Minimizing the volume in scheduling an outtree with communication delays and duplication

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

Alix Munier Kordon
Laboratory LIP6
4, place Jussieu
F-75 252 Paris cedex 05

April 23, 2003

Abstract

We consider in this paper a scheduling problem with small communication delays and an unbounded number of processors. It is known that duplication can improve the makespan of schedules. However, scheduling algorithms may create a huge amount of duplicates. The volume of a schedule is defined as the total number of tasks (*i.e.* original and duplicates). Assuming 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 volume of a feasible schedule with makespan t. We derive some dominance properties and prove that this problem is polynomial using a dynamic programming algorithm.

1 Introduction

With the recent development of parallel architectures arise a new class of scheduling problems in which communication delays are considered. The target machine is a set of parallel processors connected by a network. A parallel program is modeled as usual 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.

Several studies are devoted to this kind of problems (*c.f.* the two surveys [3, 11]). Most of them are NP-hard even with very strong assumptions. For example, if we assume unitary processing times and unitary communication delays (UET-UCT task systems), and if a non restricted number of processors are available, Picouleau proved in [9] that the makespan minimization problem denoted by $\overline{P}|prec, c_{jk} = 1, p_j = 1|C_{max}$ is NP-hard.

Task duplication might be useful to reduce the influence of communication delays. 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.

This feature also reduces the problem complexity : if communication delays are less than or equal to the processing times of the tasks, Colin and Chrétienne [4] proved that the makespan minimization problem on an unbounded number of processors is polynomial.

Unfortunately, if no assumption is made on the communication delays, the problem $\overline{P}|prec, c_{jk} = c > 1, p_j = 1, duplication|C_{max}$ is NP-hard [8]. So the UET-UCT assumption seems to be a borderline between easy and hard problems if duplication is allowed.

However, in computer systems, duplication is not costless. 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. In order to reduce it, one can solve the scheduling problem assuming a limited number of processors. But then the problems become difficult to handle, even for simple graph structures. An outcome of the result of Veltman [10] is that $P|in-tree, c_{jk} = 1, p_j = 1, duplication|C_{max}$ is NP-hard (in this case duplication is useless). Some approximations results can also be mentioned : we proved in [7] that a greedy algorithm with relative performance equal to $2-\dfrac{1}{m}$ can be developed for the problem $P|prec, c_{jk} = 1, p_j = 1, duplication|C_{max}$.

In [8] an approximation algorithm for $\overline{P}|prec, c_{ij}, p_j, duplication|C_{max}$ with worst case performance ratio 2 is developed, and Munier proved in [6] that the ideas of this algorithm can be used to solve $P|tree, c_{ij}, p_j|C_{max}$ with a relative performance bounded by $1 + (1 - \frac{1}{m})(2 - \frac{}{1+\rho})$, where m denotes the number of processors, ρ the ratio between communication delays and processing times.

The control of the number of duplicates is tackled by another way in this paper. We still consider an unbounded number of processors, and we study the minimization of the whole number of duplicates, called the volume of the schedule, among the feasible schedules with a makespan at most t. As the problem can be easily proven NP-hard for general graph structures, or general communication delays, we focus on a sub-problem, in order to develop tools that might later be used for deriving approximation and/or lower bounds for more general settings. 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 $c \leq d$.

Section 2 presents the problem and several useful dominance properties. In the third section, we prove a recurrence equation satisfied by the mimimum volume of sub-trees. In section 4, we propose a polynomial time algorithm to determine the minimum volume for any value t of the makespan, together with a feasible schedule that realizes it. In section 5, we present some experimental results. The last section discuss the perspectives of this work.

2 Problem Formulation

Let us consider a set T of tasks with duration $d \in I\!N^*$ indexed from 1 to $|T|$ and 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 consider the problem of scheduling task set T on an unbounded number of processors allowing duplication. A feasible schedule assigns to each task a set of duplicates, also called copies, and to each copy a non-negative starting time and a processor, such that if $j \in \Gamma^+(i)$ for any copy of j starting at time β on a processor π, either a copy of i starts on a processor $\pi' \neq \pi$ at the latest at time $\beta - d - c$ or a copy of i starts on the same processor π at the latest at time $\beta - d$.

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 $\Pi_i(\sigma)$ the set of processors performing $i \in T$ and by $n_i(\sigma)$ the number of copies of i. $v(\sigma)$ is the total number of tasks (*i.e.* , original and duplicates) of σ. Clearly,

$$v(\sigma) = \sum_{i \in T} n_i(\sigma)$$

We call this value the *volume* of σ. If there is no confusion, σ may be omitted in the notations.

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

A set $S' \subset S(t)$ is said to be dominant if $S' \cap S^*(t) \neq \emptyset$. In order to reduce the combinatorics of our scheduling problem, we prove three dominance properties 1, 2 and 3, that allow us to consider a dominant subset of schedules.

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

1. *If $\pi \in \Pi_i(\sigma) \cap \Pi_{p(i)}(\sigma)$ and if $p(i)$ is performed by π at time γ, then i is performed by π at $\gamma + d$.*
2. *If $\pi \in \Pi_i(\sigma) - \Pi_{p(i)}(\sigma)$ and if the starting time of the earliest execution of $p(i)$ is α, then i is performed by π at $\alpha + c + d$,*

Proof

Let σ be a feasible schedule in $S(t)$ 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 $\sigma' \in S(t)$ with $v(\sigma') \leq v(\sigma)$ 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 sarting 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 volume and the makespan of σ can be built. Hence if $\sigma \in S^*(t)$ then so is σ'. $\square$

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

Property 2 *The set of feasible schedules σ such that, for any task $i \in T$, the executions of i are all performed at the same time denoted by $t_i(\sigma)$, is dominant.*

Proof

Let $\sigma \in S(t)$ be a feasible schedule. Let i be the task with minimum index with two different execution times respectively β and $\beta + k$ with $k > 0$. We build a schedule with makespan less than t and volume less than $v(\sigma)$ such that for any task $j \leq i$, all copies of j are performed at the same time.

If $i = 1$, then we can obviously perform all the copies of the root at the same time β without violating precedence constraints. Hence, we consider $i > 1$.

Since $p(i) < i$, all the copies of $p(i)$ are performed at the same time α. By property 1, the copies of i are performed either at time $\alpha + d$ or at time $\alpha+d+c$. Hence $\beta = \alpha+d, k = c$. Since the first instance of i is performed at time β, any task j from $\Gamma^+(i)$ should start, according to property 1, at most at time $\beta+d+c$, which is exactly the completion time of the second instance of i. So, this instance can be removed without violating any precedence constraint. Obviously, this can be done with any copy of i that is performed after β.

Applying iteratively this procedure to a schedule $\sigma \in S^*(t)$ leads to an optimal schedule that satisfies property 2. □

Notice that the property 2 is not true anymore if the communication delays are not all equal to c. For example, let us consider the five tasks $T = \{1, 2, 3, 4, 5\}$ of duration 6 and the out-tree pictured by figure 1. For $t = 20$, the optimum schedule has two executions of task 2 with distinct starting times.

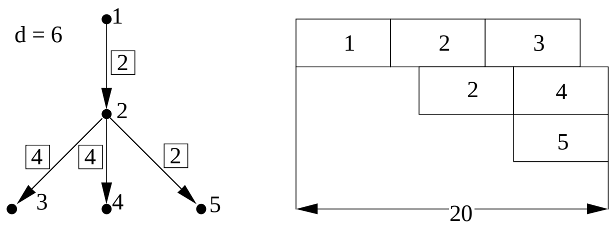

Figure 1: A counter example for property 2. Arcs of the out-tree are valued by the communications delays.

In the following we assume that feasible schedules satisfy properties 1 and 2. Let us now present our last dominance property:

Property 3 *Let $i \in T$. 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 volume until $n_i(\sigma) = 1$.

If $J \neq \emptyset$, by property 2, all the executions of $j \in \Gamma^+(i)$ are performed at time $t_i(\sigma) + d$, so $J = \Gamma^+(i)$. Moreover, for any copy of j performed on a processor π, 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 $v(\sigma') \leq v(\sigma)$ such that $n_i(\sigma') = \max\{1, \sum_{j \in J} n_j(\sigma')\}$. Applying this transformation iteratively leads the property 3. □

In order to get an upper bound on the number of copies in any feasible schedule σ satisfying the dominance properties 1, 2 and 3 we denote by l_i the number of leaves of $\mathcal{A}(i)$. We can then stress the following property:

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, by recurrence, if J is defined as previously for task i,

$$n_i(\sigma) \leq max(1, \sum_{j \in J} l_j) \leq \sum_{j \in \Gamma^+(i)} l_j$$

□

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

3 A recurrence relation

The aim of this part is to compute the minimum volume $V^*(t)$ for any $t \in \mathbb{N}^*$ and a corresponding schedule. First, we present a small example to illustrate some simple remarks on the structure of optimal schedules. Then, we introduce some additional notations and variables and we prove the recurrence relation on which the computation of $V^*(t)$ will be based.

Let us consider the small example pictured by figure 2. The first schedule has a minimum number of duplicates. This example leads to the two following remarks :

- Even if the completion time has a minimum value, there is no need to duplicate any path from the root to a leaf to get a feasible schedule as the algorithm in [4] does. It is clear that this algorithm produces an important number of useless duplicates.

- If we decide not to duplicate task 7, we get a schedule of volume equal to 12. So, we must here duplicate an internal node without duplicating its predecessor. This means that the structure of solutions may be much complicated than in the case where the number of duplicates is not limited.

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

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

- $S_i(t)$ the set of feasible schedules of $\mathcal{A}(i)$ with makespan at most t,

- $V_i(t) = \min_{\sigma \in S_i(t)} v(\sigma)$. If $S_i(t) = \emptyset$, $V_i(t) = +\infty$.

- $S_i(n,t)$ the subset of schedules of $S_i(t)$ such that the root i has at most n copies.

- $V_i(n,t) = \min_{\sigma \in S_i(n,t)} v(\sigma)$. If $S_i(n,t) = \emptyset$,$V_i(n,t) = +\infty$.

Clearly, $S_i(n,t) \subseteq S_i(n+1,t)$, so $V_i(n,t) \geq V_i(n+1,t)$. Moreover, by property 3, we get

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

Hence

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

With the previous definitions it can be directly established that:

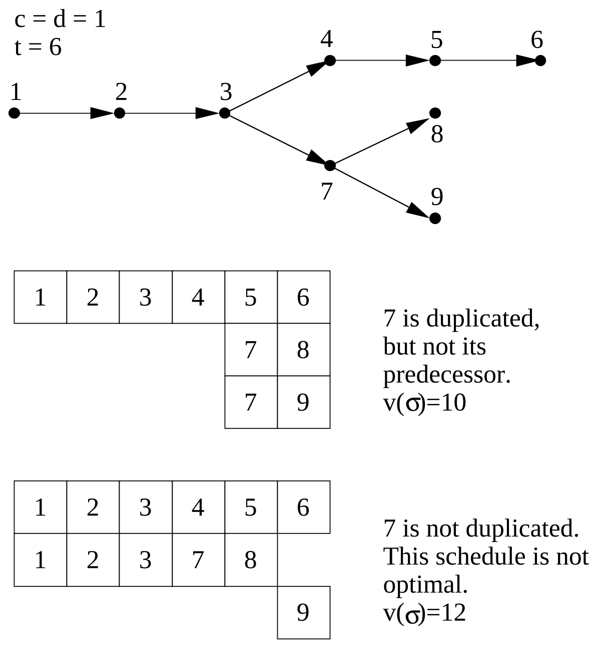
c = d = 1
t = 6
1
2
3
4
5
6
7
8
9
1 2 3 4 5 6
7 8
7 9
7 is duplicated, but not its predecessor.
v(σ)=10
1 2 3 4 5 6
1 2 3 7 8
9
7 is not duplicated. This schedule is not optimal.
v(σ)=12

Figure 2: Two remarks concerning the structure of an optimal schedule

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

Now, we can prove the following inequality :

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

$$v(\sigma) \geq \max\{1, \sum_{j \in J}(n_j(\sigma) + V_j(n_j(\sigma), t-d))\} + \sum_{j \in \Gamma^+(i)-J} V_j(l_j, t-d-c)$$

Proof
Tasks (copies) performed by σ can be partitioned into 3 sets :

- The set of copies of i. By definition, $n_i(\sigma) = \max\{1, \sum_{j \in J} n_j(\sigma)\}$.
- The set of tasks from $\mathcal{A}(j)$, for $j \in J$. The volume of the sub-schedule of σ for $\mathcal{A}(j)$ is by definition at least $V_j(n_j(\sigma), t-d)$.

- The set of tasks from $\mathcal{A}(j)$, for $j \in \Gamma^+(i) - J$. The volume of the sub-schedule of σ for $\mathcal{A}(j)$ is greater than $V_j(n_j(\sigma), t-d-c)$. By property 3, $n_j(\sigma) \leq l_j$, so $V_j(n_j(\sigma), t-d-c) \geq V_j(l_j, t-d-c)$. $\Box$

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

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

$$V_i(n,t) = 1_{J=\emptyset} + \min_{J \subset \Gamma^+(i)} \{ \sum_{j \in \Gamma^+(i)-J} V_j(l_j, t-d-c)$$

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

Proof
Let $\sigma \in S_i(n,t)$ be a schedule with a minimum volume $v(\sigma) = V_i(n,t)$. By lemma 2, $V_i(n,t)$ is greater than the right term of the equality. If this term is infinite, also is $V_i(n,t)$.

Conversely, suppose now that this term is finite. Then, we can build an optimal schedule of $S_i(n,t)$ by induction using the following scheme :

(B) If i is a leaf, since $V_i(n,t)$ is finite necessarily we have $t \geq d$. Hence, we can perform one execution of i at time 0, obtaining a volume of 1.

(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 volume $V_j(n_j^*, t-d)$ for $j \in J^*$ and of volume $V_j(l_j, t-d-c)$ 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.

It is clear that this schedule is feasible and its volume is exactly the right term of the equality. □

4 Computation of the volumes

In this section, we prove that the volumes $V_i(n,t)$ 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 $V_i(n,t)$. Lastly, we evaluate the complexity of this algorithm.

Time domain From property 1, the makespan of any schedule from $S_1(t)$, $t \in \mathbb{N}$ starting at time 0 can be decomposed as $t_{\alpha,\beta} = \alpha d + \beta(d+c)$, with $(\alpha,\beta) \in \mathbb{N} \times \mathbb{N}$. 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 [2]). 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 $V_1(t) = V_1(t^*) = |T|$.
2. $\forall t$ such that $d \leq t < t^*$, let $t_{\alpha,\beta}$ be the greatest value $t_{\alpha,\beta}$ such that $t_{\alpha,\beta} \leq t$. Then $V_i(t) = V_i(t_{\alpha,\beta})$.
3. If $t < d$, there is no feasible schedule, so $V_i(t) = +\infty$.

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

Computation of $V_i(n,t)$ Let us consider a task $i \in T$ such that $\Gamma^+(i) \neq \emptyset$. In order to compute the recurrence equation 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 $V_j(n',t')$. The aim is here to compute $V_i(n,t)$, $\forall n \leq l_i$ and $\forall t \in \tau$.

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

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

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

- If no successor of i is performed at the completion time of i in σ, then $v(\sigma) \geq 1 + \sum_{j \in \Gamma^+(i)} V_j(l_j, t-d-c)$. The right term of the inequality is the volume 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 volumes $V_j(l_j, t-d-c)$. Hence $V_i(n,t) \leq 1 + \displaystyle\sum_{j \in \Gamma^+(i)} V_j(l_j, t-d-c)$
- Otherwise, at least one successor of i is performed at its completion time and we get $v(\sigma) \geq F_i(n,t,|\Gamma^+(i)|)$. Similarly, the right term is the volume of a feasible schedule with at most n copies of i and makespan at most t, so that $V_i(n,t) \leq F_i(n,t,|\Gamma^+(i)|)$.

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

$$V_i(n,t) = \min\{F_i(n,t,|\Gamma^+(i)|), 1 + \sum_{j \in \Gamma^+(i)} V_j(l_j, t-d-c)\}$$

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

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

$$F_i(n,t,1) = \min_{1 \leq m \leq n} (m + V_{j_1}(m, t-d))$$

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

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

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

  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} :

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

  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:

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

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

  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 $F_i(n-m,t,k)$ 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 :

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

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

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

For a fixed value t, an optimal schedule associated with $V_1(t)$ can be built using a recursive algorithm of complexity also bounded by $O(|T|^5)$.

5 Experimental results

For these experiments, we generated random trees using algorithms presented in [1]. The questions that we aimed to answer, depending on the parameters of the input, were the following:

- What improvement of the makespan can be expected from using duplication?
- What is the minimal cost of such an improvement (in terms of number of duplicates)

For 100 tasks and several values of the ratio $\rho = \frac{c}{d}$, we generated 500 instances of the scheduling problem and we computed the following values:

1. **Percentage of duplication**, ie. the percentage of instances belonging to the set $\mathcal{T}$ for which the duplication reduces the minimum makespan of a schedule.
2. **Average value of improvement**, ie. if $C_{max}(\mathcal{A})$ (Resp. $C^d_{max}(\mathcal{A})$) for $\mathcal{A} \in \mathcal{T}$ denotes the minimal value of a schedule of $\mathcal{A}$ without (Resp. with) duplication, we computed the average value of the ratio $\dfrac{C_{max}(\mathcal{A}) - C^d_{max}(\mathcal{A})}{C_{max}(\mathcal{A})}$.
3. **Average minimum volume**, ie. the average of the volume $V(C^d_{max}(\mathcal{A}))$ for every $\mathcal{A} \in \mathcal{T}$.

The following table summarizes our results:

ρ	1	0.75	0.5	0.25	0.1
Percentage of duplication	30%	24%	20%	26%	26%
Average value of improvement	4.5%	3.5%	1.4%	1.1%	0.4%
Average minimum volume	138	140	135	144	140

We observed that experimentally, duplication doesn't seem to reduce the makespan of the optimal schedule significantly and the results presented here are quite poor. Our explanation is that the structure of the trees randomly generated is very irregular and such that every node has a few number of immediate successors. For this class of trees, the duplication doesn't dramatically improve the makespan of the schedules. But the improvement has an important cost since the number of duplicates is between 35% and 40% of the number of tasks. Notice that the results are slightly better for $\rho = 1$. However, the small improvement of the makespan due to duplication makes difficult on our random trees other kind of experiments that could have been interesting, for example as measuring the variation of the minimum volume according to the makespan of the schedule.

Further research may lead us to look for some particular structures for which the improvement is significant, in order to answer the following question: if we allow a makespan which is $\alpha * C_{max}^{opt}$ then the minimum volume will be $\beta * |T|$. How can β be expressed in terms of α?

We actually think that full binary trees are good candidates for such a study. Indeed, the duplication improves significantly the makespan, as shown by the following table for a tree fo heigt 6 (*i.e.* with 127 nodes).

ρ	1	0.75	0.5	0.25	0.1
Average value of improvement	46%	40%	30%	17.6%	7.8%
Average minimum volume	448	448	448	448	448

Figure 3 shows, for a full binary tree of height 6, the variations of the minimum volume with respect to the makespan for the values $\rho = 1$ and $\rho = 0.5$.

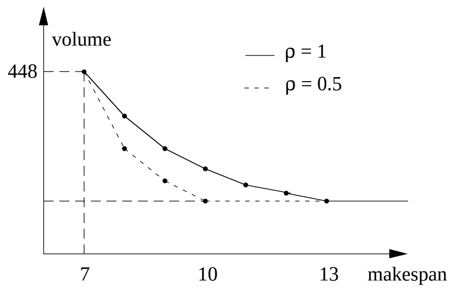

Figure 3: Optimal volume according to the makespan for a full binary tree with $\rho = 1$ and $\rho = 0.5$

In [5], Jung et al. proved that, for a full binary tree of n nodes, any schedule with makespan t has a volume greater than $\frac{d}{c} n^{\frac{c}{2dt}}$. Notice that, in our case (*i.e.* $n = 127$ and small communication value), this bound is very far from the optimal value. For example, for $\rho = 1$ we get $n^{\frac{1}{2t}}$ which is even less than n. Better lower bounds for small communication delays should be interesting to investigate.

6 Perspectives

In this paper, we introduced a new objective function for scheduling with communication delays in order to reduce both the number of duplicates and the makespan. We proved that the minimization of the volume for an out-tree with equal processing times d and equal communication delays $c \leq d$ is polynomial. Many questions arose from this result, as :

- Is it possible to predict the number of duplicates necessary to get a bounded error with respect to the optimal makespan?
- Is it possible to extend these results to other special classes of precedence graphs ?
- Is it possible to use this result for minimizing the makespan for a scheduling problem with communication delays and a limited number of resources ?

7 Acknowledgement

We wish to thank F.Guinand and the anonymous referee for the their helpful remarks concerning the readability and motivations of this paper.

References

[1] L Alonso and R Schott. *Random Generation of trees*. Kluwer academic Publishers, 1995.

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

[3] P Chrétienne and C Picouleau. Scheduling with communication delays : a survey. In P Chrétienne, E.G Coffman, J.K Lenstra, and Z Liu, editors, *Scheduling Theory and its Applications*, pages 65–89. John Wiley Ltd, 1995.

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

[5] H Jung, L Kirousis, and P Spirakis. Lower bounds and efficient algorithms for multiprocessor scheduling of DAG's with communication delays. In *1.SPAA*, pages 254–264, 1989.

[6] A Munier. Approximation algorithms for scheduling trees with general communication delays. *Parallel Computing*, 25:41–48, 1999.

[7] A Munier and C Hanen. Using duplication for scheduling unitary tasks on m processors with communication delays. *Theoretical Computer Science*, (178):119–127, 1997.

[8] C Papadimitriou and M Yannakakis. Towards an architecture independent analysis of parallel algorithms. *SIAM Journal on Computing*, 19:322–328, 1990.

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

[10] B Veltman. *Multiprocessor scheduling with communication delays*. PhD thesis, CWI-Amsterdam, 1993.

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