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
IUP Miage, university Paris 12
71, rue Saint-Simon
F-94 017 Creteil Cedex

June 5, 2001

Abstract

We consider in this paper a scheduling problem given by n tasks of same processing time d, an out-tree, communication delays all equal to $c \leq d$ and an integer t. The problem is to find the minimum volume of a feasible schedule with makespan t. Studying the dominance properties of such schedules, we 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][9]). 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 infinitely many processors are available, Picouleau proved in [7] 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 infinitely many processors is polynomial. Unfortunately, if no assumption is made on the communication delays, the problem $\overline{P}|duplication, c_{jk}|C_{max}$ is NP-hard [6]. So the UET-UCT assumption seems to be a borderline between easy and hard problems if duplication is allowed.

The main drawback concerning duplication on an unlimited number of processors is that the number of duplicates may be important. One solution to reduce it is to fix a limited number of processors. Veltman [8] proved that the problem without duplication $P|in - tree, c_{jk} = 1, p_j = 1|C_{max}$ is NP-hard. But for this class of graph, duplication is useless since a task has at most one successor. Notice that we proved in [5] that a greedy algorithm with relative performance equal to $2 - \dfrac{1}{m}$ may be developed for the problem $P|duplication, c_{jk} = 1, p_j = 1|C_{max}$.

In this paper, we consider another way to reduce the number of duplications. We study the minimization of the overall number of duplicates (called the volume of the schedule) for a class of feasible schedules with a bounded makespan t. We develop a polynomial algorithm for tasks with same duration d, an out-tree and communication delays all equal to $c \leq d$.

In the second section, we introduce the notations and we prove several useful dominance properties. In the third section, we prove a recurrence property on the definition of the volumes. In the section 4, we prove that we can compute polynomially these values and an optimal schedule for any makespan t. In section 5, we present some experimental results. The last section is devoted to a discussion on 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 number 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 $c \leq d$.

The duplication is allowed and the number of processors is unlimited. For any feasible schedule σ, $\Pi_i(\sigma)$ denotes the set of processors performing $i \in T$ and $n_i(\sigma)$ the number of executions of i. $v(\sigma)$ is the total number of tasks (*i.e.* , original and duplicates) of σ. Clearly,

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

This value is called the *volume* of σ. If there is no confusion, σ may be omitted in the notations.

$\forall i \in T$ and $\forall t \in I\!N^*$, we will denote $S_i(t)$ the set of feasible schedule of $\mathcal{A}(i)$ with completion time bounded by t and $V_i(t)$ the minimum volume of a schedule of $S_i(t)$. We will also consider that every element from $S_i(t)$ verifies the dominance property 1, 2 and 3 proved in the following. If $S_i(t) = \emptyset$, we set $V_i(t) = \infty$. Clearly,

$$V_i(t) = \min_{\sigma \in S_i(t)} v(\sigma)$$

The aim of this paper is to compute $V_1(t)$ for any $t \in I\!N^*$ and a corresponding schedule.

Property 1 *The set of schedules such that :*

1. *for any task $i > 1$ with $\pi \in \Pi_i(\sigma) \cap \Pi_{p(i)}(\sigma)$, if $p(i)$ is performed by π at time t, then i is performed by π at $t + d$,*
2. *for any task $i > 1$ with $\pi \in \Pi_i(\sigma) - \Pi_{p(i)}(\sigma)$, if the starting time of the earliest execution of $p(i)$ is t, then then i is performed by π at $t + c + p$,*

is dominant.

Proof
Let σ be a feasible schedule.

1. If there is no task assigned to π between the execution of $p(i)$ and i, then i can be obviously performed by π at time $t + d$. Otherwise, this execution of i can be removed from π and, if needed, executed by another processor earlier at time $t + c$.

2. The proof is similar to the previous one. $\Box$

Property 2 *The set of schedules 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 σ be a schedule which verifies the property 1. Let i be the earliest task with two executions performed respectively at times t and $t + \alpha$ with $\alpha > 0$. If $i = 1$, then we can obviously perform all the executions of the root at time 0. So, we consider $i > 1$.

All the executions of $p(i)$ are performed at time $t' = t - d$. Indeed, if $t' < t - d$, then by property 1, the two considered executions of i are performed by other processors. So, they are performed simultaneously at time $t' + c + p$.

So, the first execution of i is performed at the completion of $p(i)$ and the second one at time $t + c$. Since the first instance of i is performed at time t, any tasks from $\Gamma^+(i)$ is started at most at time $t + d + c$, which is exactly the completion time of the second instance of i. So, this instance can be removed without any modification on its immediate successors. $\Box$

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 four tasks $T = \{1, 2, 3, 4\}$ and the out-tree pictured by figure 1. For $t = 20$, the optimum schedule has two executions of task 2 with distinct starting times.

The following property is a simple consequence of property 2 :

Property 3 *The set of schedules such that,*

1. *for any task i, if J is the set of immediate successors of i with $\forall j \in J$, $t_j(\sigma) = t_i(\sigma) + d$, then $n_i(\sigma) = 1_{J=\emptyset} + \sum_{i \in J} n_j(\sigma)$,*

2. *the number of executions of any task $i \in T$ is bounded by the number of leaves of $\mathcal{A}(i)$,*

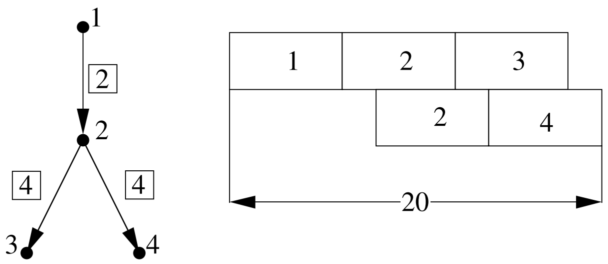

Figure 1: A counter example for property 2

is dominant. In the following, this number of leaves will be denoted by l_i.

Proof

1. By property 2, all the executions of $j \in J$ are performed at $t_j(\sigma)$, so $n_i(\sigma) \geq 1_{J=\emptyset} + \sum_{i \in J} n_j(\sigma)$. Since the overall number of duplicates is minimum, the equality holds.

2. We prove by recurrence on the structure of the tree that, $\forall i \in T, n_i(\sigma) \leq l_i$.

   - If task $i \in T$ is a leaf, then, by the previous equality, $n_i(\sigma) = 1 = l_i$,
   - otherwise, by recurrence, $n_i(\sigma) \leq 1_{J=\emptyset} + \sum_{j \in J} l_j$. Since

$$\max(1_{J=\emptyset}, \sum_{j \in J} l_j) \leq \sum_{j \in \Gamma^+(i)} l_j$$

this property is true. □

3 A recurrence relation on the minimum volumes

The aim of this part is to show a recurrence relation on the minimum volumes. First, we present a small example to illustrate some simple remarks on the structure of optimal schedules. Then, we introduce some more notations and we prove the recurrence relation.

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 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 to not 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.

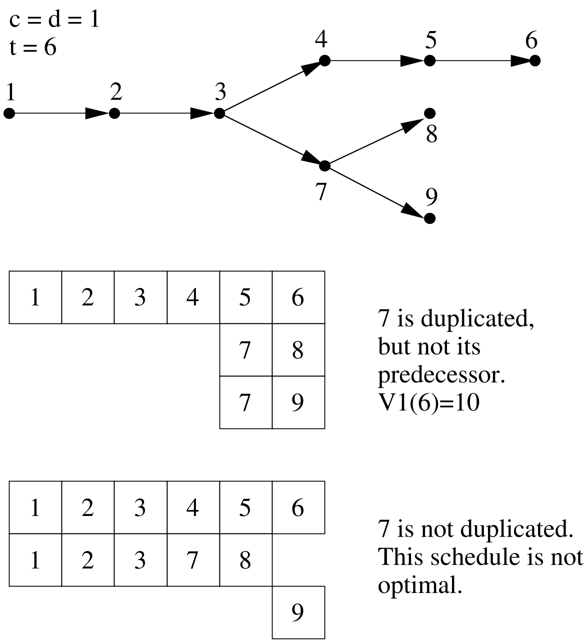

Figure 2: Two remarks concerning the structure of an optimal schedule

In the following, we will denote by $S_i(n,t)$ the set of feasible schedule of $\mathcal{A}(i)$ with completion time at most t and such that the root i has at most n executions. We also suppose that $S_i(n,t)$ is reduced to schedules verifying the 3 previous dominance properties.

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

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

where l_i denotes the number of leaves of $\mathcal{A}(i)$.

$\forall n \in \{1,\ldots,l_i\}$, $V_i(n,t)$ is the minimum volume of a schedule from $S_i(n,t)$. Here also, $V_i(n,t) \geq V_i(n+1,t)$, so we get

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

By convention, if $S_i(n,t) = \emptyset$, we set $V_i(n,t) = +\infty$. This leads to the following lemma :

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

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 beginning their executions at the completion time of i. Then,*

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

Proof
Tasks (original and duplicates) from σ can be partitioned into 3 sets :

- $n_i(\sigma)$ executions of i,
- tasks from $\mathcal{A}(j)$, $\forall j \in J$. The volume of the sub-schedule of σ for $\mathcal{A}(j)$ is then greater than $V_j(n_j(\sigma), t-d)$.
- Tasks from $\mathcal{A}(j)$, $\forall j \in \Gamma^+(i) - J$. 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, $n_j(\sigma) \leq l_i$, so $V_j(n_j(\sigma), t-d-c) \geq V_j(l_j, t-d-c)$. $\square$

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

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

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

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

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 minimum is finite. Then, we can build an optimal schedule of $S_i(n,t)$ by induction by setting :

(B) If i is a leaf, then $n = 1$. Since this minimum is finite, $t \geq 1$ and perform one execution of i at time t.

(I) Now $\Gamma^+(i) \neq \emptyset$. Let $J^* \subseteq \Gamma^+(i)$ verifying the equality and the corresponding number of executions $n_j^* > 0, j \in J^*$.

1. perform $1_{J^*=\emptyset} + \sum_{j \in J^*} n_j^*$ executions of i at time 0,
2. $\forall j \in J^*$, start one optimal schedule from $S(n_j^*, t-d)$ at time d on the same processors than n_j^* executions of i.
3. $\forall j \in \Gamma^+(i) - J^*$, start one optimal schedule from $S(l_j, t-d-c)$ at time $d+c$ on new processors. □

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 show that only a polynomial number of times $t \in I\!N$ have to be considered. Then, we will introduce some middle steps for the computation of $V_i(n,t)$. Lastly, we will evaluate the complexity of this algorithm.

Times domain The makespan of any schedule verifying property 1 may be equal to $t_{\alpha,\beta} = \alpha d + \beta c$, with $(\alpha, \beta) \in I\!N^+ \times I\!N$. Now, let us consider $t^* = \alpha^* d + \beta^* c$ the minimal length of a schedule of $\mathcal{A}$ without duplication (this value may polynomially computed by the algorithm of P.Chrétienne [2]). We get $t^* \leq |T|d + (|T| - 1)c$.

1. $\forall t \geq t^*$, $\mathcal{A}$ may be scheduled with makespan t^* without any duplicate, so $V_1(t) = V_1(t^*)$.

2. $\forall t$ such that $d \leq t < t^*$, let the greatest value $t_{\alpha,\beta}$ with $t_{\alpha,\beta} \leq t$. 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 times domain τ to the values $t_{\alpha,\beta}$ with $\alpha \leq \alpha^*$, $\beta \leq |T| - 1$ and $\alpha^* d + \beta^* c \leq t^*$. 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 $\Gamma^+(i) \neq \emptyset$. 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 $\Gamma^+(i) = \{j_1, \ldots, j_{|\Gamma^+(i)|}\}$ the set of immediate successors of i. $\forall k \in \{1, \ldots, |\Gamma^+(i)|\}$, we will denote by $F_i(n, t, k)$ the minimum volume of a schedule σ of the subgraph $\mathcal{A}(i) - \mathcal{A}(j_{k+1}) \ldots, -\mathcal{A}(j_{\Gamma^+(i)})$ such that :

1. the makespan of σ is at most t,

2. the number of executions of i in σ is bounded by n,

3. there is at least one $j \in \{j_1, \ldots, j_k\}$ which executions are performed immediately at the end of i in σ.

Any schedule from $S_i(t, n)$ has either no successors of i performed at the completion time of i, either at least one. So, 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 $k \in \{1, \ldots, |\Gamma^+(i)|\}$.

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

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

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

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

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

Let J^* be an optimal subset. Three cases may occur :

1. If $j_{k+1} \notin J^*$, then there is a communication delay between i and j_{k+1} and :

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

2. If $J^* = \{j_{k+1}\}$, then

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

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

3. Otherwise, $\{j_{k+1}\} \subset J^*$ and

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

$F_i(n,t,k+1)$ 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 generate random trees using [1]. 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 allow to reduce the makespan of the schedules.

The following table summarizes our experimental results for random trees with 100 tasks.

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 maximum volume	138	140	135	144	140

For every value of the ratio $\rho = \frac{c}{d}$, we generate 500 instances of the scheduling problem and we compute 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 $\frac{C_{max}(\mathcal{A}) - C^d_{max}(\mathcal{A})}{C_{max}(\mathcal{A})}$.

3. **Average maximum volume**, ie. the average of the volume $V(C^d_{max}(\mathcal{A}))$ for every $\mathcal{A} \in \mathcal{T}$.

Notice that the results are slightly better for $\rho = 1$. Now, we can compare this results with full binary trees : it is a case for which the duplication is very efficient to reduce the makespan of schedules. The following table presents the result for a complete binary tree of height 6 (ie., 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 maximum volume	448	448	448	448	448

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 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 ?

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

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

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

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

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