# Scheduling Tasks with Communication Delays on Parallel Processors

*JORGE URRUTIA AND NEJIB ZAGUIA*
*Department of Computer Science*
*University of Ottawa*
*Ottawa, Ontario, CANADA*

ABSTRACT

Let $J_n=\{v_1,\ldots,v_n\}$ be a set of $n$ jobs to be executed and $E$ a set of precedence constraints on $J_n$. Assume that we have available a set $M_k=\{m_1,\ldots,m_k\}$ of $k$ identical machines that are to execute the jobs in $J_n$ such that the time needed by any machine to execute each job in $J_n$ is one unit.

Our main result in this paper is to give an $O(n \log^2(n))$ algorithm to find an optimal scheduling *with communication delays* on sets $J_n$ of tasks for which their precedence constraints induce a tree order on $J_n$, i.e. an order $P(J_n,<)$ on $J_n$ such that the covering graph of $P(J_n,<)$ is a tree. A communication delay is the time it takes for some information to be transferred between two different machines $m_i$ and $m_j$ of $M_k$, say from $m_i$ to $m_j$ before $m_j$ can start processing a certain job.

## 1. Introduction

Let $J_n=\{v_1,\ldots,v_n\}$ be a set of $n$ jobs to be executed and $E$ a set of precedence constraints on $J_n$. Assume that we have available a set $M_k=\{m_1,\ldots,m_k\}$ of $k$ identical machines that are to execute the job in $J_n$ and that the time needed by any machine to execute each job in $J_n$ is one unit of time. Given $J_n$ and $M_k$, a scheduling $f$ of $J_n$ is a function $f:J_n \rightarrow M_k x N$ that assigns to each job $v_i \in T$ a machine $m(v_i)$ and a completion time $c(v_i)\geq 1$ such that the following conditions are satisfied:

i) no two jobs are scheduled in the same machine at the same time

ii) if there is a precedence relation in $E$ that dictates that $v_i$ has to be executed before $v_j$, then the completion time $c(v_i)$ of $v_i$ is smaller than the completion time $c(v_j)$ of $v_j$.

In this paper we study schedulings in which another restriction called *communication delays* are considered. Suppose that two jobs $v_s$ and $v_t$ are such that $v_s$ has to be completed before $v_t$. If $v_s$ and $v_t$ are executed in different machines, say $m_i$ and $m_j$ respectively, some information has to be passed from $m_i$ to $m_j$. The time it takes to transmit that information from $m_i$ to $m_j$ is called a communication delay.

Accordingly, we say that a scheduling f is a scheduling with communication delays if in addition to i) and ii) it also satisfies:

iii) if $v_i$ has to be executed before $v_j$ and $m(v_i) \neq m(v_j)$ then $c(v_j) \geq c(v_i)+2$

The set of precedence relations E on the elements of $T_n$ induces a partial order $P(J_n,<)$ on the elements of $T_n$ in which a job $v_i$ is smaller than a job $v_j$ if there is a precedence constraint that dictates that job $v_i$ has to be executed before job $v_j$; we will denote this by $v_i < v_j$.

A precedence constraint $v_i < v_k$ in $P(J_n,<)$ is redundant if there is a job $v_j$ such that $v_i < v_j$ and $v_j < v_k$. A precedence constraint $v_i < v_k$ is essential if there is no job $v_j$ such that $v_i < v_j$ and $v_j < v_k$. The covering graph of $P(J_n, <)$ is the graph with vertex set $J_n$ in which $v_i$ is connected to $v_j$ if $v_i < v_j$ is an essential precedence constraint of $P(J_n,<)$.

It is known that scheduling with communication delays is NP-complete for sets of tasks with arbitrary precedence relations, even for the case when we use two processors [6]. Nevertheless, there are some special cases for which polynomial time scheduling algorithms exist [1,6].

Our main result in this paper is to give an $O(n \log^2(n))$ algorithm to find an optimal scheduling *with communication delays* on partial orders $P(J_n,<)$ such that the covering graph of $P(J_n,<)$ is a tree using an arbitrary number of machines.

## 2. Terminology and Definitions

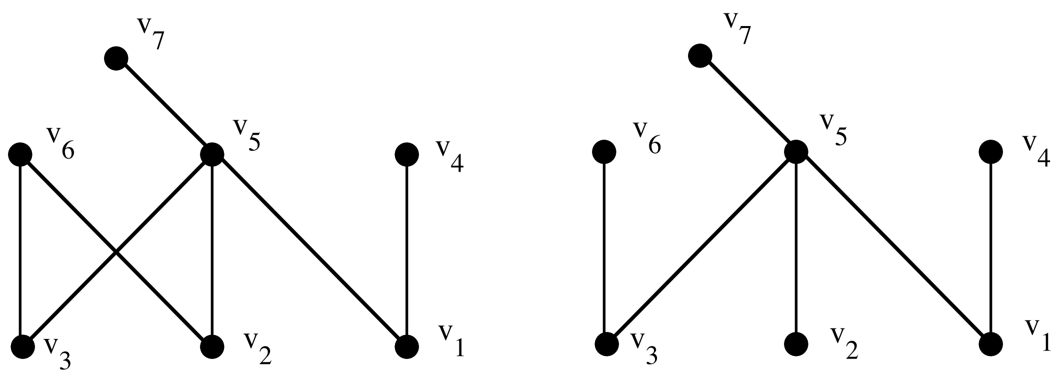
An *ordered set* $P(J_n,<)$ on a set $J_n$ of n elements consists of a binary relation < over the set $J_n$ that satisfies:

(a) For any $v_i, v_j, v_k \in J_n$ such that $v_i < v_j$ and $v_j < v_k$ we have $v_i < v_k$ (transitivity), and

(b) $v_i \not< v_i$ (antisymmetry).

Given two elements $v_i, v_j \in J_n$, we say that $v_i$ is a *lower cove* of $v_j$ if $v_i < v_j$ and there is no element $v_k$ of $J_n$ different from $v_i$ and $v_j$ such that $v_i < v_k < v_j$. The *covering graph* of $P(J_n,<)$ is the graph with vertex set $J_n$ in which two vertices $v_i$ and $v_j$ are adjacent if $v_i$ is a lower cover of $v_j$ or, $v_j$ is a lower cover of $v_i$.

It is customary to represent an ordered set $P(J_n,<)$ on the plane by drawing its covering graph on the plane in such a way that the elements of $P(J_n,<)$ are represented by small circles in such a way that if $v_i$ is lower cover of $v_j$ then $v_i$ is joined to $v_j$ with a strictly monotonically increasing curve from $v_i$ to $v_j$. See Figure 1.




Notice that $v_1$ is smaller than $v_7$ since $v_1 < v_5$ and $v_5 < v_7$.

(a)

A tree ordered set

(b)

Figure 1

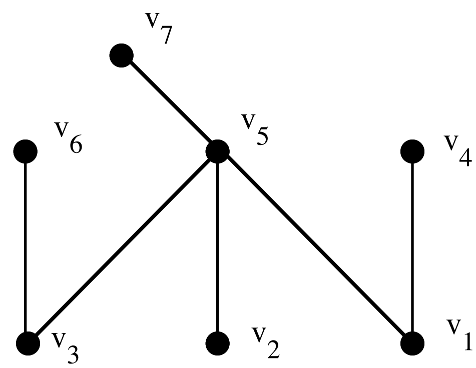
An equivalent formulation of iii), and one that will be more useful to us is the following:

iii') If a task $v_i$ is scheduled with termination time $c(v_i)$, then at most one task $v_j > v_i$ can be scheduled with completion time $c(v_i)+1$ and at most one task $v_k < v_i$ can be scheduled with completion time $c(v_i)-1$.

To see that iii) and iii') are equivalent, simply notice that by iii) if a task $v_i$ is completed at time $c(v_i)$, then the machine $m_s$ that executed $v_i$ can complete only one job $v_j > v_i$ at time $c(v_i)+1$. Any other job $v_k > v_i$ executed by a machine $m_t$ different from $m_s$ is delayed by at least one extra unit due to a communication delay. Similarly, at most one job $v_j < v_i$ can be completed at time $c(v_i)-1$.

Given a scheduling f with communication delays of an ordered set $P(J_n,<)$ we define the completion time $C(f)$ of f as the largest completion time in f over all of the elements of $J_n$. A scheduling f is optimal if its completion time $C(f)$ is the smallest possible over all possible schedulings of $P(J_n,<)$. $C(f)$ will be called the *optimal scheduling time* of $P(J_n,<)$.

| | | | |
|---|---|---|---|
| $m(v_1)=m_1$ | $c(v_1)=1$ | $m(v_5)=m_2$ | $c(v_5)=3$ |
| $m(v_2)=m_2$ | $c(v_2)=1$ | $m(v_6)=m_3$ | $c(v_6)=2$ |
| $m(v_3)=m_3$ | $c(v_3)=1$ | $m(v_7)=m_2$ | $c(v_7)=4$ |
| $m(v_4)=m_1$ | $c(v_4)=2$ | | |




An optimal scheduling of a tree order using 3 machines

$m_1$ , $m_2$ and $m_3$ .

Figure 2

In Figure 2, we show an optimal scheduling with delays for a tree. Notice that by x) $v_5$ cannot be assigned completion time 2. To see this, notice that by iii') at most one of $v_1$, $v_2$ and $v_3$ can have completion time $c(v_5)-1$, and since $c(v_i)\geq 1$, $i=1,2,3$ the completion time of $v_5$ is at least 3. Also, it is possible to assign to $v_7$ completion time $c(v_7) = c(v_5) + 1$ since the same machine $m_2$ executes both jobs.

## 3. Finding Optimal Schedulings with Communication Delays for Tree-Orders

In this section we will develop a $O(n \log^2(n))$ algorithm to find optimal schedulings with communication delays for tree orders. To avoid carrying cumbersome notation, we shall refer to optimal schedulings of trees, not tree orders.

It is easy to see that for the case when the number of machines available is at least the number of jobs to be performed, the most important parameter to take into consideration is the completion time. All we have to keep in mind is that, under our assumptions, if $v_i < v_j$ and $c(v_j)=c(v_i)+1$, then by iii'), both of these tasks are processed by the same machine. Otherwise we may assume that $v_i$ and $v_j$ are processed by different machines. In view of our previous discussion and to ease the presentation of our results, in what follows, we will concentrate only in the completion times of our tasks and ignore the machines assigned to our tasks.

Our main objective in this section is to prove the following theorem:

**Theorem 1:** Optimal scheduling with communication delays in tree orders $P(J_n,<)$ with n elements using at most n machines can be found in $O(n \log^2(n))$ time.

In order to prove our result we need to introduce some concepts:

A scheduling f for $P(J_n,<)$ is called an □ schedule if $1 \leq c(v_i) \leq$ □ for all the elements of $J_n$. Let $v_i$ be an element of $J_n$ and suppose that we have an □-schedule of $P(J_n,<)$. We define Min(□, $P(J_n,<), v_i$) to be the earliest completion time that can be assigned to $v_i$ over all □-schedules of $P(J_n,<)$. Similarly we can define Max(□, $P(J_n,<), v_i$). In Figure 2(a) we have a tree order $P(J_{10},<)$ with a 6-schedule. In Figure 2(b) we have a different 6-schedule of $P(J_{10},<)$ in which Max(6, $P(J_{10},<)$, v) is achieved and in Figure 2(c) Min(6, $P(J_{10},<)$, v) is achieved.

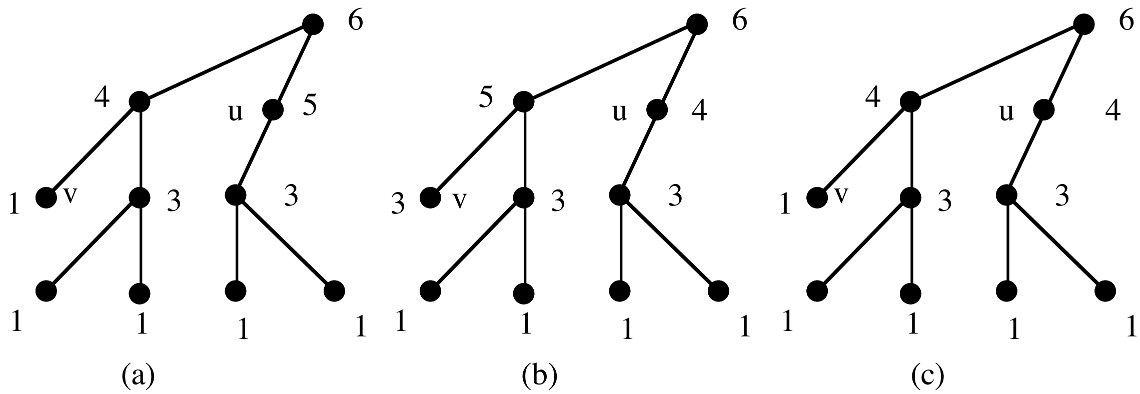



Figure 3

We will prove that given an □-schedule of a tree order $P(J_n,<)$ and an element $v_i$ of $J_n$ we can develop two procedures FMIN(□, $P(J_n,<),v_i$) and FMAX(□, $P(J_n,<),v_i$) that in linear time obtain new □-schedules for $P(J_n,<)$ such that the completion time of $v_i$ is Min(□, $P(J_n,<), v_i$) or Max(□, $P(J_n,<), v_i$) respectively.

Some results will be needed to develop our procedures.

**Lemma 1**: Let $P(J_n,<)$ be a tree order with an □-scheduling and $v_i$ be an element of P. Then there is an □-schedule of $P(J_n,<)$ in which the completion time of $v_i$ is Min(□, $P(J_n,<), v_i$) (respectively Max(□, $P(J_n,<), v_i$)) such that the termination time of any lower (resp. upper) covers $v_j$ of $v_i$ is Min(□, $P(J_n,<), v_j$) (resp. Max(□, $P(J_n,<), v_j$)).

**Proof**: Let us consider an □-schedule f of $P(J_n,<)$ in which the completion time of $v_i$ is Min(□, $P(J_n,<), v_i$). Suppose that there is a lower cover $v_j$ of $v_i$ that has a completion time greater than Min(□, $P(J_n,<), v_j$). Consider a different □-schedule f' of $P(J_n,<)$ such that the completion time of $v_j$ in f' is Min($\square$, P($J_n$,<), $v_j$). Consider the covering graph T of P($J_n$,<). Since P($J_n$,<) is a tree order, T is a tree. When we delete from T the edge connecting $v_j$ to $v_i$, we split T into two subtrees, one, $T_j$ containing $v_j$ and the other $T_i$ containing $v_i$. Let $P_j(S_j$,<) be the tree order induced in P($J_n$,<) by the set of all the vertices of $T_j$. Let f'' be the schedule obtained from f and f' as follows: If an element $v_k$ of $J_n$ is not a vertex of $T_j$ assign to $v_k$ the completion time it had in f; if $v_k$ is a vertex of $T_i$ then assign to $v_k$ the completion time it has in f'. It is easy to see that f'' is a valid $\square$-schedule for P($J_n$,<). A similar argument can be applied to Max($\square$, P($J_n$,<), $v_j$) for the case when $v_j$ is an upper cover of $v_i$ and we seek to maximize the completion time of $v_i$ in an $\square$-schedule of P($J_n$,<).

QED

**Lemma 2**: Let $v_i$ be an element of $J_n$ and suppose we have an $\square$-schedule f of P($J_n$,< such that:

a) For all the lower covers of $v_j$ of $v_i$ the schedule assigned to $v_j$ by f is Min($\square$, P($J_n$,<), $v_j$)

and

b) If $v_j$ lower cover of $v_i$ and $u_s$ is an upper cover of $v_j$ different from $v_i$ the completion time of $u_s$ is Max($\square$, P($J_n$,<), $u_s$).

Then Min($\square$, P($J_n$,<), $v_i$) can be calculated as follows:

Let $\square$={Min($\square$, P($J_n$,<), $v_j$): $v_j$ is a lower cover of $v_i$}. Then if there are at least two lower covers $v_j$ and $v_k$ of $v_i$ such that Min($\square$, P($J_n$,<), $v_j$)=Min($\square$, P($J_n$,<), $v_k$)=$\square$ then the earliest completion time Min($\square$, P($J_n$,<), $v_i$) of $v_i$ is $\square$ + 2.

**Proof:** Suppose then that there is only one lower cover $v_j$ of $v_i$ such that $\square$=Min($\square$, P($J_n$,<), $v_j$), i.e. for any lower cover $v_k$ of $v_i$ different from $v_j$ we have Min($\square$, P($J_n$,<), $v_k$) < $\square$. Then if the completion times Max($\square$, P($J_n$,<), $u_s$) in f of all the upper covers of $v_j$ are at least $\square$ + 2, then Min($\square$, P($J_n$,<), $v_i$)=$\square$ +1, otherwise Min($\square$, P($J_n$,<), $v_i$)=$\square$ +2. To prove this, all we need to do is to notice that if the completion time Max($\square$, P($J_n$,<), $u_k$) in f of one upper cover of $v_j$ is exactly $\square$ + 1, then the machine that executes $v_j$ is the same that executes $u_k$ in f, and thus the earliest completion time Min($\square$, P($J_n$,<), $v_i$) is $\square$ + 2, otherwise we can assign to $v_i$ completion time $\square$ + 1, which is clearly the earliest completion time for $v_i$ over all $\square$-schedules of P($J_n$,<).

A similar argument holds for maximizing the completion time of an element $v_i$ of P($J_n$,<). One more definition will be needed before we can proceed to give two recursive procedures that given an □-schedule f of a tree order $P(J_n,<)$ will enable us to calculate $Min(□, P(J_n,<), v_i)$ and $Max(□, P(J_n,<), v_j)$.

QED

Since $P(J_n,<)$ is a tree order, the covering graph T of $P(J_n,<)$ is a tree. Let $e_{i,j}=v_i\text{-}v_j$ be an edge of T. Then $T\text{-}e_{i,j}$ consists of two subtrees $T_i$ and $T_j$ of T such that $v_i$ is a vertex of $T_i$ and $v_j$ is a vertex of $T_j$. Further let $S_i=\{v_k□J_n$: $v_k$ is a vertex of $T_i\}$ and $S_j=\{v_k□J_n$: $v_k$ is a vertex of $T_j\}$. We now define $P(S_i,<)$ and $P(S_j,<)$ as the suborders of $P(J_n,<)$ induced by $S_i$ and $S_j$ respectively (see Figure 4). Clearly the covering graphs of $P(S_i,<)$ and $P(S_j,<)$ are $T_i$ and $T_j$ respectively. We can now describe FMIN and FMAX.

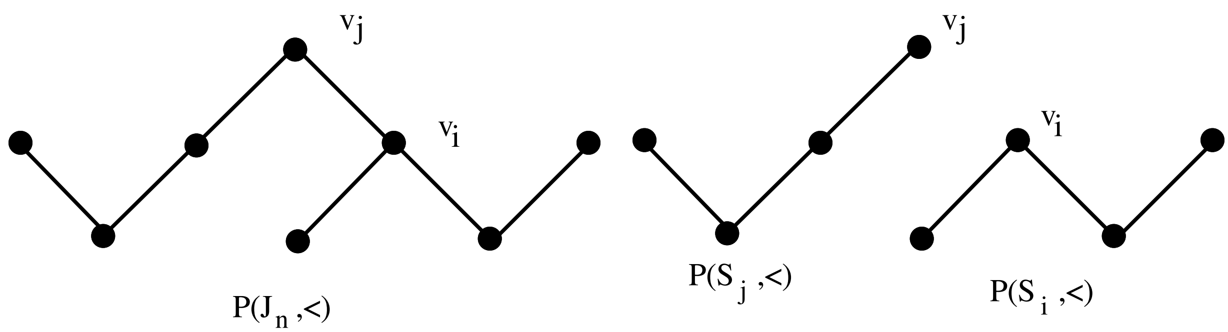



Figure 4

**Procedure FMIN**(□, $P(J_n,<), v_i$)

If v is a minimal element of $P(J_n,<)$ then

$Min(□, P(J_n,<), v_i)=1$ and $c(v)=1$

Else

For each lower cover $v_j$ of $v_i$ in $P(J_n,<)$ calculate $□_j=Min(□, P_j(S_j,<), v_j)$ using **FMIN**(□, $P_j(S_j,<), v_j$).

Let □=Max{$□_j$: $v_j$ is a lower cover of $v_i$}

If at least two $□_j$ achieve the maximum value □ in $\{□_1,...,□_m\}$ then

$Min(□, P(J_n,<), v_1)$ =□+2, c(v)=□+2

else

Let $v_j$ be the unique lower cover of $v_i$ such that □=$□_j$=Min(□, $P_j(S_j,<), v_j$).

For each upper cover $v_k$ of $v_j$ calculate $□_k$=Max(□, $P_k(S_k,<), v_k\}$ using **FMAX**(□, $P_k(S_k,<), u_j$)

if □≥□+2 then

$Min(□, P(J_n,<), v_i)$=□+1, c(v)=□+1

else

Min(□, P($J_n$,<), $v_i$)=□+2

End if.

**EndFMIN.**

**Procedure FMAX**(□, P($J_n$,<), $v_i$)

If v is a maximal element of P($J_n$,<) then

Max(□, P($J_n$,<), $v_i$)=□ and c(v)=□

Else

For each upper cover $v_j$ of $v_i$ in P($J_n$,<) calculate □$_j$=Max(□, $P_j$($S_j$,<), $v_j$) using **FMAX**(□, $P_j$($S_j$,<), $v_j$).

Let □=Min{□$_j$: $v_j$ is an upper cover of $v_i$}

If at least two □$_j$ achieve the minimum value □ in {□$_1$,...,□$_m$} then

Max(□, P($J_n$,<), $v_1$) =□-2, c(v)=□-2

else

Let $v_j$ be the unique upper cover of $v_i$ such that □=□$_j$=Max(□, $P_j$($S_j$,<), $v_j$).

For each lower cover $v_k$ of $v_j$ calculate □=Min(□, $P_k$($S_k$,<), $v_k$}

using **FMIN**(□, $P_k$($S_k$,<), $u_j$)

if □≤□-2 then

Max(□, P($J_n$,<), $v_i$)=□11, c(v)=□-1

else

Min(□, P($J_n$,<), $v_i$)=□-2

End if.

**EndFMAX.**

**Complexity analysis and correctness of the procedures**

We first prove the following lemma:

**Lemma 3**: Given an □□scheduling of P($J_n$,<) and a vertex $v_i$ of P($J_n$,<), procedures **FMIN**(□, P($J_n$,<),□$v_i$) and **FMAX**(□, P($J_n$,<), $v_i$) correctly minimize (maximize) the completion time c($v_i$) of a vertex $v_i$ over all □-schedulings of P($J_n$,<).

**Proof**: Our proof is by induction on the number of elements in $J_n$ for both of **FMIN**(□, P($J_n$,<), $v_i$) and **FMAX**(□, P($J_n$,<), $v_i$).

If $J_n$ has one element, then our procedures work correctly.

Let us assume that *both* of **FMIN**(□, $P(J_n,<), v_i$) and **FMAX**(□, $P(J_n,<), v_i$) work correctly for tree orders $P(J_n,<)$ with less than n vertices, $n\geq 2$ and let us prove that they work for tree orders with n vertices.

Let $P(J_n,<)$ be a tree with n elements, $n\geq 2$, and let $v_i$ be an element of $P(J_n,<)$. Let us consider a □-schedule f' of $P(J_n,<)$ for which the completion time $c'(v_i)$ of $v_i$ in f' is Min(□, $P(J_n,<), v_i$).

Consider any □□schedule f of $P(J_n,<)$. We shall prove that if we apply **FMIN**(□,$P(J_n,<), v_i$) to f, the completion time $c(v_i)$ assigned to $v_i$ by **MIN**(□,$P(J_n,<), v_i$) is smaller than or equal to $c'(v_i)$. Let $v_j$ be a lower cover of $v_i$ in $P(J_n,<)$ and $c(v_j)$ be the completion time of $v_j$ in f. Clearly f induces a valid □-schedule in $P(S_j,<)$ and since $P(S_j,<)$ has less elements that $P(J_n,<)$ when during the execution of **FMIN**(□,$P(J_n,<), v_i$) a recursive call is made to **FMIN**(□, $P(S_j,<), v_j$), the completion time $□_j$ assigned to $v_j$ is Min(□, $P(S_j,<), v_j$) over all possible □-schedules of $P(S_j,<)$.

We now show that Min(□, $P(S_j,<), v_j$) as calculated here is exactly Min(□, $P(J_n,<), v_j$). To show this, consider a scheduling f'' of $P(J_n,<)$ such that the completion time $c''(v_j)$ of $v_j$ is exactly Min(□, $P(S_j,<), v_j$). By the same argument as before, the completion time assigned to $v_j$ when we apply **FMIN**(□, $P(S_j,<), v_j$) to f'' is the smallest over all possible □-schedules of $P(S_j,<)$, i.e. the completion time assigned ti $v_j$ has to be $□_j$. Thus $□_j \leq c''(v_j)$ which by assumption is Min(□, $P(J_n,<), v_j$). Our claim now follows.

Using similar arguments, we can now prove that if $v_k$ is an upper cover of $v_j$ different from $v_i$ then if we apply **FMAX**(□,$P(S_k,<), v_k$) to the subtree order of $P(S_k,<)$ obtained when we delete from the covering graph T of $P(J_n,<)$ the edge $v_j$-$v_k$ the completion time assigned to $v_k$ by **FMAX**(□,$P(S_k,<), v_k$) is exactly Max(□, $P(J_n,<), v_k$). It now follows by Lemma 2 that the completion time assigned to $v_i$ by **FMIN**(□, $P(J_n,<), v_i$) is exactly Min(□, $P(J_n,<), v_i$). A dual argument applies now for the procedure **FMAX**(□,$P(S_k,<), v_k$).

QED

We now prove:

**Lemma 4. FMIN**(□, $P(J_n,<), v_i$) and **FMAX**(□, $P(J_n,<), v_i$) work in O(n) time, where n is the number of elements of $J_n$.

**Proof**: Let g(□, P($J_n$,<), $v_i$) be the number of steps **FMIN**(□, P($J_n$,<), $v_i$) takes (**FMAX**(□, P($J_n$,<), $v_i$). To execute **FMIN**(□, P($J_n$,<), $v_i$), we first calculate for all the lower covers $v_j$ of $v_i$ □$_i$=Min(□, $P_j(S_j)$, $v_j$) using **FMIN**(□, P($J_n$,<), $v_i$). After this is done, Min(□, P($v_i$), $v_i$) is found in O(k) time using Lemma 2, where k is the number of lower covers of $v_i$. Then:

$$(\square, T, v) = O(k) + \sum_{i=1}^{k} \square(\square, \square_i, \square_i)$$

which is O(E), where E is the set of edges of T. However T is a tree, and thus O(E) is linear which proves our result.

QED

We now develop a recursive divide and conquer algorithm to find optimal schedulings in tree orders in O(n $\log^2$n). To this end we need the following result known as the one-third two-thirds theorem:

**Theorem 2**: Let T be any tree. Then there is a vertex v of T such that all of the components of T-v have at most 2n/3 vertices. Moreover, v can be found in linear time.

Consider a vertex $v_i$ of a tree order P($J_n$,<), with lower covers say {$v_j$,...,$v_k$} and upper covers {$v_s$,...,$v_t$}. Let $P_{low}(v_i)$ and $P_{upp}(v_i)$ be the subtrees of P($J_n$,<) containing $v_i$ obtained by deleting all the edges joining $v_i$ to its upper covers (resp. lower covers) (see Figure 5).

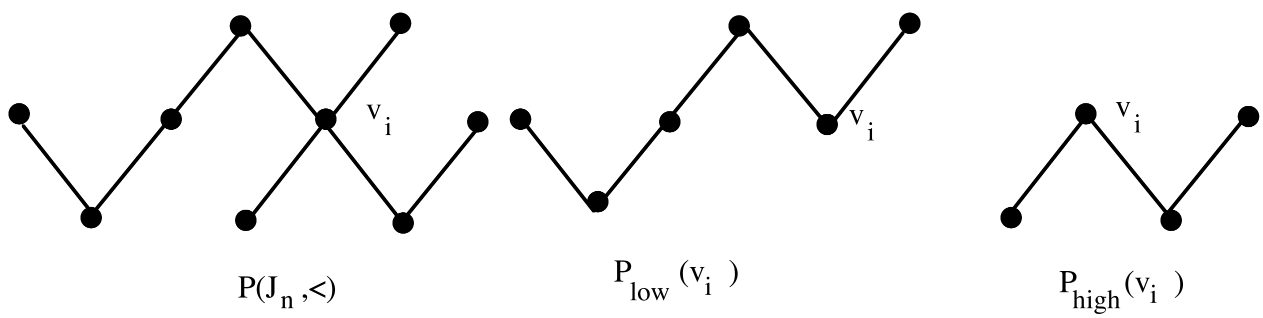



Figure 5

Given a tree order T our algorithm to find an optimal schedule for a tree order T follows the outline:

**Algorithm OPTTREE**

1) Find a vertex $v_i$ of $P(J_n,<)$ as in Theorem 2.

2) For every $v_j$ lower or upper cover of $v_i$ obtain an optimal schedulings for $P(S_j,<)$, with completion time $\square_j$.

3) Using these optimal schedulings, obtain optimal schedulings for $P_{low}(v_i)$ and $P_{upp}(v_i)$

4) Merge the optimal schedulings for for $P_{low}(v_i)$ and $P_{upp}(v_i)$ into an optimal scheduling for $P(J_n,<)$.

Due to the choice of $v_i$, if we can show how to achieve Step 3 in linear time and Step 4 in O(n log n) time, it immediately follows that our algorithm OPTREE runs in $O(n \log^2 n)$ time.

We now prove that Step 3 can be achieved in linear time.

**Lemma 5**: Suppose that for every lower (upper) cover $v_j$ of $v_i$ we have an optimal scheduling for $P(S_j,<)$ with completion time $\square_j$. Then optimal schedulings for $P_{low}(v_i)$ (resp. $P_{upp}(v_i)$) can be obtained in linear time.

**Proof**: Suppose that the lower covers of $v_i$ are $v_k,...,v_m$. Let $\square=\max\{\square_k,...,\square_m\}$. Clearly the optimal completion time for time $P_{low}(v_i)$ can not be smaller than $\square$. For each lower cover $v_j$ of $v_i$ in $P(J_n,<)$ find a $\square$-schedule for $P(S_j,<)$ which minimizes the completion time $\square_j$ of $v_j$. This can be achieved in linear time using MIN($\square$, $P(S_j)$, $v_j$). Let $\square'=\mathrm{Max}\{\square_k,...,\square_m\}$. If $\square'\leq\square-2$ then we can assign to v termination time $\square$, thus achieving an $\square$-scheduling of $P_{low}(v_i)$ which has optimal termination time.

If $\square'=\square-1$, two cases arise:

1) $\square'$ is achieved by at least two elements of $\{\square_k,...,\square_m\}$. Then by condition iii) the completion time $c(v_i)$ of $v_i$ has to be $\square'+2=\square+1$.

2) $\square'$ is achieved by exactly one element in $\{\square_k,...,\square_m\}$, say $\square_k$. If $v_k$ has an upper cover, say $w_1$ in $P(S_k,<)$ then the completion time for $w_1$ in $P(S_k,<)$ is $\square$. Thus by iii) we cannot assign completion times of $\square$ for both $w_1$ and $v_i$. Thus $v_i$ must be assigned completion time $\square+1$. On

the other if $u_1$ is a maximal element of $P(S_k,<)$ then we can assign completion time □ to $v_i$ .

If □=□ and it is achieved by at least two elements in $\{$□$_1,...,$□$_k\}$ then by iii) $v_i$ must be assigned termination time □+2. Otherwise, we can assign to v a completion time of □+1.

Clearly the scheduling obtained for $P_{low}(v_i)$ is optimal and the whole process takes linear time as claimed. A dual argument proves our result for $P_{upp}(v)_i$.

QED.

We now show that Step 4 can be achieved in O(n log(n)) time.

**Lemma 6**: Suppose that we have optimal schedulings for $P_{low}(v_i)$ and $P_{upp}(v_i)$ with completion times □ and □ respectively. Then we can obtain an optimal schedule for $P(J_n,<)$ in O(n log n).

**Proof**: Let □=max{□,□}. The first thing to notice is that the optimal completion time of $P(J_n,<)$ could be any integer in the range □ to □=□+□. Our problem is now to find the smallest integer □ in the range □ to □ for which an □-schedule for $P(J_n,<)$ exists. This can be accomplished by a performing a binary search for □ in the range □=min to max=□. At each step of our search, we check if a □-schedule for T exists, for □=$\frac{\text{min+max}}{2}$ . If a □-schedule exists for T, we make max=□, otherwise min=□. Since □+□ is at most n, the number of iterations of our search is logarithmic.

We now proceed to prove that for any integer □ in the range □ to □+□ we can test in linear time if an □-scheduling for $P(J_n,<)$ exists.

Let g and h be optimal schedulings for $P_{low}(v_i)$ and $P_{upp}(v_i)$ with completion times □ and □ respectively. Since □> □,□ these schedules are □-schedules of $P_{low}(v_i)$ and $P_{upp}(v_i)$ respectively. Using g and h, find □-schedules g' and h' for $P_{low}(v_i)$ and $T_{high(v)}$ by using **FMAX**(□, $P_{low}(v_i)$, $v_i$) and **FMIN**(□, $P_{upp}(v_i)$, $v_i$).

If the completion time of $v_i$ in the □-scheduling g' of $P_{low}(v_i)$ is greater that the completion time of $v_i$ in the □-scheduling h' of $P_{upp}(v_i)$ then no □-scheduling f exists, for $P(J_n,<)$, since otherwise f would induce □-schedulings of $P_{low}(v_i)$ and $T_{high(v)}$ with completion time $c(v_i)$ greater than or equal to Min(□, $P_{low}(v_i)$,v) and smaller than or equal to Max(□,$P_{upp}(v_i)$,v), which is a contradiction.

On the other hand if the completion time of $v_i$ in the □-scheduling g' of $P_{low}(v_i)$ is smaller than or equal to the completion time of $v_i$ in the □-scheduling h' of $P_{upp}(v_i)$, then we can get an □-schedule for T in which the completion time for any u □ $P_{low}(v_i)$, $u \neq v$ is the same as the completion time of  in h' and for any u □ $P_{upp}(v_i)$ including v itself, the completion time of u is that of g'.

Since **FMIN**(□, $P_{low}(v_i)$, g, v) and **FMAX**(□, $P_{upp}(v_i)$, h, v) can be carried out in linear time checking if a □-schedule for T exists can be done in linear time. Our result now follows.

QED

## 3. Concluding Remarks

We have presented an $O(n \log^2 n)$ time algorithm to find optimal schedulings with communication delays for tree orders. We believe that this algorithm, however, is not optimal and that it may be possible to obtain an $O(n \log n)$ time algorithm or even a linear time one.

In some special cases, it is possible to find linear time algorithms. For example if a tree order $P(J_n,<)$ is such that it has a unique maximal element, i.e. an element $v_i$ such that for any $v_j$ □ $P(J_n,<)$, $v_i \neq v_j$ we have $v_j < v_i$, it is easy to find a linear time scheduling algorithm.

To see this, let us assume that the elements of T are labelled $v_1,\ldots,v_n$ in such a way that if $i<j$ then $v_j \not< v_i$. It is easy to see that the following linear time procedure will obtain an optimal scheduling with communication delays for $P(J_n,<)$.

For i=1 to n do<br>
    If $v_i$ is a minimal element,<br>
        $c(v_i)=1$<br>
    Else<br>
        Let $v_{i(1)},\ldots,v_{i(k)}$ be the lower covers of $v_i$. Assume w.l.o.g. that $c(v_{i(j)}) \leq c(v_{i(k)})$, $1 \leq j < k$<br>
        If $k \geq 2$ and $c(v_{i(k-1)})=c(v_{i(k)})$<br>
            $c(v_i)=c(v_{i(k)})+2$

else

$c(v_i)=c(v_{i(k)})+1$

endif

endif.

**References**

[1] H.H. Ali and H. El-Rewini, "An Optimal Algorithm for Scheduling Interval Ordered Tasks with Communication on N Processors", University of Nebraska at Omaha, Mathematics and Computer Science Department, Technical Report 91-20, 1990.

[2] T. Casavant and J. Kuhl, "A Taxonomy of Scheduling in General Purpose Distributed Computing Systems", *IEEE Transactions on Software Engineering*, **SE-14:2**, February 1988.

[3] E. Coffman and R. Graham, "Optimal Scheduling for Two Processor Systems", *Acta Informatica* **1**, 1972, 200-213.

[4] T. Hu, "Parallel Sequencing and Assembly Line Problems", *Operations Research* **9**, 1961, 841-848.

[5] C.H. Papadimitriou and M. Yanakakis, "Scheduling Interval-Ordered Tasks", *SIAM Journal of Computing* 8, 1979, 405-409.

[6] M. Prastein, "Precedence-Constrained Schedulings with Minimum Time and Communication", M.S. Thesis, University of Illinois at Urbana-Champaign, 1987.

[7] R. Sethi, "Scheduling Graphs on Two Processors", *SIAM Journal of Computing* **5**, 1976, 73-82.

[8] J. Ullmann, "NP-Complete Scheduling Problems", *Journal of Computing and System Sciences* 1**0**, 1975, 384-393.