A Dynamic Programming algorithm for minimizing total cost of duplication in scheduling an outtree with communication delays and duplication

Claire Hanen
Laboratory LIP6
4, place Jussieu
F-75 252 Paris cedex 05

Dalila Tayachi
Laboratory LIP6
4, place Jussieu
F-75 252 Paris cedex 05

July 13, 2004

Abstract

This paper introduces an algorithm using dynamic programming to solve the problem of finding the minimum cost of duplication in scheduling with small communication delays and an unbounded number of processors. When duplication is allowed, makespan can be improved but it creates a cost wich can be important depending on the number and the cost of duplicates. The cost of a duplication in a schedule is defined as the sum of costs of tasks (*i.e.* original and duplicates). we assume in this work that the tasks have the same processing time d, that communication delays are all equal to $c \leq d$ and that the precedence graph is an out-tree. We study the problem of finding the minimum cost of a feasible schedule with makespan t.

1 Introduction

The problem we consider here is a scheduling problem with interprocessor communication delays. This problem is modeled by a set of dependent tasks that will be executed by a set of parallel processors connected by a network.

Besides classical precedence constraints between tasks, communication delays must be included in account. This problem is modeled by a directed acyclic graph, the nodes of which are tasks. An arc from task i to task j means that i computes data that is an input for j. If these two tasks are not performed by the same processor, a delay must be considered between the completion of i and the beginning of j to dispatch the data through the network. The aim is to find a schedule that minimizes the makespan. This problem is NP-hard. Even if we assume a non restricted number of processors, unitary processing times and unitary communication delays (UET-UCT task systems), the problem of finding the minimum makespan denoted by $\overline{P}|prec, c_{jk} = 1, p_j = 1|C_{max}$ is NP-hard [5]. For more details see the two surveys [6]and [2]. Polynomial algorithms were developed in very special cases when assumption are supposed on the structure of graph precedence, the number of processors and/or the communication delays. For example, if we assume that communication delays are less than or equal to the processing times of the tasks (this case is the small communication delays), and that we have an unbounded number of processors and an out-tree, Chretienne [1] shows that the problem is solvable in polynomial time.

Another important assumption is duplication. Indeed, if a task i has several successors $j_1, \ldots, j_k$, then performing task i on k processors may allow the execution of $j_1, \ldots, j_k$ on these processors just after i, avoiding communication through the network. Colin and chretienne [3] solved the problem of minimizing the makespan in the case of duplication, small communication delays,and an unbounded number of processors. However, in computer systems, cost of duplication can be very expensive. Indeed, the initial data for each duplicated task must be sent to each processor through the network. And, while the network bandwidth is supposed without limitations, too many duplicates may limit the system performance. Existing scheduling algorithms for tasks systems on an unlimited number of processors make use of a great number of duplicates which can imply a high cost. Recently, C.Hanen and A.M.Kordon [4] studied the minimization of the whole number of duplicates, among the feasible schedules with a makespan at most t in the case of an unbounded number of processor. They proposed a polynomial algorithm based on dynamic programming.

In this paper we aim to solve the problem of minimizing the total cost of duplicates. We assume that the precedence structure is an out-tree, that all tasks have same duration d, and that communication delays are all equal to $c \leq d$. we present in section 2 the problem and several useful dominance

properties which are the adaptation of dominance properties developed in C.Hanen and A.M.Kordon [4]. In section 3, we propose a polynomial time algorithm to determine the minimum cost for any value t of the makespan, together with a feasible schedule that realizes it. The last section discuss the perspectives of this work.

2 Problem definition

Let T a set of tasks with duration $d \in I\!N^*$ indexed from 1 to $|T|$ and $G = (T, \mathcal{A})$ an out-tree $\mathcal{A}$ rooted by task 1. $\forall i \in T$, $\mathcal{A}(i)$ denotes the sub-tree of $\mathcal{A}$ rooted by i and, if $i \neq 1$, $p(i)$ denotes the unique immediate predecessor of i in $\mathcal{A}$. $\forall i \in T$, $\Gamma^+(i)$ is the set of immediate successors of i in $\mathcal{A}$. We consider that the tasks are numbered such that, $\forall j \in \Gamma^+(i)$, $i < j$. We suppose that the value of the communication delays is $c \in I\!N^*$ with $c \leq d$.

We assume that we have an unbounded number of processors and that any task i may be duplicated (*i.e.* performed several times)in order to reduce the communication delays between i and some of its immediate successors. A feasible schedule assigns to each task a set of duplicates, called copies, and to each copy a non-negative starting time and a processor, such that:

1. For any arc (i, j) of G, if a copy of j is performed at time β on processor P, then either a copy of i is performed at time $\beta - d - c$ on a processor $P' \neq P$ or a copy of i starts on the same processor P at time $\beta - d$.

2. Each processor performs at most one copy of a task per time unit.

The makespan of a feasible schedule is the difference between the last completion time of a duplicate, and the start time of the first copy of the root of $\mathcal{A}$. For any feasible schedule σ, we denote by $\mathcal{P}_i(\sigma)$ the set of processors performing $i \in T$, by w_i the cost of a copy i and by $n_i(\sigma)$ the number of copies of i. $w(\sigma)$ is the total cost of tasks (*i.e.* , original and duplicates) of σ is calculated as below:

$$w(\sigma) = \sum_{i \in T} n_i(\sigma) \times w_i$$

We are interested in this article in determining the minimum cost of a schedule with makespan at most t. Given $t \in I\!N^*$ we denote by $D(t)$ the set of feasible schedules with makespan bounded by t, and by $D^*(t)$ the subset of schedules of $D(t)$ with minimum cost. We denote by $W^*(t)$ the cost of a schedule in $D^*(t)$.

A set $D' \subset D(t)$ is said to be dominant if $D' \cap D^*(t) \neq \emptyset$. Three dominance properties 1, 2 and 3, that allow us to consider a dominant subset of schedules will be presented.

Property 1 *The set of feasible schedules verifying the two following properties is dominant: for any task $i > 1$, let us consider a processor $P \in \mathcal{P}_i$.*

1. *If $P \in \mathcal{P}_i(\sigma) \cap \mathcal{P}_{p(i)}(\sigma)$ and if $p(i)$ is performed by P at time γ, then i is performed by P at $\gamma + d$.*

2. *If $P \in \mathcal{P}_i(\sigma) - \mathcal{P}_{p(i)}(\sigma)$ and if the starting time of the earliest execution of $p(i)$ is α, then i is performed by P at $\alpha + c + d$,*

Proof
Let σ be a feasible schedule in $D(t)$ and i a task. We prove that if σ does not satisfy one of the dominance properties for some copy of task i, then a schedule $\sigma' \in D(t)$ with $w(\sigma') \leq w(\sigma)$ that meets the requirements for this copy of i can be built.

1. Assume that condition 1 is not satisfied. If for some processor $\pi \in \Pi_i(\sigma) \cap \Pi_{p(i)}(\sigma)$, $p(i)$ starts at γ on π, then a copy of i is performed by π at time $\gamma + d + \epsilon$, for some $\epsilon > 0$. If $\epsilon \geq c$ then we can build a new schedule by moving the copy of i and all the tasks in $\mathcal{A}(i)$ performed on π in σ from π to a new processor π' with the same starting times (so that condition 2 may not be satisfied, and the rule of next paragraph may be applied). Otherwise, $\epsilon < c \leq d$ so that no task can be performed on π between $p(i)$ and i. Hence, without violating the precedence constraints, the copy of i can start on π at time $\gamma + d$.

2. Assume that condition 2 is not satisfied. Then a copy of i starts on some processor π at time β, while the first copy of $p(i)$ starts on another processor π' at time α. As σ is feasible, $\beta \geq \alpha + d + c$. If $\beta > \alpha + c + d$ then we can remove i and all the tasks in $\mathcal{A}(i)$ performed on π in σ from π, and assign the copies on a new processor π'' as follows: start i at time $\alpha + c + d$, and schedule the other removed copies with the same starting times as in σ. The new schedule is feasible and its makespan is at most t.

By applying these transformations iteratively to each copy of each task in increasing order of number and starting times, then a schedule σ' that

satisfies the two properties and has at most the same number of copies and the makespan of σ can be built. This implys that we can built a scheduling with at most the same cost of duplication. We deduce that if $\sigma \in D^*(t)$ then so is σ'. □

In the following we consider only feasible schedules that meet property 1.

Property 2 *The set of feasible schedules σ for wich all copies of a task are performed at the same time is dominant.*

Proof
Let $\sigma \in D(t)$ be a feasible schedule. Let i a task whose first copy is performed at time α, another copy of i is performed at time $\alpha + k$ with $k \succ 0$.We build a schedule with makespan less than t and cost less than $w(\sigma)$ such that for any task, all copies are performed at the same time. If we consider j a task in $\Gamma^+(i)$, according to property 1 either a copy of j is performed at time $\alpha + d$ or at time $\alpha + d + c$. If a copy of j is performed at time $\alpha + d$, then a copy of i is necessarly performed on the same processor at time α. Otherwise, the copy of j is performed at most at time $\alpha + d + c$ and we can consider that the first copy of i starting at time α delivers data to j through the network. Hence the copy of i performed at time $\alpha + k$ can be removed without violating any precedence constraint. We can do that for any copy of i performed after time α. If we apply iteratively this procedure to a schedule $\sigma \in D^*(t)$ we obtain an optimal scheduling satisfying property 2.

In the following we assume that feasible schedules satisfy properties 1 and 2. We present now the third dominance property:

Property 3 *The set of schedules σ such that, for all $i \in T$,*

$$n_i(\sigma) = \max\{1, \sum_{j \in J} n_j(\sigma)\}$$

with $J = \{j \in \Gamma^+(i) | t_j(\sigma) = t_i(\sigma) + d\}$ is dominant.

Proof
If $J = \emptyset$ then by property 2, all successors j of i satisfy $t_j(\sigma) = t_i(\sigma) + d + c$. Hence only one copy of i performed at time $t_i(\sigma)$ is sufficient to meet the precedence constraints. So if $n_i(\sigma) > 1$, then we can remove useless copies of

i and get a feasible schedule with a lower number of copies until $n_i(\sigma) = 1$. Hence we get feasible schedule with a cost $w(\sigma')$lower then $w(\sigma)$.
If $J \neq \emptyset$ For any copy of $j \in J$ performed on a processor P, there must be a copy of i that is performed just before on the same processor. Hence $n_i(\sigma) \geq \sum_{j \in J} n_j(\sigma)$. If there a copy of i is not mapped with some task of J, it can be removed without violating the precedence constraints. Hence in both cases, one can build a schedule σ' with a number of copies small then the number of copies of the schedule σ teherfore $w(\sigma') \leq w(\sigma)$such that $n_i(\sigma') = \max\{1, \sum_{j \in J} n_j(\sigma')\}$. Applying this transformation iteratively leads the property 3. □

It's useful to limit the number of copies in any feasible schedule σ satisfying the dominance properties 1, 2 and 3. If we denote by l_i the number of leaves of $\mathcal{A}(i)$, the following property offers an upper bound on the number of copies.

Property 4 $\forall i \in T, n_i(\sigma) \leq l_i$.

Proof
We prove it by recurrence on the tree structure.

- If task $i \in T$ is a leaf, then by property 3, $n_i(\sigma) = 1 = l_i$.
- Otherwise, if J is defined as previously for task i, by recurrence,we have $n_j(\sigma) \leq l_j$ so $\sum_{j \in J} n_j(\sigma) \leq \sum_{j \in J} l_j$. We know by property 3that $n_i(\sigma) = \max\{1, \sum_{j \in J} n_j(\sigma)\}$, so $n_i(\sigma) \leq max(1, \sum_{j \in J} l_j) \leq \sum_{j \in \Gamma^+(i)} l_j \leq l_i$ □

In the rest of the paper, feasible schedule means that the schedule is feasible and satisfies properties 1, 2 and 3.

3 Description of the algorithm

The algorithm we present here is based on dynamic programming. It allows us to compute the minimum cost of duplication $W^*(t)$ for any $t \in I\!N^*$ and a corresponding schedule. Minimum completion time can be obtained without duplicating any path from the root to a leaf to get a feasible schedule as the algorithm in [3] does. It is clear that this algorithm produces an important number of useless duplicates and therefore it produces a higher cost.

We now introduce some notations that will be useful to derive a dynamic programming scheme for the computation of $W^*(t)$.

$\forall i \in T, \forall t \in I\!N^*$ and $\forall n \in I\!N^*$, we will denote by

- $D_i(t)$ the set of feasible schedules of $\mathcal{A}(i)$ with makespan at most t,
- $W_i(t) = \min_{\sigma \in D_i(t)} w(\sigma)$. If $D_i(t) = \emptyset$, $W_i(t) = +\infty$.
- $D_i(n,t)$ the subset of schedules of $D_i(t)$ such that the root i has at most n copies.
- $W_i(n,t) = \min_{\sigma \in D_i(n,t)} w(\sigma)$. If $D_i(n,t) = \emptyset$,$W_i(n,t) = +\infty$.

Clearly, $D_i(n,t) \subseteq D_i(n+1,t)$, so $W_i(n,t) \geq W_i(n+1,t)$. Moreover, by property 4, we get

$$D_i(t) = \bigcup_{n \in I\!N^*} D_i(n,t) = D_i(l_i,t)$$

Hence

$$W_i(t) = \min_{n \in \{1,\ldots,l_i\}} W_i(n,t) = W_i(l_i,t)$$

Two lemma are deduced:

Lemma 1 *If j is a leaf then $\forall t \geq d$, $W_j(1,t) = w_j$ and $\forall t < d$, $W_j(1,t) = +\infty$.*
If j is any node and $n > l_j$, then $W_j(n,t) = W_j(l_j,t)$

Now, we can prove the following inequality :

Lemma 2 *Let $\sigma \in D_i(n,t)$ and J be the set of immediate successors of i starting their executions at the completion time of i. Then,*

$$w(\sigma) \geq \max\{w_i, \sum_{j \in J}(n_j(\sigma) + w_j(n_j(\sigma), t-d))\} + \sum_{j \in \Gamma^+(i)-J} w_j(l_j, t-d-c)$$

Proof
Cost of tasks (copies) performed by σ is the sum of three costs: cost of copies of i, cost of copies of tasks of $\mathcal{A}(j)$,denoted by $W(\mathcal{A}(j))$ for $j \in J$ and the cost of tasks of $\mathcal{A}(j)$, for $j \in \Gamma^+(i) - J$. We distinguish two cases:

- If $J = \emptyset$ then $n_i(\sigma) = 1$ and $w(\sigma) = w_i + \sum_{j\in\Gamma^+(i)-J} W(\mathcal{A}(j))$. This cost is greater then $w_i + \sum_{j\in\Gamma^+(i)-J} W_j(n_j(\sigma), t-d-c)$ or we know by property 3 that $n_j(\sigma) \leq l_j$ so $W_j(n_j(\sigma), t-d-c) \geq W_j(l_j, t-d-c)$ and $w(\sigma) \geq w_i + \sum_{j\in\Gamma^+(i)-J} W_j(l_j, t-d-c)$.

- If $J \neq \emptyset$ then $w(\sigma) = \sum_{j\in J} n_j(\sigma)\times w_i + \sum_{j\in J} W(\mathcal{A}(j)) + \sum_{j\in\Gamma^+(i)-J} W(\mathcal{A}(j))$
  $\geq \sum_{j\in J} n_j(\sigma) \times w_i + \sum_{j\in J} W_j(n_j, t-d) + \sum_{j\in\Gamma^+(i)-J} W_j(l_j, t-d-c)$
  so $w(\sigma) \geq \sum_{j\in J} n_j(\sigma)\times w_i + W_j(n_j(\sigma), t-d) + \sum_{j\in\Gamma^+(i)-J} W_j(l_j, t-d-c)$
  Hence, in the two cases we have

$$w(\sigma) \geq \max\{w_i, \sum_{j\in J}(n_j(\sigma)\times w_i + W_j(n_j(\sigma), t-d))\} + \sum_{j\in\Gamma^+(i)-J} W_j(l_j, t-d-c)$$

The following theorem will be obtained by proving the converse inequality :

Theorem 1 *$\forall t \in I\!N^*$, $\forall i \in T$ such that $\Gamma^+(i) \neq \emptyset$, $\forall n \leq l_i$,*

$$W_i(n,t) = \min_{J\subset\Gamma^+(i)} \{wi_{J=\emptyset} + \sum_{j\in\Gamma^+(i)-J} W_j(l_j, t-d-c)$$

$$+ \min_{0<n_j\leq l_j, \sum_{j\in J} n_j \leq n} \sum_{j\in J} n_j \times w_i + W_j(n_j(\sigma), t-d)\}$$

Proof
Let $\sigma \in D_i(n,t)$ be a schedule with a minimum cost $w(\sigma) = W_i(n,t)$. We have by lemma 2, $W_i(n,t)$ is greater than the right term of the equality. If this term is infinite, also is $W_i(n,t)$.

Conversely, if $W_i(n,t)$is finite, we can build a feasible schedule of $D_i(n,t)$ with a cost exactly the right term of equality as below:

- If i is a leaf, since $W_i(n,t)$ is finite,necessarily we have $t \geq d$. Hence, we can perform one execution of i at time 0, obtaining a cost of w_i.

- If i is not a leaf, $\Gamma^+(i) \neq \emptyset$. Let J^* be a subset of $\Gamma^+(i)$ that realizes the right term of the equality and the associated numbers of executions $n_j^* > 0, j \in J^*$.Let σ_j be a schedule of cost $w_j(n_j^*, t-d)$ for $j \in J^*$ and of cost $w_j(l_j, t-d-c)$ for $j \in \Gamma^+(i) - J^*$. Let us assign disjoint subset of processors to the schedules $\sigma_j, j \in \Gamma^+(i)$.

1. $\forall j \in \Gamma^+(i) - J^*$, make a right shift on σ_j so that the first task starts at time $d+c$. As the makespan of σ_j is at most $t-d-c$, the resulting schedule has a makespan at most t.

2. if $J^* = \emptyset$, perform a copy of i at time 0.

3. Otherwise, for each $j \in J^*$, shift σ_j so that j starts at time d. On each of the n_j^* processors that perform a copy of j, start a copy of i at time 0.

The schedule obtained is feasible and its cost is exactly the right term of the equality. □

4 Computation of the costs

In this section, we prove that the costs $w_i(n,t)$ may be polynomially computed using the relation expressed by theorem 1. Firstly, we reduce the total number of useful values of the state variable $t \in \mathbb{N}$ to a finite set of polynomial size. Then, we introduce some intermediate steps for the computation of $W_i(n,t)$. Lastly, we evaluate the complexity of this algorithm.

4.1 Time domain

From property 1, the makespan of any schedule from $D_1(t)$, $t \in \mathbb{N}$ starting at time 0 can be decomposed as $t_{h,k} = hd+k(d+c)$, with $(h,k) \in \mathbb{N} \times \mathbb{N}$. Now, let us consider t^* the minimal length of a schedule of $\mathcal{A}$ without duplication (this value can be polynomially computed by the algorithm of P.Chrétienne [1]). t^* is bounded by $|T|d + (|T|-1)c$.

1. $\forall t \geq t^*$, $\mathcal{A}$ may be scheduled with makespan t^* without duplication, so $W_1(t) = W_1(t^*) = \sum_{i=1}^{|T|} w_i$.

2. $\forall t$ such that $d \leq t < t^*$, let $t_{h,k}$ be the greatest value $t_{h,k}$ such that $t_{h,k} \leq t$. Then $W_i(t) = W_i(t_{h,k})$.

3. If $t < d$, there is no feasible schedule, so $W_i(t) = +\infty$.

So, we will limit the time domain τ to the values $t_{h,k}$ with $h \leq \lceil \frac{t^*}{d} \rceil$, $k \leq \lceil \frac{t^*}{(d+c)} \rceil$ and $hd + k(d+c) \leq t^*$. The size of this domain is roughly bounded by $|T|^2$.

4.2 Computation of $W_i(n,t)$

Let us consider a task $i \in T$ such that $\Gamma^+(i) \neq \emptyset$. In order to compute the minimum cost $W_i(n,t)$ of theorem 1, we decompose it over the set of successors of i. We suppose that, for every $j \in \Gamma^+(i), \forall n' \leq l_j$ and $\forall t' \in \tau$ we have computed $W_j(n',t')$. The aim is here to compute $W_i(n,t)$, $\forall n \leq l_i$ and $\forall t \in \tau$.

Let us consider $\Gamma^+(i) = \{j_1, \ldots, j_{|\Gamma^+(i)|}\}$ the set of immediate successors of i. $\forall k \in \{1, \ldots, |\Gamma^+(i)|\}$, we will denote by $C_i(n,t,k)$ the minimum cost of a schedule σ of the subgraph $\mathcal{A}(i) - \mathcal{A}(j_{k+1}) \ldots - \mathcal{A}(j_{|\Gamma^+(i)|})$ such that :

1. the makespan of σ is at most t,
2. $n_i(\sigma) \leq n$,
3. there is at least one $j \in \{j_1, \ldots, j_k\}$ such that $t_j(\sigma) = t_i(\sigma) + d$.

Let us consider a feasible schedule $\sigma \in D_i(n,t)$.

- If no successor of i is performed at the completion time of i in σ, then $w(\sigma) \geq w_i + \sum_{j \in \Gamma^+(i)} w_j(l_j, t-d-c)$. The right term of the inequality is the cost of a schedule built by performing a single copy of i at time 0, and starting at time $d+c$ the schedules of $\mathcal{A}(j), j \in \Gamma^+(i)$ with costs $w_j(l_j, t-d-c)$. Hence $w_i(n,t) \leq w_i + \sum_{j \in \Gamma^+(i)} w_j(l_j, t-d-c)$
- Otherwise, at least one successor of i is performed at its completion time and we get $w(\sigma) \geq C_i(n,t,|\Gamma^+(i)|)$. Similarly, the right term is the cost of a feasible schedule with at most n copies of i and makespan at most t, so that $w_i(n,t) \leq C_i(n,t,|\Gamma^+(i)|)$.

Hence, if we consider a schedule σ such that $w(\sigma) = w_i(n,t)$, we get:

$$w_i(n,t) = \min\{C_i(n,t,|\Gamma^+(i)|), w_i + \sum_{j \in \Gamma^+(i)} w_j(l_j, t-d-c)\}$$

The second term of this minimum can be easily computed. We will present now how to compute the value $C_i(n,t,k)$ by recurrence on $k \in \{1, \ldots, |\Gamma^+(i)|\}$.

- If $k = 1$, then j_1 is performed at the end of i, so

$$H_i(n,t,1) = \min_{1 \leq m \leq n} (mw_i + w_{j_1}(m, t-d))$$

- Now, let us consider $k > 1$. By theorem 1 and since $J \neq \emptyset$, we get :

$$H_i(n,t,k+1) = \min_{J \subseteq \{j_1,\dots j_{k+1}\}, J \neq \emptyset} \{ \sum_{j \in \{j_1,\dots j_{k+1}\}-J} W_j(l_j, t-d-c) +$$

$$\min_{0<n_j \leq l_j, \sum_{j \in J} n_j \leq n} \sum_{j \in J} (n_j w_i + W_j(n_j, t-d)) \}$$

Let J^* be an optimal subset (for which the right term of the previous equality is minimum). Three cases may occur :

1. If $j_{k+1} \notin J^*$, then there is a communication delay between i and j_{k+1} and thus the sub-schedule of $\mathcal{A}(j_{k+1})$ has a makespan bounded by $t-d-c$ and an unconstrained number of copies of j_{k+1} :

$$C_i(n,t,k+1) = W_{j_{k+1}}(l_{j_{k+1}}, t-d-c) + C_i(n,t,k)$$

2. If $J^* = \{j_{k+1}\}$, then the sub-schedules of $\mathcal{A}(j_l)$, $1 \leq l \leq k$ have a makespan bounded by $t-d-c$, the sub-schedule of $\mathcal{A}(j_{k+1})$ has a makespan bounded by $t-d$ and to each copy of j_{k+1} corresponds a copy of i. So we get:

$$C_i(n,t,k+1) = \sum_{j \in \{j_1,\dots,j_k\}} W_j(l_j, t-d-c)$$

$$+ \min_{1 \leq m \leq n} (m w_i + W_{j_{k+1}}(m, t-d))$$

3. Otherwise, $\{j_{k+1}\} \subset J^*$ and thus the sub-schedule of i and the sub-trees $\mathcal{A}(j_l)$, $1 \leq l \leq k$ satisfy the requirements of $C_i(n-m,t,k)$ for some m. As previoulsy the sub-schedule of $\mathcal{A}(j_{k+1})$ has a makespan bounded by $t-d$ and to each copy of j_{k+1} corresponds a copy of i. Hence we get :

$$C_i(n,t,k+1) = \min_{1 \leq m < \min(n, l_{j_{k+1}})} (m w_i + W_{j_{k+1}}(m, t-d) + C_i(n-m,t,k))$$

$C_i(n,t,k+1)$ will be obtained by getting the minimum of these 3 values.

4.3 Complexity of the algorithm

Let us consider $i \in T$, $n \in \{1, \ldots, l_i\}$ and $t \in \tau$. The complexity of the computation of $W_i(n,t)$ is $O(|\Gamma^+(i)|(|\Gamma^+(i)|+n))$. We deduce that the complexity of $W_i(n,t), n \in \{1, \ldots, l_i\}$ is $O(|\Gamma^+(i)|)l_i^2$. So, the complexity of the computation of every value $W_i(n,t)$ is $O(|T|^3.|\tau|) = O(|T|^5)$.

For a fixed value t, an optimal schedule associated with $W_1(t)$ can be built using a recursive algorithm of complexity also bounded by $O(|T|^5)$.

$W_j(l_j, t-d-c)$

References

[1] P. Chrétienne. A polynomial time to optimally schedule tasks over an ideal distributed system under tree-like precedence constraints. *European Journal Operations Research*, 2:225–230, 1989.

[2] P. Chrétienne and C. Picouleau. *Scheduling with communication delays a survey : in P. Chretienne, E.G. Coffman, J.K. Lenstra, Z. Liu (Eds.), Scheduling theory and its applications*, pages 65–89. John Wiley Ltd., New york, 1995.

[3] J-Y. Colin and P. Chrétienne. CPM scheduling with small communication delays and task duplication. *Operations Research*, 39:681–684, 1991.

[4] C. Hanen and A.M. Kordon. Minimizing the volume in scheduling an out-tree with communication delays and duplication. *Parallel computing*, 28:1573–1585, 2002.

[5] C. Picouleau. Two new NP-complete scheduling problems with communication delays ans unlimited number of processors. *Discrete Applied Mathematics*, 60:331–342, 1995.

[6] B. Veltman and B.J. Lenstra. Multiprocessor scheduling with communication delays. *Parallel computing*, 16:173–182, 1990.