j^\star \leq d_i^\star$, so k is saturated. As $k \notin \Sigma$, $t_k \leq \Delta$ and thus, $t_k = \Delta$, which is impossible by definition of Δ.

Case 2. Let us suppose that $m_k \neq m_i$. In that case, there will be an additional communication delay between k and j, $i.e. t_k + 1 + c_{kj} > \Delta$.

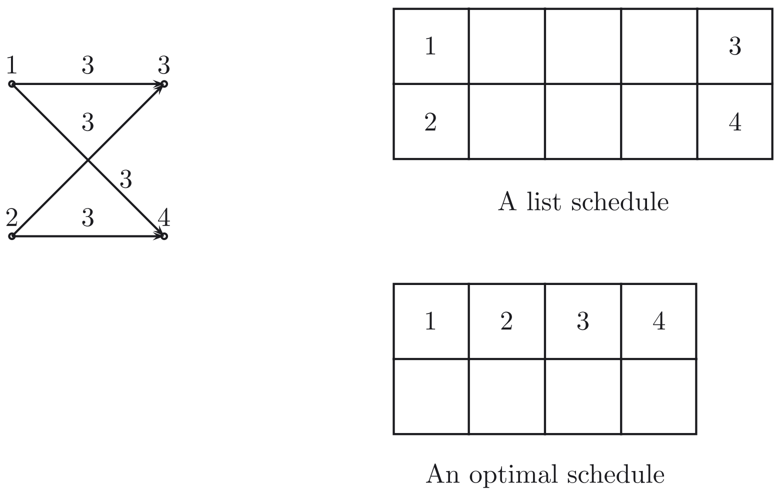


FIGURE 4. List schedules are not optimal for 2 identical processors.

Now, let us prove that $\varSigma \subset S_k$. By definition of S_k,

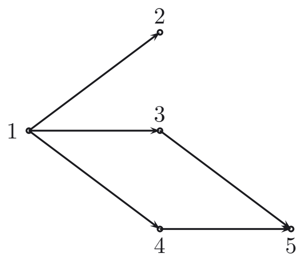
$$T = \varGamma^{-}(k) \cup \{k\} \cup \varGamma^{+}(k) \cup indep(k) = \varGamma^{-}(k) \cup \{k\} \cup S_k.$$

We show that $\varGamma^{-}(k) \cap \varSigma = \emptyset$. For every task $\ell \in \varGamma^{-}(k)$, the inequality $r^\star_\ell < r^\star_k$ holds. Since $r^\star_k < \varDelta$, we obtain $r^\star_\ell < \varDelta$. Thus, if $\ell \in \varSigma$, then $\ell \in \varSigma'$. But, if $\ell \in \varSigma'$, then by the structure of interval ordered graphs and since j is minimum, $k \in \varGamma^{-}(\ell)$ and thus there is a circuit in G, which is impossible. So, $\varSigma \subset S_k$.
Now, by Lemma 4.4, $d^\star_k = \mathrm{BS}_k(r^\star, d^\star)$ and Existence$_k(d^\star_k, r^\star, d^\star)$ is true. As $\varSigma \subset S_k$, a list scheduling algorithm using Jackson's rule computes feasible start times for the $(d^\star_i - \varDelta)+1$ tasks from $\varSigma$ before $d^\star_i$. Thus, there must be at least one task $j' \in \varSigma'$ that is processed at date $t' \leq \varDelta$ by this last algorithm. Thus, $d^\star_k + c_{kj'} \leq t' \leq \varDelta$. Now, as $\varGamma^{-}(j) \subseteq \varGamma^{-}(j')$ and communication delays are monotone, $c_{kj} \leq c_{kj'}$ and then $d^\star_k + c_{kj} \leq \varDelta$. Lastly, since $t_k + 1 + c_{kj} > \varDelta$ and $\varDelta \geq d^\star_k + c_{kj}$, we get $t_k + 1 > d^\star_k$ and thus, task k misses its deadline, a contradiction. □

5. Non optimality of list scheduling for identical processors

For m identical processors, list schedules may be never optimal, as shown by Figure 4. Thus, the extension of Leung–Palem–Pnueli to m processors for typed tasks systems is not optimal, even without initial release dates and deadlines.

However, the question remains open for unit communication delays ($i.e. c_{ij} = 1$, $\forall (i,j) \in A$). In that case, the feasibility problem is polynomial for identical



$$c_{ij} = 1, \forall (i,j) \in A$$

$$d_1 = 1$$

$$d_2 = d_3 = d_4 = 3$$

$$d_5 = 4$$

$$\forall i \in T,\, d_i^\star = d_i$$

1	2	3		5
		4		

A non feasible LPP schedule

1	4	3	5
		2	

A feasible schedule

FIGURE 5. LPP algorithm may compute a non feasible schedule (for unit communication delays).

processors [13]. However, the computation of the modified deadlines as developed previously doesn't lead to a feasible solution as illustrated by Figure 5: the modified deadline algorithm does not modify the deadlines initially fixed and the LPP algorithm may compute a non feasible schedule.

6. Conclusions

We prove that the algorithm of Leung *et al.* [8] may be extended to communication delays to compute a feasible schedule for an interval ordered precedence graph with monotone communication delays, unit execution time tasks, release dates and due dates for dedicated processors. The first open question is the extension of this work to a typed task system with unit communication delays. Another interesting open question is the computation of a feasible schedule for a non limited number of identical processors for an interval ordered precedence graph with monotone communication delays and unit execution time tasks.

References

[1] H.H. Ali and H.H. El.-Rewini, An optimal algorithm for scheduling interval ordered tasks with communication on processors. *J. Comput. Syst. Sci.* **2** (1995) 301–307.

[2] P. Chrétienne and C. Picouleau, Scheduling with communication delays : a survey. in *Scheduling Theory and its Applications,* edited P. Chrétienne, E.G. Coffman, J.K. Lenstra and Z. Liu. John Wiley Ltd (1995) 65–89.

[3] B. Dupont de Dinechin, Scheduling monotone interval orders on typed task systems. In *PLANSIG 2007, 26th Worshop of the UK Planning and Scheduling Special Interest Group* (2007) 25–31.

[4] R. Giroudeau, J.-C. Konig, F.K. Moulai and J. Palaysi, Complexity and approximation for precedence constrained scheduling problems with large communication delays. *Theor. Comput. Sci.* **401** (2008) 107–119.

[5] W. Horn, Some simple scheduling algorithms. *Naval Research Logistics Quarterly* **21** (1974) 177–185.

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

[7] K.Jansen, Analysis of Scheduling Problems with Typed Task Systems. *Discrete Appl. Math.* **52** (1994) 223–232.

[8] A. Leung, K.V. Palem and A. Pnueli, Scheduling Time-Constrained Instructions on Pipelined Processors. *ACM Transact. Program. Languages Syst.* **23** (2001) 73–103.

[9] K. Palem and B. Simons, Scheduling time-critical instructions on risc machines. *ACM Transactions on Programming Languages and Systems* **4** (1993) 632–658.

[10] C.H Papadimitriou and M. Yannakakis, Scheduling interval-ordered tasks. *SIAM J. Comput.* **8** (1979) 405–409.

[11] B. Veltman, B.J. Lageweg and J.K. Lenstra, Multiprocessor scheduling with communication delays. *Parallel Comput.* **16** (1990) 173–182.

[12] J. Verriet, The complexity of scheduling typed task systems with and without communication delays. External Report 1998-26, UU-CS (1998).

[13] J. Verriet, Scheduling interval-ordered tasks with non-uniform deadlines subject to non-zero communication delays. *Parallel Comput.* **25** (1999) 3–21.

[14] W. Yu, H. Hoogeveen and J.K. Lenstra, Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard. *J. Scheduling* **7** (2004) 333–348.