Approximation algorithms for scheduling with a limited number of communications

Chams Lahlou

Approximation Algorithms for Scheduling with a Limited Number of Communications

Chams Lahlou
Laboratoire d'Informatique de Paris 6.
4, place Jussieu. 75252 Paris cedex 05.
email : Chams.Lahlou@lip6.fr

Abstract : We consider the case of a UET tree and an unlimited number of processors. We give a 2-approximation algorithm for minimizing the total number of communications, when there are no communication delays. This algorithm allows us to design a 6-approximation algorithm for the makespan minimization problem when there are unit length communication delays and the processors are interconnected by a simgle bus. Finally, we compare our 6-approximation algorithm with other algorithms by simulations. We show that its mean performance is regular (around 1.25) and we explain, for certain cases, the performance of these algorithms by considering some parameters of the problem.

1 Introduction

In a multiprocessor system there are usually interprocessor communications during the execution of a parallel program because of data transfers between tasks. It is therefore important to consider scheduling models with realistic communication constraints if one wants to apply results from scheduling theory to practical parallel computation problems. For instance, in order to take into account the duration of a data transfer between two tasks scheduled by different processors, Rayward-Smith [10] proposed in 1987 a basic model with unit execution time and unit communication delays (UET-UCT for short). Since then, scheduling problems with communication delays have been studied and important results have been obtained (the reader is referred to [4] for a survey on the subject). However, in most of these problems it is supposed that there is no communication contention. Whereas this assumption seems reasonable for some multiprocessor systems (if processors are fully connected for instance), it is not realistic if the communication resources are strongly limited as it is the case for a single bus machine.

In this paper we consider two scheduling models where the communication contention is modelled by a limitation of the number of communications. The parallel program is represented by a precedence graph which is a directed acyclic graph whose vertices represent tasks and arcs represent data dependencies among tasks. If two dependent tasks are not scheduled by the same processor there is an interprocessor communication between their execution for the data transfer. The first model (UET-bound) has been proposed by Afrati, Papadimitriou and Papageorgiou [1] on the one hand, and Prastein [9] on the other hand: Tasks have unit execution time, there are no communication delays and the total number of communications is limited. More recently, Finta and Liu [5] have considered the case of a single bus machine with the UET-UCT assumption. In their model, called UET-UCT-bus, they study the special case where the bus capacity is one, that is there can be at most one communication per unit of time (a similar model for the particular case with two processors has been also proposed by Norman, Pelagatti and Thanish [8]). They also define two data semantics, that is the kind of data dependencies among the tasks. For the case of *independent-data semantics*, all data sent by a task are different and hence if a task has to communicate to k tasks there are exactly k communications. On the contrary, for the case of *common-data semantics* all data sent by a task are identical. So if a task has to send data to k tasks scheduled by a set of k' processors only k' communications are needed. In [1] the independent data semantics is considered and in [9] the common data semantics. In both cases, it is shown that scheduling a UET tree on an unlimited number of identical processors in at most T unit of time and with at most C communications is an NP-complete problem. In [5] the authors are concerned with the two semantics. They show that finding a minimum length schedule (i.e., with a minimum makespan) of a UET-UCT precedence graph on a single bus machine with capacity one is NP-hard for several cases, particularly if the task graph is a binary tree and the number of processors is infinite.

The purpose of our work is to show that it is possible to design approximation algorithms for the makespan minimization problem in the UET-UCT-bus model by using approximation algorithms for the communications minimization problem in the UET-bound model. Our paper is organized as follows. In Section 2, we establish the connection between the two models from the approximation point of view. In Section 3, we present an algorithm for the problem of minimizing the total number of communications in the UET-bound model, for the special case of a tree and an unlimited number of processors. We then prove that its relative performance is bounded by 2. In Section 4, we first show that under restrictive assumptions it is possible to find in a polynomial time a schedule of an arbitrary precedence graph on a single bus machine. Then we give a 6-approximation algorithm for the makespan minimization problem in the UET-UCT-bus model, again for the special case of a tree and an unlimited number of processors. In Section

5, we compare the previous algorithm with some heuristics by the use of simulations, and Section 6 concludes the paper.

2 The Connection between UET-bound and UET-UCT-bus

A schedule for a precedence graph $G = (V, E)$, in the model UET-bound, assigns a starting time t_i and a unique processor π_i to each task i of V such that (1) for every couple of tasks (i, j), if $\pi_i = \pi_j$ then $t_i \neq t_j$ (a processor schedules at most one task at at each unit of time), and (2) for each precedence constraint $(i, j) \in E$, $t_j \geq t_i + 1$.

In the model UET-UCT-bus, a schedule for a precedence graph $G = (V, E)$, with the independent-data semantics, on a single bus machine with capacity B assigns a starting time t_i and a processor π_i to each task i of V such that (1) for every couple of tasks (i, j), if $\pi_i = \pi_j$ then $t_i \neq t_j$, (2) for each precedence constraint $(i, j) \in E$, if $\pi_i = \pi_j$ then $t_j \geq t_i + 1$ else there is a *communication task* m_{ij} such that $t_{m_{ij}} \geq t_i + 1$ and $t_j \geq t_{m_{ij}} + 1$, and (3) at each time t, $|\{m_{ij} : t_{m_{ij}} = t\}| \leq B$ (there can be no more than B communications per unit of time). If the common-data semantics is considered, all data sent by a task i are identical, so we have to add the following constraint : (4) if m_{ij} and $m_{ij'}$ are two communication tasks such that $\pi_i \neq \pi_j$ and $\pi_j = \pi_{j'}$ then $m_{ij} = m_{ij'}$ else $m_{ij} \neq m_{i'j'}$.

Let us now consider two optimization problems, MINCOM-BOUND for the UET-bound model and MINTIME-BUS for the UET-UCT-bus model. In the MINCOM-BOUND problem we are asked to find a schedule with makespan at most T units of time (that is $\max_{i \in V}\{t_i + 1\} \leq T$) such that the total number of communications is minimum (the definition of communication depends on the choice of the data semantics). Notice that we have $l \leq T < n$ (where l is the length of a longest path in G, and $n = |V|$ is the number of tasks) since there exists no schedule with makespan less than l, and since, if $T \geq n$, we can get an optimum solution by scheduling all tasks on a single processor. Moreover, since $l \leq T$, it is always possible to find a solution. In the MINTIME-BUS problem we are asked to find a schedule with minimum makespan.

We now show how the two problems are related. Let $Algo_1$ be an approximation algorithm for MINCOM-BOUND, $c(t)$ the total number of communications in a solution provided by $Algo_1$ when the makespan is limited to t, and $c^*(t)$ the total number of communications in an optimum solution. Consider the following algorithm $Algo_2$ for MINTIME-BUS :

1. Let l be the length of a longest path in the precedence graph (in number of tasks). We set $t = l$; While $c(t) > (t-1)B$ we set t=t+1;

2. (*creation of communication delays*) the schedule provided at the end of the previous step is scanned from left to right, and each communication (i, j) at time t is replaced by a communication of one unit length : so task j and all its successors are shifted to the right by one unit of time.

3. (*scheduling of communications*) the schedule obtained at the end of the previous step is again scanned from left to right. If at a time t there are more than B communications, the extra communications are delayed by one unit of time.

We then have the following result.

Theorem 1 *If $Algo_1$ is an α-approximation algorithm then $Algo_2$ is a 3α-approximation algorithms.*

Proof – Let T^* denotes the makespan of an optimum schedule for MINTIME-BUS. We show that $Algo_2$ provides a schedule with makespan at most $3\alpha T^*$. Let T_1, T_2 and T_3 be the makespan of the schedules we get by steps 1, 2 and 3 respectively. Since T_1 may be increased (during step 2) by one time unit at each time t where there are communications, and because there are at most $T_1 - 1$ such instants: $T_2 \leq 2T_1 - 1$. Again, T_2 may be increased (during step 3) by one time unit each time there are more than B communications. Since there are at most $(T_1 - 1)B$ communications (by step 1), T_2 is increased at most $T_1 - 2$ times. Hence we get $T_3 \leq T_2 + T_1 - 2 = 3(T_1 - 1)$.

Let T_1^* be the minimum makespan of an optimum schedule such that $c^*(T_1^*) \leq (T_1^* - 1)B$. Since $c^*(T_1^*) \leq (T_1^* - 1)B$, the associated schedule is a valid solution for a relaxed version of MINTIME-BUS (more precisely, when only the total number of communications is limited). Thus we have $T_1^* \leq T^*$. So we consider two cases:

1. If $T_1 = T_1^*$: $T_1 \leq T^*$ and $T_3 \leq 3T^*$.

2. If $T_1 > T_1^*$: by step 1, $(T_1 - 2)B < c(T_1 - 1)$ and (by definition of $Algo_1$) $c(T_1 - 1) \leq \alpha c^*(T_1 - 1)$. If $t \geq t'$ we have $c^*(t) \leq c^*(t')$, so $c^*(T_1 - 1) \leq c^*(T_1^*)$. Hence, $(T_1 - 2)B < \alpha c^*(T_1^*)$ and, since $c^*(T_1^*) \leq (T_1^* - 1)B$, we have $T_1 \leq \alpha T_1^* \leq \alpha T^*$.

Finally, $T_3 \leq 3T_1 \leq 3\alpha T^*$ and the theorem is proved.

□

Note that this result does not depend neither on the bus capacity, nor on the data semantics.

3 Approximation in the UET-bound Model

We are concerned in this Section with the problem MINCOM-BOUND when the precedence graph is a tree (an out-tree, more precisely) and the number of processors is unlimited. This problem is NP-hard for both data semantics since the associated decision problems are NP-complete (see [1] and [9]).

Let A be a tree of height h with n unit execution tasks, and T the makespan limitation ($h \leq T < n$). We consider the independent-data semantics, so the cost (the total number of communications) of a schedule is the number of arcs (i, j) such that i and j are not scheduled by the same processor. Since $T < n$, there is at least one communication in every schedule, and it is possible to define the performance ratio $\frac{c}{c^*}$ where c^* is the cost of a an optimum schedule and c the cost of an approximation algorithm.

We now present an algorithm which approximates the optimum within a ratio of 2 ($\frac{c}{c^*} \leq 2$).

3.1 Presentation of the Algorithm

We first introduce some notations. Let i be a task of A. We denote by $A(i)$ the subtree with root i, by $\Gamma^+(i)$ the set of immediate successors of i, and r_i the rank of i in A, that is the length in number of arcs of the path from the root of A to i.

The schedule of A provided by the algorithm is denoted by σ and an optimum schedule is denoted by σ^*. For a task i of A, π_i is the processor that schedules i in σ, t_i the starting time of i, and $S(i)$ the set of tasks of $A(i)$ that are scheduled by π_i. We define in the same way π_i^*, t_i^* and $S^*(i)$ for an optimum schedule σ^*.

The cost $c(i)$ of a task i in σ is equal to the number of immediate successors of i that are not scheduled by π_i. The cost of a subtree $A(i)$, denoted by $c(A(i))$, is equal to $\sum_{j \in A(i)} c(j)$. The costs $c^*(i)$ and $c^*(A(i))$ in σ^* are defined in the same way.

It is hard to characterize the structure of an optimum solution, but there are however two easy cases. If $A(i)$ is a subtree of A we have:

1. If $r_i + |A(i)| \leq T$, $A(i)$ can be scheduled optimally int $T - r_i$ unit of time by executing all tasks of $A(i)$ by the same processor.

2. If all immediate successors j of i are such that each subtree $A(j)$ can be scheduled by only one processor, we get an optimum schedule of $A(i)$ with makespan at most $T - r_i$ by scheduling as many subtrees $A(j)$ as possible by the processor π_i.

Our algorithm is based on these two special cases:

1. **for** every i in A :

   (a) $S(i) = \{i\}$.

   (b) **if** $r_i + |A(i)| \leq T$ **then** $\tilde{x}_i = 0$.

   (c) **else** $\tilde{x}_i = 1$ and $t_i = r_i$.

2. The schedule σ_i of $A(i)$ is built recursively (starting from the leaves) as follows:

   (a) **if** $\tilde{x}_i = 0$ **then** we set $S(i) = A(i)$ and all tasks of $S(i)$ are scheduled since time r_i by a new processor π_i.

   (b) **else**

      i. **if** i has no immediate successor j such that $\tilde{x}_j = 0$ **then** let k be the immediate successor of i such that $|S(k)|$ is minimal ; we set $S(i) = S(i) \bigcup S(k)$ and $\pi_i = \pi_k$.

      ii. **else** let $S_0 = \{j : j \in \Gamma^+(i) \text{ and } \tilde{x}_j = 0\}$ and l be the list of elements j of S_0 sorted according to the non-decreasing values $|S(j)|$. Let j' be the first element in l.
      while $r_i + |S(i)| + |S(j')| \leq T$ **do**
      $S(i) = S(i) \bigcup S(j')$ and $l = l \backslash \{j'\}$.
      Then let $S_1 = \{j : j \in \Gamma^+(i) \text{ and } \tilde{x}_j = 1\}$. If $S_1 \neq \emptyset$, let $k \in S_1$ a task such that $|S(k)|$ is minimum. If $r_i + |S(i)| + |S(k)| \leq T$ we set $S(i) = S(i) \bigcup S(k)$.
      Finally, we choose a new processor for π_i.

      iii. Tasks of $S(i)$ are then scheduled by π_i: if k has to be scheduled by π_i, the schedule σ_k start at time $t_i + 1 = r_k$ ($t_i = r_i$ by step 1.(b)) ; schedules σ_j, for $j \in S_0$ and $\pi_j = \pi_i$, are then scheduled by π_i one after the other.

An example of execution for the tree of Figure 1 is represented on Figure 2.

3.2 The Performance Guarantee

The proof of the guarantee is based on a classification of the tree nodes. To each i of A is associated a variable x_i whose value is defined recursively (starting from the leaves) by:

$$x_i = \begin{cases} 0 & \text{if } r_i + |A(i)| \leq T \\ 1 & \text{if } r_i + |A(i)| > T \text{ and } \forall j \in \Gamma^+(i) : x_j = 0 \\ 2 & \text{if } r_i + |A(i)| > T \text{ and } \forall j \in \Gamma^+(i) : x_j \in \{1, 2\} \\ 3 & \text{if } r_i + |A(i)| > T \text{ and } \exists\ (j, j') \in \Gamma^+(i) \times \Gamma^+(i) : x_j = 0 \text{ and } x_{j'} \neq 0 \\ 4 & \text{else.} \end{cases}$$

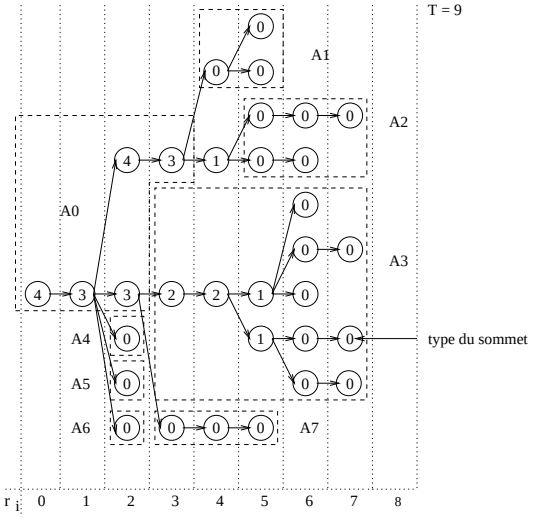

Figure 1: Decomposition of a tree

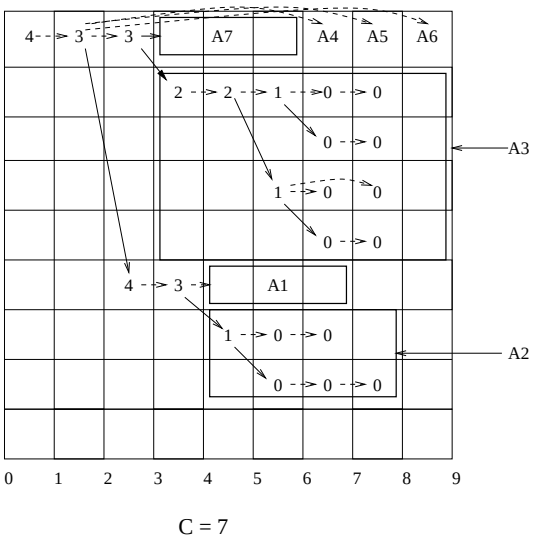

Figure 2: An example of execution

We say that the type of a node i is 0, 1, 2, 3 or 4, according to the value of x_i. In the same way, we say that the type of a subtree $A(i)$ is the type of its root i. This classification leads to the following decomposition of the tree: if we delete, in A, all arcs of type $(3,0)$, $(3,1)$ and $(3,2)$ we get a set of subtrees $\mathcal{A} = \{A_0, \ldots, A_K\}$. Since arcs of type $(3,3)$, $(3,4)$, $(4,4)$ and $(4,3)$ are not deleted, all nodes of type 3 and 4 belong to the same subtree. An example of decomposition is shown on Figure 1.

3.2.1 General Properties

Let i be a node of A, we denote:

- by $\Gamma_0^+(i)$ the set of 0-type immediate successors of i.
- by $\Gamma_{\overline{0}}^+(i)$ the set of all other immediate successors of i.
- by $S_0(i)$ the set of tasks in $\Gamma_0^+(i)$ scheduled by π_i.
- by $S_0^*(i)$ the set of tasks in $\Gamma_0^+(i)$ scheduled by π_i^*.
- by $s_{max}(i)$ the maximum number of subtrees $A(j)$, $j \in \Gamma_0^+(i)$, that can be entirely scheduled by π_i, that is $s_{max}(i) = \max_{S \subseteq \Gamma_0^+(i)} |\{S : r_i + 1 + \sum_{j \in S} |A(j)| \leq T\}$.
- by $b(i) = |\Gamma_0^+(i)| - |S_0(i)|$ the number of 0-type nodes not scheduled by π_i.

A schedule built by the algorithm has the following properties:

Lemma 1 *For every i in A :*

1. $|S_0(i)| = s_{max}(i)$.
2. $c(i) \leq b(i) + |\Gamma_{\overline{0}}^+(i)|$.
3. $c(i) \geq b(i) + |\Gamma_{\overline{0}}^+(i)| - 1$.

Proof –

1. The algorithm allocates all tasks of a 0-type subtree to the same processor. Since at the end of the step where i is considered, a maximum number of 0-type immediate successors of i are allocated to π_i since time r_i, the property is true.
2. At the end of the step where i is considered, at least $|S_0(i)|$ immediate successors of i are allocated to π_i, so $c(i) \leq |\Gamma^+(i)| - |S_0(i)| = |\Gamma_{\overline{0}}^+(i)| + |\Gamma_0^+(i)| - |S_0(i)| = |\Gamma_{\overline{0}}^+(i)| + b(i)$.

3. If i has at least one non 0-type immediate successor, at most one non 0-type immediate successor of i can be allocated to π_i (by step 2.(b).ii). So $c(i)$ is at least equal to $|\Gamma_{\overline{0}}^{+}(i)| + b(i) - 1$.

□

We shall use the following bound for the proof of the performance guarantee:

Lemma 2 *For $i \in A$:*

$$b(i) \leq c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))$$

Proof – Let k_0 and k_0' be the number of elements in $S_0^*(i)$ such that $S^*(j) = A(j)$ and $S^*(j) \neq A(j)$, respectively. Let k_1 be the number of elements in $\Gamma_{\overline{0}}^{+}(i)$ allocated to π_i^*. By definition, $c^*(i) = |\Gamma^+(i)| - k_0 - k_0' - k_1$. Since each of the k_0' 0-type subtrees creates at least one communication, we have $\sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq k_0'$. So we get $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq |\Gamma^+(i)| - k_0 - k_1$. Since $k_0 \leq s_{max}(i) = |S_0(i)|$ (case 1 of property 1), $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq |\Gamma^+(i)| - |S_0(i)| - k_1 = |\Gamma_0^+(i)| - |S_0(i)| + |\Gamma_{\overline{0}}^{+}(i)| - k_1$. But $b(i) = |\Gamma_0^+(i)| - |S_0(i)|$ and $k_1 \leq |\Gamma_{\overline{0}}^{+}(i)|$. It is then true that $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq b(i)$.

□

To make easier the understanding of the performance guarantee proof, we first give a proof for a simple case:

Theorem 2 *If A is a 2-type tree then $c \leq 2c^*$.*

Proof – Let N_1 and N_2 be the 1-type and 2-type nodes of A, respectively. By definition of the algorithm, the cost of a 0-type subtree is 0 in σ, so $c = \sum_{i \in N_1 \cup N_2} c(i)$. Since each 2-type node i has exactly one immediate successor allocated to π_i (see step 2.(b).i), $\sum_{i \in N_2} c(i) = \sum_{i \in N_2} (|\Gamma^+(i)| - 1) = (|N_2| + |N_1| - 1) - N_2$. All immediate successors of a 1-type node i are 0-type nodes, so $c(i) = |\Gamma_0^+(i)| - |S_0(i)| = b(i)$, again by definition of the algorithm. Thus we have $c = \sum_{i \in N_1} b(i) + |N_1| - 1$, and since $1 \leq c(i)$ for $i \in N_1$ we get $c = \sum_{i \in N_1} b(i) + \sum_{i \in N_1} c(i) - 1 \leq 2 \sum_{i \in N_1} b(i) - 1$. Finally, by lemma 2 we have $c \leq 2c^*$.

□

We use the same idea for the general proof: the total cost c is decomposed according to the type of the nodes of A. However, the proof is much more difficult because some nodes can have a zero cost in σ^* but not in σ: for

instance, if i is a 3-type node with only one 1-type immediate successor, it is possible that all immediate successors of i are allocated to π_i^* but there is no proof that they are also allocated to π_i. Therefore it is not possible to bound $c(i)$ by an expression with $c^*(i)$. For i in N_4 (the set of 4-type nodes of A) $c(i) = |\Gamma^+(i)| - 1$, by definition of the algorithm. The value $|N_4|$ can be suppressed in the expression of c, as it is the case for the value $|N_2|$ in the proof of the previous theorem. On the contrary, for a task i in N_3 (the set of 3-type nodes of A) we can not use this trick, and the bound of the cost c looks like $|N_3| + \cdots$. If we want to get a guarantee we have to bound the value $|N_3|$. For this purpose, we shall prove that in fact for some 3-type node with a zero cost in σ^* but not in σ, there exists one successor j (not necessarily immediate) such that: either $c^*(A(j)) \geq 1$, or $c^*(j) + \sum_{k \in \Gamma_0^+(j) c^*(A(k))} \geq 1$, or $c^*(j) \geq 1$. By this way, it is possible to associate to each 3-type node with a zero cost in σ^* but not in σ a successor with a non zero cost.

3.2.2 Properties of 3-type Nodes

We study 3-type nodes according to their cost in σ and σ^*. We also consider 4-type nodes because such a node can be an immediate successor or predecessor of a 3-type node.

Let N_3 and N_4 be the sets of 3-type and 4-type nodes of A, respectively. Consider the following partition of N_3 :

- $N_3^1 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0,\ c(i) = |\Gamma_{\overline{0}}^+(i)|$ and i has at least one 3-type successor$\}$
- $N_3^2 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0,\ c(i) = |\Gamma_{\overline{0}}^+(i)|$ and i has no 3-type successor$\}$
- $N_3^3 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0$ and $c(i) = |\Gamma_{\overline{0}}^+(i)| - 1\}$
- $N_3^4 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 1$ and $c(i) = |\Gamma_{\overline{0}}^+(i)| + 1\}$
- $N_3^5 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 1$ and $c(i) \leq |\Gamma_{\overline{0}}^+(i)|\}$
- $N_3^6 = \{i : c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq 2\}$

and the following partition of N_4 :

- $N_4^1 = \{i : c^*(i) = 0\}$
- $N_4^2 = \{i : c^*(i) \geq 1\}$

It is indeed a partition: if $i \in N_3^1 \cup N_3^2 \cup N_3^3$ then $b(i) = 0$, and thus either $c(i) = |\Gamma_0^+(i)|$ or $c(i) = |\Gamma_0^+(i)| - 1$ (by cases 2 and 3 of lemma 1); if $i \in N_3^4 \cup N_3^5$ then $b(i) \leq 1$, and we have either $c(i) = |\Gamma_0^+(i)| + 1$ or $c(i) \leq |\Gamma_0^+(i)|$ (by the same lemma).

We now express more precisely the value of the bound $b(i)$ of lemma 2 for certain nodes in N_3.

For nodes in $N_3^1 \cup N_3^2 \cup N_3^3$ we have :

Lemma 3 *If i is in an element of $N_3^1 \cup N_3^2 \cup N_3^3$ then :*

1. $b(i) = c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))$
2. $\sum_{j \in S_0^*(i)} |S^*(j)| = \sum_{j \in S_0(i)} |S(j)|$

Proof –

1. It is obvious by lemma 2.
2. Since $c^*(i) = 0$ we have $S_0^*(i) = \Gamma_0^+(i)$, and since $\sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0$, each of the subtrees $A(j)$ ($j \in \Gamma_0^+(i)$) is entirely scheduled by π_i^*. Because $b(i) = 0$ we have $|S_0(i)| = |\Gamma_0^+(i)|$ (see definition of $b(i)$), that is $S_0(i) = \Gamma_0^+(i)$, so $S_0(i) = S_0^*(i)$. hence $\sum_{j \in S_0^*(i)} |S^*(j)| = \sum_{j \in S_0(i)} |S^*(j)| = \sum_{j \in S_0(i)} |S(j)|$.

□

To give an analogous result for nodes in N_3^4 we need the following lemma.

Lemma 4 *Let $i \in A$ and $S' \subseteq \Gamma_0^+(i)$. If $|S'| \geq |S_0(i)|$ then :*

$$\sum_{j \in S'} |A(j)| \geq \sum_{j \in S_0(i)} |S(j)|$$

Proof – Tasks of 0-type subtree $A(j)$ are allocated to the same processor by the algorithm, so $|S(j)| = |A(j)|$. Since nodes j in $S_0(i)$ are chosen one after the other according to the non-decreasing values $|S(j)|$, the lemma is true.

□

Then we have the following.

Lemma 5 *If i is an element of N_3^4 then :*

1. $b(i) = c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))$
2. $\sum_{j \in S_0^*(i)} |S^*(j)| \geq \sum_{j \in S_0(i)} |S(j)|$

3. All non 0-type immediate successors of i are scheduled by π_i^*.

unlimited –

1. If $b(i) = 0$ then $c(i) \leq |\Gamma_0^+(i)|$ by case 2 of lemma 1, which is impossible. So we have $b(i) \geq 1$, and since $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 1$ the property is true.

2. Since $b(i) = 1$, $s_{max}(i) = |\Gamma_0^+(i)| - 1$, that is $|S_0(i)| = |\Gamma_0^+(i)| - 1$ by lemma 1; so there exists a subtree with root in $\Gamma_0^+(i)$ that is not scheduled by π_i. We consider two cases:

   (a) if $\sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0$, each of the subtrees$A(j)$, $j \in \Gamma_0^+(i)$, is entirely scheduled by π_j^*.
   So we have $\sum_{j \in S_0^*(i)} |S^*(j)| = \sum_{j \in S_0^*(i)} |A(j)|$. If all of these subtrees are scheduled by π_i^* we get $s_{max}(i) = |\Gamma_0^+(i)|$, which is impossible. So there exists at least one of them that is not allocated to π_i^*, and since $c^*(i) = 1$, exactly one of them is not scheduled by π_i^*. Now, because $|S_0^*(i)| = |S_0(i)|$ we can use lemma 4 and get $\sum_{j \in S_0^*(i)} |A(j)| \geq \sum_{j \in S_0(i)} |S(j)|$.

   (b) If $\sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 1$, each of the subtrees $A(j)$, $j \in \Gamma_0^+(i)$, is entirely scheduled by π_j^*, except one. Let $S_0^{*'}(i)$ be the set of roots of these subtrees. Since $c^*(i) = 0$, all these roots are allocated to π_i^*.
   So we have $\sum_{j \in S_0^*(i)} |S^*(j)| \geq \sum_{j \in S_0^{*'}(i)} |S^*(j)| = \sum_{j \in S_0^{*'}(i)} |A(j)|$. Since $|S_0^{*'}(i)| = |S_0(i)|$ we just have to use lemma 4 to prove the result.

3. We have seen that task i has a 0-type immediate successor that is not allocated to π_i^*. Since either $c^*(i) = 0$ or $c^*(i) = 1$ the lemma is proved.

□

3.2.3 Characterization of some Successors of 3-type Nodes

We now prove that for some 3-type nodes with cost zero in σ^* but not in σ, there exists a successor j such that one of the following inequality is true:

1. $c^*(A(j)) \geq 1$
2. $c^*(j) + \sum_{k \in \Gamma_0^+(j) c^*(A(k))} \geq 1$
3. $c^*(j) \geq 1$

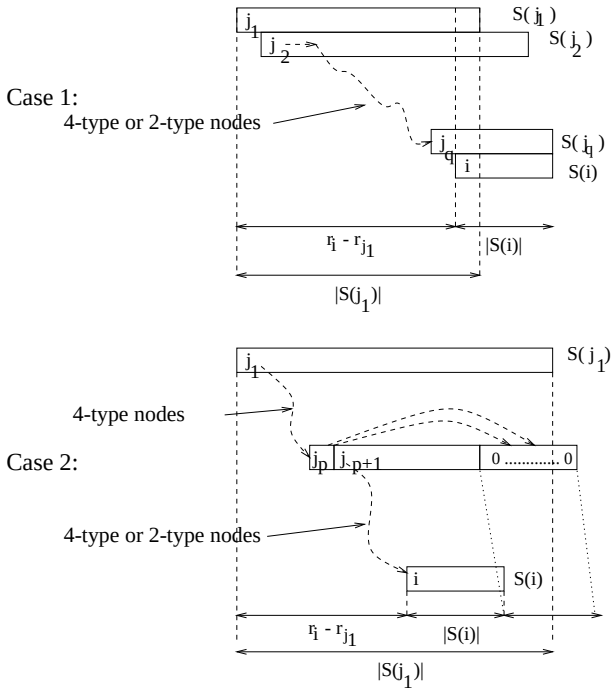

Figure 3: Case 1 and 2 of lemma 6

We first establish a central property that allows us to bound (in certain cases) the total number of successors of a task i that are allocated to π_i.

Lemma 6 *Let $l = (j_1, \ldots, j_q)$ be a path in A without nodes in $N_2 \cup N_3^3 \cup N_3^4 \cup N_4$. If i is an immediate successor of j_q then:*

$$|S(j_1)| \leq r_i - r_{j_1} + |S(i)| + \sum_{k \in l} \sum_{j \in S_0(k)} |S(j)|$$

Proof – Let L be the set of nodes in $l \cap (N_3^3 \cup N_3^4)$. We study two cases, according to the value $\alpha = |L|$:

1. $\alpha = 0$: (case 1 of Figure 3) the type of nodes in l is 2 or 4. Moreover, these nodes cannot have 0-type immediate successor by definition, so a node i in l has only one immediate successor allocated to π_i. We have then $|S(j_1)| \leq |S(j_2)| + 1$ (if it is not the case, there exists an immediate successor j of j_1 allocated to π_{j_1} such that $|S(j)| > |S(j_2)|$; so j_2 is executed by π_{j_1} and j is not, which is impossible). By repeating this reasoning we have $|S(j_1)| \leq |S(i)| + r_i - (r_i + 1)$. Since $\sum_{k \in l} \sum_{j \in S_0(k)} |S(j)| = 0$, we get $|S(j_1)| \leq r_i - r_{j_1} + |S(i)| + \sum_{k \in l} \sum_{j \in S_0(k)} |S(j)|$.

2. $\alpha \geq 1$: (case 2 of Figure 3) we prove the inequality pby induction. Let j_p be the first node of the path l that is in L. By definition of the algorithm, j_p has at most one non 0-type imediate successor not allocated to π_{j_p}.

   If there is one such immediate successor, either j_{p+1} is allocated to π_{j_p}, or there exists $j' \in \Gamma^+(j_p)$ such that $|S(j')| \leq |S(j_{p+1})|$. So we get, in the former case $|S(j_p)| = 1 + |S(j_{p+1})| + \sum_{j \in S_0(j_p)} |S(j)|$ and in the latter case $|S(j_p)| = 1 + |S(j')| + \sum_{j \in S_0(j_p)} |S(j)| \leq 1 + |S(j_{p+1})| + \sum_{j \in S_0(j_p)} |S(j)|$.

   If there is no such immediate successor, $|S(j_p)| = 1 + \sum_{j \in S_0(j_p)} |S(j)| \leq 1 + |S(j_{p+1})| + \sum_{j \in S_0(j_p)} |S(j)|$. In all cases we have $|S(j_p)| \leq 1 + |S(j_{p+1})| + \sum_{j \in S_0(j_p)} |S(j)|$.

   (a) $\alpha = 1$: the type of a node in the path from j_{p+1} to j_q is 4, so $|S(j_{p+1})| \leq r_i - r_{j_{p+1}} + |S(i)|$. Thus we have $|S(j_p)| \leq 1 + r_i - r_{j_{p+1}} + |S(i)| + \sum_{j \in S_0(j_p)} |S(j)| = r_i - r_{j_p} + |S(i)| + \sum_{j \in S_0(j_p)} |S(j)|$. If $j_p = j_1$ the inequality is proved; else the type of a node in l from j_1 toj_p, different from j_p, is 4. So $|S(j_1)| \leq r_{j_p} - r_{j_1} + |S(j_p)|$ and we get $|S(j_1)| \leq r_i - r_{j_1} + |S(i)| + \sum_{j \in S_0(j_p)} |S(j)|$.

   (b) $\alpha \geq 2$: suppose the inequality is true for $|L| = \alpha$ and consider the case $|L| = \alpha + 1$. Let $j_{p'}$ be the second node of the path l that is in L, and let $L' = L \backslash \{j_p\}$. By the hypothesis of induction, we have $|S(j_{p'})| \leq r_i - r_{j_{p'}} + |S(i)| + \sum_{k \in L'} \sum_{j \in S_0(k)} |S(j)|$. The type of a node in l from j_{p+1} to $j_{p'}$, different from $j_{p'}$, is 4, so $|S(j_{p+1})| \leq r_{j_{p'}} - r_{j_{p+1}} + |S(j_{p'})|$. Again, by the hypothesis of induction we get $|S(j_{p+1})| \leq r_{j_{p'}} - r_{j_{p+1}} + r_i - r_{j_{p'}} + |S(i)| + \sum_{k \in L'} \sum_{j \in S_0(k)} |S(j)|$, that is $|S(j_{p+1})| \leq r_i - r_{j_{p+1}} + |S(i)| + \sum_{k \in L'} \sum_{j \in S_0(k)} |S(j)|$. We have proved that $|S(j_p)| \leq 1 + |S(j_{p+1})| + \sum_{j \in S_0(j_p)} |S(j)|$, so $|S(j_p)| \leq 1 + r_i - r_{j_{p+1}} + |S(i)| + \sum_{k \in L'} \sum_{j \in S_0(k)} |S(j)| + \sum_{j \in S_0(j_p)} |S(j)| = r_i - r_{j_p} + |S(i)| + \sum_{k \in L} \sum_{j \in S_0(k)} |S(j)|$.
   If $j_p = j_1$ the inequality is true. Else, the type of a node in l from j_1 to j_p, different from j_p, is 4, so $|S(j_1)| \leq |S(j_p)| + r_{j_p} - r_{j_1}$.
   Finally, we have $|S(j_1)| \leq r_i - r_{j_1} + |S(i)| + \sum_{k \in L} \sum_{j \in S_0(k)} |S(j)|$.

The lemma is then true since the inequality is true in all cases.

□

To prove that some successors j are such that $c^*(A(j)) \geq 1$ we need an intermediate result.

Lemma 7 *If, for a 0-type or 1-type node i, $|S^*(i)| < |S(i)|$ then $c^*(A(i)) \geq b(i) + 1$.*

Proof – if the type of i is 0 then the lemma is obviously true. If the type of i is 1, suppose we have $|S^*(i)| < |S(i)|$. Let k_0 be the number of immediate successor of i scheduled by π_i. Let k_0^* and k_1^* be the number of immediate successors j of i scheduled by π_i^* such that $S(j) = A(j)$ and $S(j) \neq A(j)$, respectively. By definition we have $c^*(i) = |\Gamma^+(i)| - k_0^* - k_1^*$, and because the type of i is 1 we get $c^*(i) = |\Gamma_0^+(i)| - k_0^* - k_1^*$. Each of the k_1^* subtrees create at least one communication, so $c^*(A(i)) \geq c^*(i) + k_1^* = |\Gamma_0^+(i)| - k_0^*$. But $k_0^* \leq s_{max}(i) - 1 = |S_0(i)| - 1$, by the case 1 of property 1 (we have $|S^*(i)| \geq |S(i)|$ if it is not the case, which is impossible). Hence, we conclude that $c^*(A(i)) \geq |\Gamma_0^+(i)| - |S_0(i)| + 1 = b(i) + 1$.

□

So by this result and lemma 6 we obtain:

Lemma 8 *Let i be in $N_3^1 \cup N_3^2$. If i has no successor in $N_3^1 \cup N_3^2 \cup N_3^5 \cup N_3^6$, there exists a 1-type node $i' \in A(i)$ such that $c^*(A(i')) \geq b(i') + 1$.*

Proof – Since i has no successor in $N_3^1 \cup N_3^2 \cup N_3^5 \cup N_3^6$, non 0-type and non 1-type successors of i are in $N_2 \cup N_3^3 \cup N_3^4 \cup N_4$: each of these successors has at least one of its non 0-type immediate successor scheduled by its processor, in σ and in σ^*, by definition of the partition and by lemma 5. So there exists a 1-type successor i' of i allocated to π_i^*. Let $l = (i, j_1, \ldots, j_q, i')$ the path from i to i' in A, and let $l' = (j_1, \ldots, j_q)$. We have then

$$|S^*(j_1)| \geq r_{i'} - r_{j_1} + |S^*(i')| + \sum_{k \in l'} \sum_{j \in S_0^*(k)} |S^*(j)|$$

Since $i \in N_3^1 \cup N_3^2$, i has no immediate successor (with type different from 0) allocated to π_i, but all of its immediate successors (with type different from 0) are allocated to π_i^*. So $\pi_{j_1} \neq \pi_i$ and $\pi_{j_1}^* = \pi_i^*$. In the same way, since $i \in N_3^1 \cup N_3^2$ we can use case 2 of lemma 3 and we get $\sum_{j \in S_0^*(i)} |S^*(j)| = \sum_{j \in S_0(i)} |S(j)|$. If $|S(j_1)| \leq |S^*(j_1)|$, we have $t_{j_1} + 1 + |S(j_1)| + \sum_{j \in S_0(i)} |S(j)| = r_{j_1} + 1 + |S(j_1)| + \sum_{j \in S_0^*(i)} |S^*(j)| \leq t_{j_1}^* + 1 + |S^*(j_1)| + \sum_{j \in S_0^*(i)} |S^*(j)| \leq T$ and j_1 should have been allocated to π_i (by definition of the algorithm). So we have

$$|S(j_1)| > |S^*(j_1)|$$

Consider now the two following cases:

1. $\pi_{i'} = \pi_{j_1}$: we have $|S(j_1)| = r_{i'} - r_{j_1} + |S(i')| + \sum_{j \in S_0(j_1)} |S(j)|$. Since $|S(j_1)| > |S^*(j_1)|$, we get $|S(i')| > |S^*(i')| + \sum_{k \in l'} \sum_{j \in S_0^*(k)} |S^*(j)| - \sum_{j \in S_0(j_1)} |S(j)|$, so $|S(i')| > |S^*(i')| + \sum_{j \in S_0^*(j_1)} |S^*(j)| - \sum_{j \in S_0(j_1)} |S(j)|$. If $j_1 \in N_2 \cup N_4$, j_1 has no 0-type immediate successor, so $|S(i')| >$

$|S^*(i')|$. Else, $j \in N_3^3 \cup N_3^4$ so we can use lemmas 3 and 5 (case 2) and get $|S(i')| > |S^*(i')|$.

Then, by lemma 7, the result is proved.

2. $\pi_{i'} \neq \pi_{j_1}$: if we use lemma 6 with the path $l' = (j_1, \ldots, j_q)$, we get $|S(j_1)| \leq r_{i'} - r_{j_1} + |S(i')| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)|$. Again, by lemmas 3 and 5, we have $|S(i')| > |S^*(i')|$.

So the lemma is true.

□

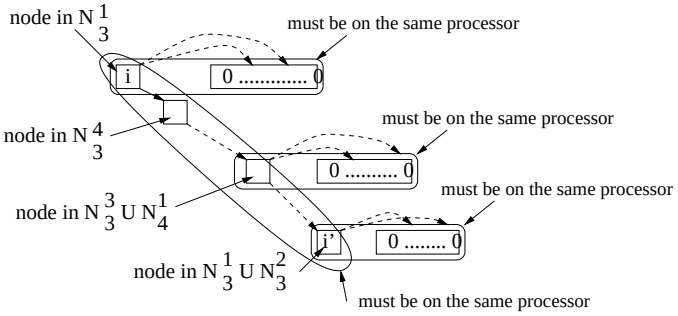

Figure 4: a contradictory argument for lemma 9

Now, we prove the existence of the other kind of successors. We say that j is an *immediate descendant* of i, if $j \in N_3^1 \cup N_3^2$ and there exists no other node that belongs to $N_3^1 \cup N_2^3$ in the path from i to j.

Lemma 9 *Let i an element of N_3^1. If i' is an immediate descendant of i then there exists a node in the path l from i to i' that belongs to $N_3^5 \cup N_3^6 \cup N_4^2$.*

Proof – Since $i \in N_3^1$ we have $c(i) = |\Gamma_0^+(i)|$ and $b(i) = 0$. So $S_0(i) = \Gamma_0^+(i)$ and there exists no immediate successor of i (with type different from 0) allocated to π_i. Hence, $|S(i)| = 1 + \sum_{j \in S_0(i)} |S(j)|$. Because $b(i') = 0$ and $c(i') = |\Gamma_0^+(i')|$ we can use the same reasoning for node i'. Thus we get:

$$|S(i)| + |S(i')| = 2 + \sum_{j \in S_0(i)} |S(j)| + \sum_{j \in S_0(i')} |S(j)| \qquad (1)$$

If i' is an immediate successor of i then $\pi_{i'}^* = \pi_i^*$ (since $i \in N_3^1$, we have $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 0$). So $|S^*(i)| \geq 2 + \sum_{j \in S_0^*(i)} |S^*(j)| + \sum_{j \in S_0^*(i')} |S^*(j)|$. Because $i' \in N_3^1 \cup N_3^2$, we can use case 2 of lemma 3 and get $|S^*(i)| \geq 2 + \sum_{j \in S_0(i)} |S(j)| + \sum_{j \in S_0(i')} |S(j)|$. By (1) we have

$|S(i)| + |S(i')| \leq |S^*(i)|$ and i' should have been scheduled by π_i: so there exists a node in the path from i to i' different from i and from i'.

If $l = (i, j_1, \ldots, j_q, i')$, let $l' = (j_1, \ldots, j_q)$. We prove by contradiction that not all nodes in l' belong to $N_3^3 \cup N_3^4 \cup N_4^1$. Suppose that all nodes in l' belong to à $N_3^3 \cup N_3^4 \cup N_4^1$ (see figure 4). We have two cases:

1. $\pi_i^* \neq \pi_{i'}^*$: Let j be a node in l. If j belongs to $N_3^1 \cup N_3^2 \cup N_3^3 \cup N_4^1$ then $c^*(j) = 0$ (by definition) and all immediate successors of j are allocated to π_j^*. Else j belongs to N_3^4, and by lemma 5, all immediate successors of j (with type different from 0) are scheduled by π_j^*.

   So we have $\pi_{j_1}^* = \pi_{j_2}^* = \cdots = \pi_{j_q}^* = \pi_{i'}^*$, and since $c^*(i) = 0$ we get $\pi_i^* = \pi_{j_1}^*$: this case is not possible.

2. $\pi_i^* = \pi_{i'}^*$: all nodes in l are scheduled by π_i^*, so $|S^*(i)| \geq 1 + r_{i'} - r_i + \sum_{j \in S_0^*(i)} |S^*(j)| + \sum_{j \in S_0^*(i')} |S^*(j)| + \sum_{k \in l'} \sum_{j \in S_0^*(k)} |S^*(j)|$. By case 2 of lemmas 3 and 5 we get :

$$|S^*(i)| \geq 1 + r_{i'} - r_i + \sum_{j \in S_0(i)} |S(j)| + \sum_{j \in S_0(i')} |S(j)| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)| \quad (2)$$

   Since all nodes in l' belong to à $N_3^3 \cup N_3^4 \cup N_4^1$, we can use lemma 6 with the path l' and get $|S(j_1)| \leq r_{i'} - r_{j_1} + |S(i')| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)|$. Then, $|S(i)| + |S(j_1)| \leq 1 + \sum_{j \in S_0(i)} |S(j)| + r_{i'} - r_{j_1} + |S(i')| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)|$. Because $r_{j_1} = r_i + 1$ we have $|S(i)| + |S(j_1)| \leq r_{i'} - r_i + \sum_{j \in S_0(i)} |S(j)| + |S(i')| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)|$. Since $|S(i')| = 1 + \sum_{j \in S_0(i')} |S(j)|$, we get $|S(i)| + |S(j_1)| \leq 1 + r_{i'} - r_i + \sum_{j \in S_0(i)} |S(j)| + \sum_{j \in S_0(i')} |S(j)| + \sum_{k \in l'} \sum_{j \in S_0(k)} |S(j)|$, that is $|S(i)| + |S(j_1)| \leq |S^*(i)|$ by inequality 2.

   We have then $t_i + |S(i)| + |S(j_1)| = r_i + |S(i)| + |S(j_1)| \leq t_i^* + |S(i)| + |S(j_1)| \leq t_i^* + |S^*(i)| \leq T$, so j_1 should have been scheduled by π_i: this case is not possible.

In the two cases, there is a contradiction with the hypothesis that all nodes in l' belong to $N_3^3 \cup N_3^4 \cup N_4^1$: the lemma is then true.

□

Since by definition we have $c^*(j) \geq 1$ for $j \in N_4^2$ and $c^*(j) + \sum_{k \in \Gamma_0^+(j)} c^*(A(k)) \geq 1$ for $j \in N_3^5 \cup N_3^6$, the existence of the successors is proved and we can now establish the main result.

Theorem 3 *The performance guarantee of the algorithm is 2.*

Proof – Let N be the set of nodes in A. We denote by N_i, $i \in \{0,1,2,3,4\}$ the set of i-type nodes, by R_1 and R_2 (respectively) the sets of roots of 1-type and 2-type subtrees.

We have:

$$\sum_{i\in N_3\cup N_4} c(i) = \sum_{i\in N_3^1\cup N_3^2\cup N_3^5} c(i) + \sum_{i\in N_3^3} c(i) + \sum_{i\in N_3^4} c(i) + \sum_{i\in N_3^6\cup N_4} c(i)$$

By definition of the partition of N_3:

$$\sum_{i\in N_3\cup N_4} c(i) \leq \sum_{i\in N_3^1\cup N_3^2\cup N_3^5} |\Gamma_0^+(i)| + \sum_{i\in N_3^3} (|\Gamma_0^+(i)|-1) + \sum_{i\in N_3^4} (|\Gamma_0^+(i)|+1) + \sum_{i\in N_3^6\cup N_4} c(i)$$

Since, for$i \in N_4$, $c(i) = |\Gamma^+(i)| - 1 = |\Gamma_0^+| - 1$ and since $c(i) \leq b(i) + |\Gamma_0^+(i)|$ (case 2 of lemma 1), we have:

$$\sum_{i\in N_3\cup N_4} c(i) \leq \sum_{i\in N_3\cup N_4} |\Gamma_0^+(i)| - |N_3^3| + |N_3^4| - |N_4| + \sum_{i\in N_3^6} b(i)$$

Now, let us consider an immediate successor j of a node in $N_3 \cup N_4$. By definition of the classification, either j belongs to $N_3 \cup N_4$, or is the root of a 1-type (or 2-type) subtree (see Figure 1). So we have :

$$\sum_{i\in N_3\cup N_4} c(i) \leq (|N_3| + |N_4| + |R_1| + |R_2|) - |N_3^3| + |N_3^4| - |N_4| + \sum_{i\in N_3^6} b(i)$$

that is

$$\sum_{i\in N_3\cup N_4} c(i) \leq |N_3^1| + |N_3^2| + 2|N_3^4| + |N_3^5| + |N_6^3| + |R_1| + |R_2| + \sum_{i\in N_3^6} b(i) \quad (3)$$

Consider now the nodes in N_3^1. Let $N_3^{1'}$ be the set of nodes in N_3^1 that have, either an immediate descendant, or no immediate descendant but one successor in $N_3^5 \cup N_3^6 \cup N_4^2$. Let $N_3^{1''} = N_3^1 \backslash N_3^{1'}$. By lemma 9 (for nodes in $N_3^{1'}$) we get

$$|N_3^{1'}| \leq |N_3^5| + |N_3^6| + |N_4^2|$$

and by (3) we have

$$\sum_{i\in N_3\cup N_4} c(i) \leq |N_3^{1''}| + |N_3^2| + 2|N_3^4| + 2|N_3^5| + 2|N_6^3| + |N_4^2| + |R_1| + |R_2| + \sum_{i\in N_3^6} b(i) \quad (4)$$

For a node i in $N_1 \cup N_2$, if its type is 1 then $c(i) = b(i)$ else its type is 2 and $c(i) = |\Gamma^+(i)| - 1$. So we have

$$\sum_{i \in N_1 \cup N_2} c(i) = \sum_{i \in N_1} b(i) + \sum_{i \in N_2} |\Gamma^+(i)| - |N_2|$$

By definition, nodes in $N_3^{1''}$ have neither immediate descendant, nor successor in $N_3^5 \cup N_3^6$. In other words, no successor of a node in $N_3^{1''}$ belongs to $N_3^1 \cup N_3^2 \cup N_3^5 \cup N_3^6$. So we can use lemma 8. It also true for nodes in N_3^2 since the have no successor in N_3 by definition. So it is possible to associate to each node in $N_3^{1''} \cup N_3^2$ a 1-type subtree $A(i)$ such that $b(i) \leq c^*(A(i)) - 1$.

Now, because $N_3^{1''} \cup N_3^2$ cannot have, by definition, a successor that belongs to $N_3^{1''} \cup N_3^2$, all these subtrees are different and we get

$$\sum_{i \in N_1 \cup N_2} c(i) \leq \sum_{i \in N_1} c^*(A(i)) - |N_3^2| - |N_3^{1''}| + \sum_{i \in N_2} |\Gamma^+(i)| - |N_2|$$

The type of an immediate successor of a 2-type node is 1 or 2, by definition of the classification. so it cannot belong to R_1 or R_2 (the immediate predecessor of a node in $R_1 \cup R_2$ is either a 3-type node or a 4-type node). Hence we have

$$\sum_{i \in N_1 \cup N_2} c(i) \leq \sum_{i \in N_1} c^*(A(i)) - |N_3^2| - |N_3^{1''}| + (|N_1| - |R_1| + |N_2| - |R_2|) - |N_2| \tag{5}$$

Since the cost of a 0-type node is 0 in σ, we have $c = \sum_{i \in N_1 \cup N_2} c(i) + \sum_{i \in N_3 \cup N_4} c(i)$, that is by adding (4) to (5)

$$c \leq \sum_{i \in N_1} c^*(A(i)) + |N_1| + 2|N_3^4| + 2|N_3^5| + 2|N_3^6| + |N_4^2| + \sum_{i \in N_3^6} b(i)$$

By definition, we have: $c^*(A(i)) \geq 1$ if $i \in N_1$; $c^*(i) \geq 1$ if $i \in N_4^2$; $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) = 1$ if $i \in N_3^4 \cup N_3^5$; and $c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j)) \geq 2$ if $i \in N_3^6$. Thus we have:

1. $|N_1| + |N_4^2| \leq \sum_{i \in N_1} c^*(A(i)) + \sum_{i \in N_4^2} c^*(i)$
2. $2|N_3^4| + 2|N_3^5| = 2 \sum_{i \in N_3^4 \cup N_3^5} \{c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))\}$
3. $2|N_3^6| \leq \sum_{i \in N_3^6} \{c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))\}$

and since $\sum_{i \in N_3^6} b(i) \leq \sum_{i \in N_3^6} \{c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))\}$ we get

$$c \leq 2 \sum_{i \in N_1} c^*(A(i)) + \sum_{i \in N_4^2} c^*(i) + \quad 2 \sum_{i \in N_3^4 \cup N_3^5 \cup N_3^6} \{c^*(i) + \sum_{j \in \Gamma_0^+(i)} c^*(A(j))\}$$

that is

$$c \leq 2c^*$$

□

3.2.4 The Tightness of the Bound

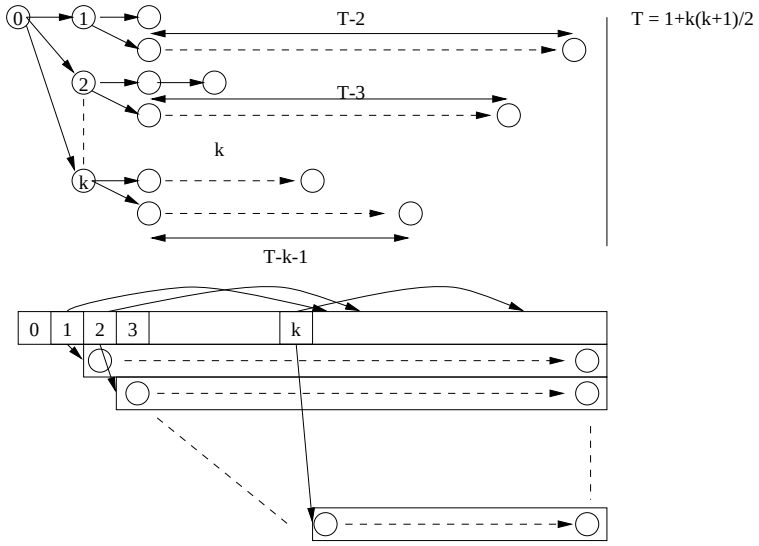

Figure 5: An instance for which the bound is tight

We show that the bound is asymptotically tight. Suppose we have the 2-type subtree of Figure 5: in a schedule built by the algorithm none of th subtrees with root 1, 2, ..., k can be scheduled by only one processor. Hence we have $k-1$ communications from the main root and k communications from the roots of the k subtrees, that is $c = 2k-1$. In an optimum schedule, there are only k communications. So we have $\frac{c}{c^*} \rightarrow 2$ when k tends to the infinity.

4 Approximation and Polynomial Results in the UET-UCT-bus Model

In this Section we are concerned with the case of a single machine bus. Contrary to the previous model, we have now communications delays.

4.1 A Polynomial Case

In [1] Afrati, Papadimitriou and Papageorgiou have proposed a polynomial algorithm that schedules a precedence graph with unit execution time on m processors in at most T unit of time and at most C communications, if C is fixed.

We slightly modify their algorithm to solve the following problem.

m-MINTIME-BUS-B-T:

Instance: a UET-UCT precedence graph G with n tasks, single machine bus R with m processors and a capacity B, and an integer T.

Goal: Schedule G on R such that the makespan is at most T.

Theorem 4 *The problem m-MINTIME–BUS-B-T can be solved in time*

$$\mathcal{O}(D^{D+1}n^{2D+1}\log n)$$

where $D = B(T-2)$.

Proof – When T is fixed there are at most $B(T-2)$ communications. So we can use the enumerative method described in [1]. Let V be the set of n nodes in G, and E the set of arcs of G. The maximum number of $B(T-2)$-tuples is bounded by $|E|^{B(T-2)}$. If we delete each of these $B(T-2)$-tuples we get at most $c = B(T-2)+1$ weakly connected components. Since each of theses components has to be assigned to a single processor, there are $\min(m,c)$ possible assignments. Then, each component must be scheduled after the communications of its inner arcs and before its outer arcs, so these arcs define release times and deadlines. It is then possible to schedule each component by solving a single-processor scheduling problem in time $\mathcal{O}(n\log n)$ (see [7]). Since the total number of these arcs is at most $B^c(T-2)^c$, we can examine all possible solution in time $\mathcal{O}(|E|^{B(T-2)}\min(m,c)B^c(T-2)^c n\log n)$. By bounding $|E|$ by n^2, and $\min(m,c+1)$ by n, we get a time $\mathcal{O}(B(T-2)^{B(T-2)+1}n^{2B(T-2)+1}\log n)$.

□

4.2 An Approximation Algorithm

Now, we consider the problem MINTIME-BUS in which we are asked to find a minimum makespan schedule of a UET-UCT tree on an infinite number of processors. We suppose that the capacity B is arbitrary.

First, we show that a list algorithm can have a very bad performance. We consider the instance shown on Figure 6. Suppose $B = 1$ and k is even: an optimum schedule if of length $2k+1$.

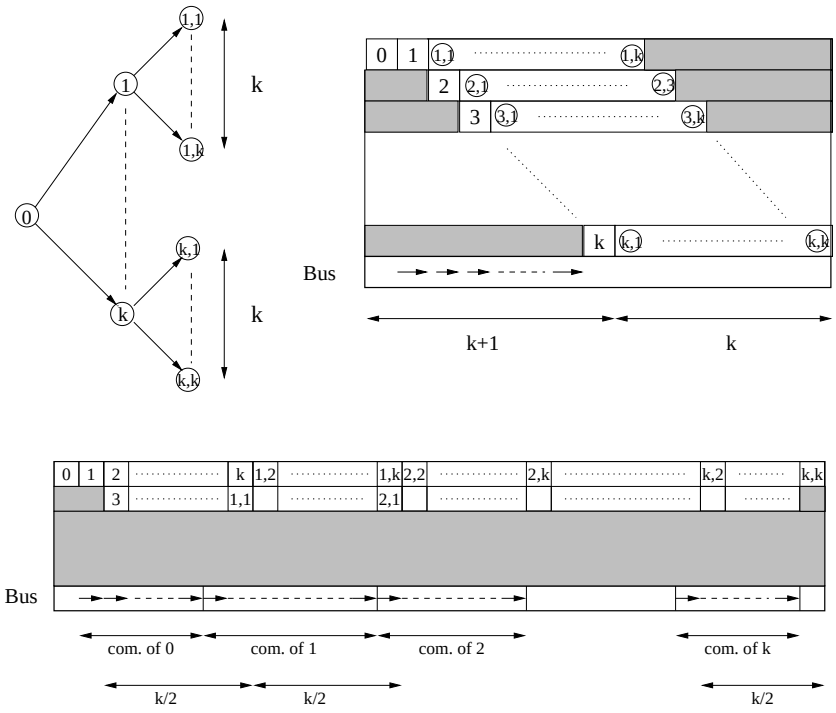

Figure 6: A bad instance for a list algorithm

If the priority list is $(0, 1, \ldots, k, (1,1), \ldots, (1,k), \ldots, (k,1), \ldots, (k,k))$, a list algorithm provides a schedule of length $2 + \frac{k^2}{2}$. So the performance ratio is not bounded and can be as bad as we want.

Now, if we use the algorithm presented in Section 3 and then algorithm $Algo_2$ (see page 3) we can get a solution of MINTIME-BUS, and we have the following approximation theorem.

Theorem 5 *MINTIME-BUS can be approximated within a ratio of 6.*

Proof – this a consequence of theorems 1 and 3.

□

5 Simulations

We have proposed an approximation algorithm (from now we call it MINCOM) for the problem MINTIME-BUS and we have analyzed its worst-case to get a performance guarantee. In order to analyze the performance in case of a random instance, we did some simulations on a computer.

5.1 The Method

To get an idea of the quality of the performance we compared our algorithm with four heuristics.

The first one is based on solving a relaxed problem. Indeed, if we do not consider the limitation constraint we get the model UET-UCT and the problem is polynomial (see Chrétienne [3]). So, to get a solution for MINTIME-BUS, we just have to use step 3 of the algorithm $Algo_2$ to schedule the communications.

The three others are list algorithms. We tested several priority list but three ones seem to have a better performance than the others: (1) nodes sorted according to the non increasing height, (2) nodes sorted according to the non increasing width (the number of leaf of a subtree), and (3) nodes sorted according to the non increasing ratio width/height.

To compare these five algorithms, we computed the ratio $\frac{\omega(H)}{\omega^-}$, where $\omega(H)$ is the makespan of a heuristic H and ω^- the makespan of a solution by the algorithm of Chrétienne. Since this algorithm solves a relaxed problem, ω^- is a lower bound of an optimum makespan ω^* for MINTIME-BUS. Hence, we have $\frac{\omega(M)}{\omega^*} \leq \frac{\omega(M)}{\omega^-}$ and, by this way, we approximate the real performance ratio.

To generate the trees we used the algorithm described in [2]. The interesting point of this method is that the trees are randomly and uniformly generated. We considered 20 sizes for the trees (from 10 to 485, with a step equal to 25), and for each size we generated 100 trees (results are stable from this value). To each tree we associated 3 values of the capacity (1, 5 and 10), that is we considered 6000 instances of the problem. We used a 200 MHz Pentium and it took about 2 hours and a half for the tests to be done.

5.2 The Results

The aim of the tests was not only to compare our algorithm MINCOM with others, but also to study the correlation between the performance ratio and some parameters of the trees: the number of nodes, the height, the width and the ratio width/height. In the following we present two kind of plots: (1) the mean performance ratio and (2) the scatter plot (the individual performance ratios) as a function of one of the parameters. By analyzing the scatter plots we can get an idea of a possible correlation and of its significance. Indeed, if the mean performance varies with a parameter (the number of nodes for instance) and if the variability does not depend on the parameter, it is almost sure that there exists a correlation (if the sample has a sufficient number of items, obviously) (see [11] for instance or an other introductory book to statistics).

5.2.1 The Mean Performance as a Function of the Number of Nodes

We first observe that the mean performance ratio of our algorithm does not depend on the bus capacity, and is equal to around 1.25. The mean performance ratio of the four other algorithms degrades with the increase of the number of nodes, except for the case of a capacity 10 where it is hard to conclude. However the performance of these algorithm gets better with the increase of the bus capacity. Finally, we can note that the method based on Chrétienne's algorithm is very efficient for the case of a capacity 10 (the optimum seems to be reached many times). This is not a surprise since if the capacity is great the model UET-UCT-bus is close to the model UET-UCT, in a sense.

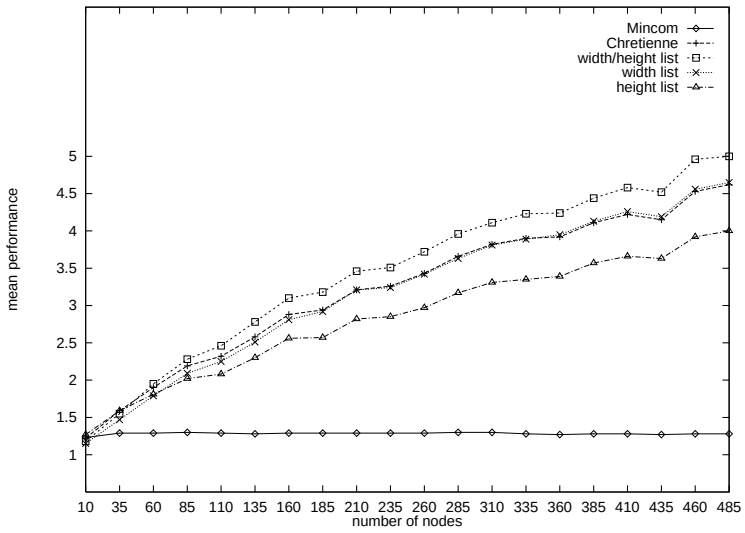

Figure 7: Capacity 1

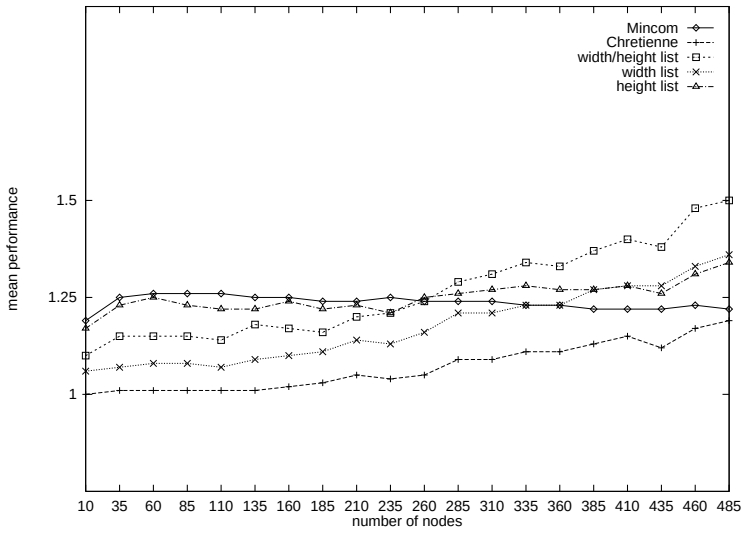

Figure 8: Capacity 5

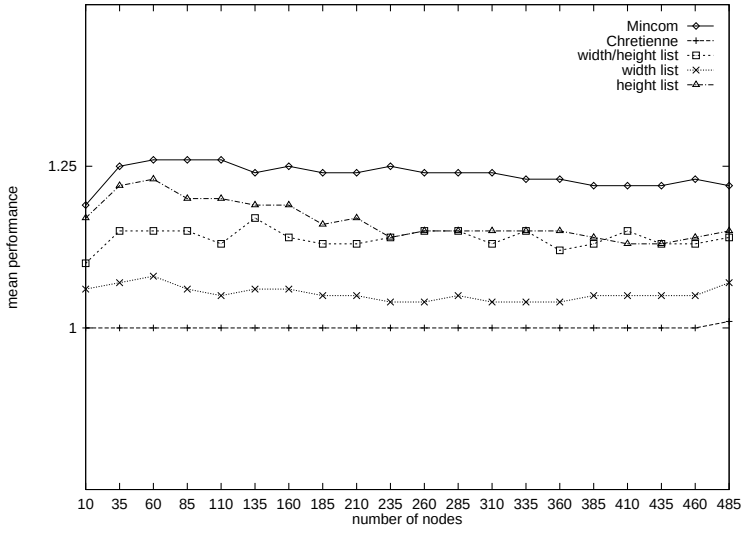

Figure 9: Capacity 10

5.2.2 The Mean Performance Ratio as a Function of Width/Height

We only give the results for the width/height parameter because they are the most significants. We can make the same observation as previously, except maybe for the case of a capacity 10 where the performance of all algorithms seems to deteriorate. However, as we shall see in the following, the last performance values are not significant because they are associated to few data.

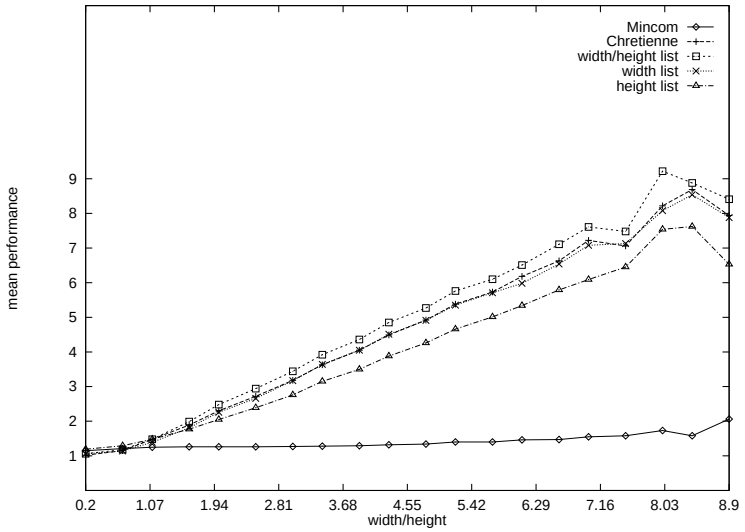

Figure 10: Capacity 1

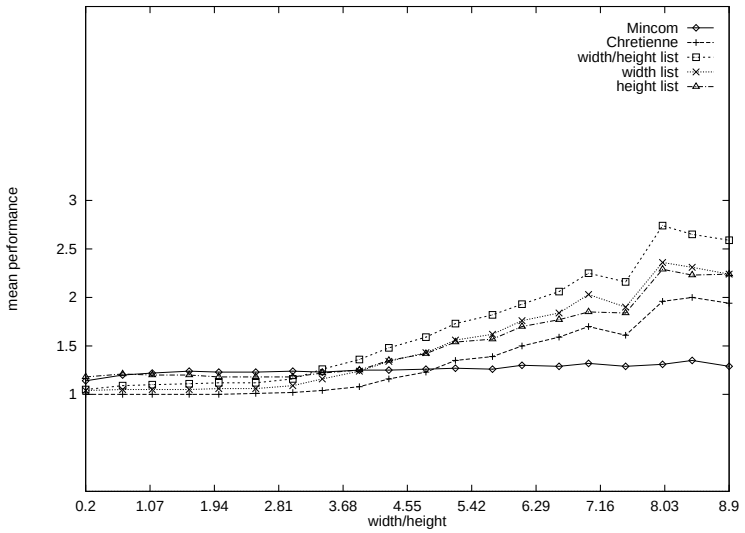

Figure 11: Capacity 5

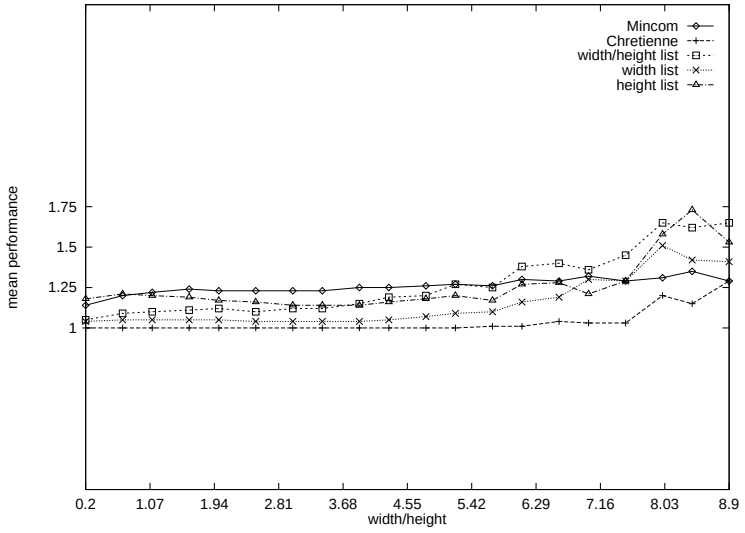

Figure 12: Capacity 10

5.2.3 Scatter Plot as a Function of the Number of Nodes

Now, we study the correlation between the performance ratio and the number of nodes of the trees. Only the case of a capacity 1 is presented, since it is the most significant.

We first observe that the performance of MINCOM does not depend on the number of nodes. On the contrary, there exists a correlation for the other algorithms, but we can not assert that this correlation is important or that the performance depends only on the size of the trees (the variability is not independent from this parameter).

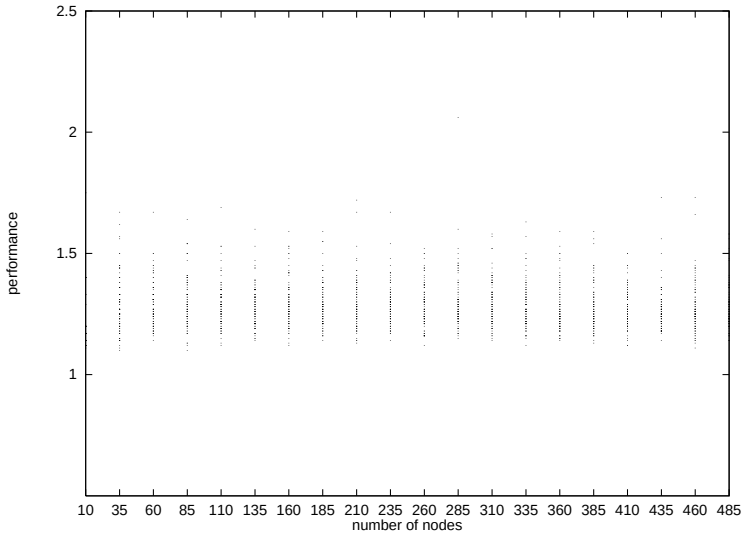

Figure 13: MINCOM (capacity 1)

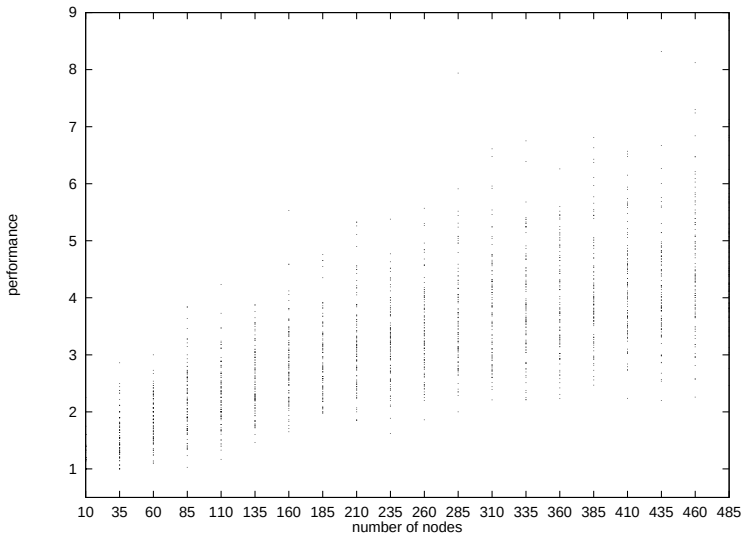

Figure 14: Heuristic based on Chrétienne's algorithm (capacity 1)

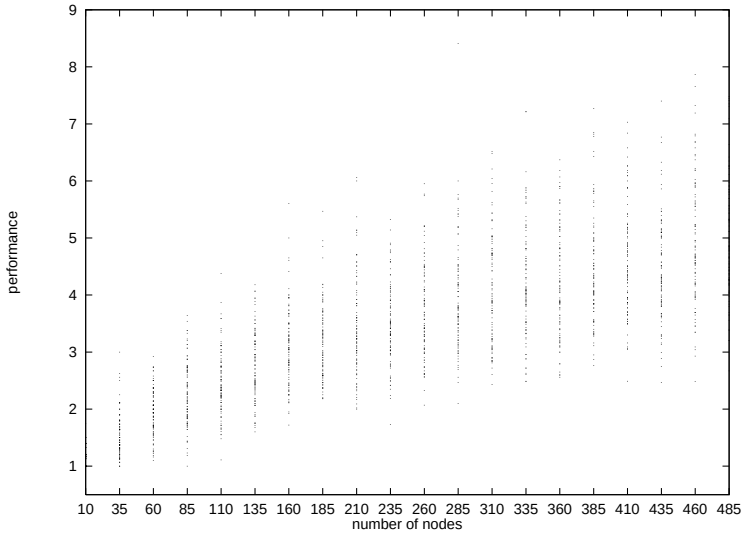

Figure 15: List algorithm according to the ratio $\frac{width}{height}$ of the tree (capacity 1)

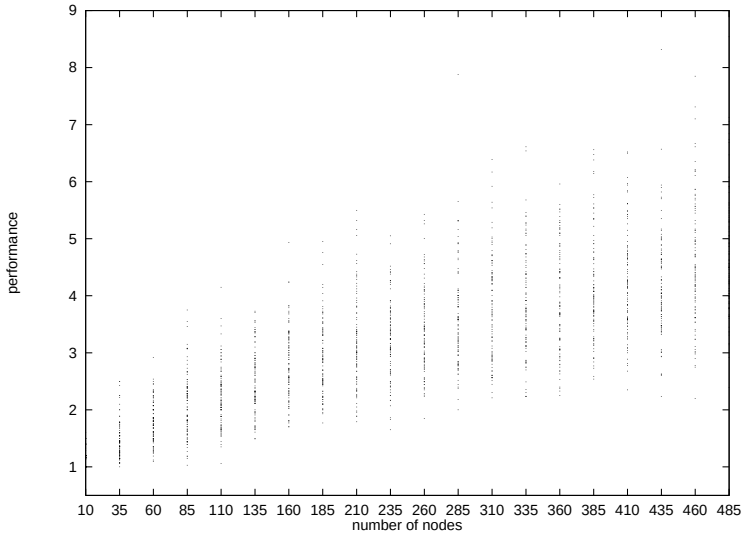

Figure 16: List Algorithm according to the width of the tree (capacity 1)

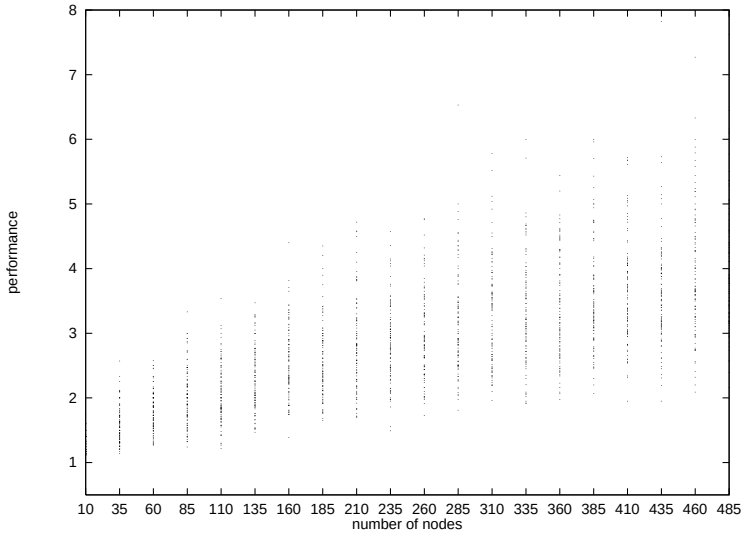

Figure 17: List algorithm according to the height of the tree (capacity 1)

5.2.4 Scatter Plot as a Function of the Ratio Width/Height

We are now concerned with the correlation between the performance ratio and the parameter width/height. We can make the following observations:

1. The performance of MINCOM does not seem to depend on the ratio width/height.

2. If capacity is 1, the correlation is linear and important for the four other algorithms.

3. If the capacity is 5, there exists a threshold from which there is a correlation. This correlation is important, except for the list algorithm according to the height.

4. If the capacity is 10, it seems that there is no correlation, except may be for the list algorithm according to the width.

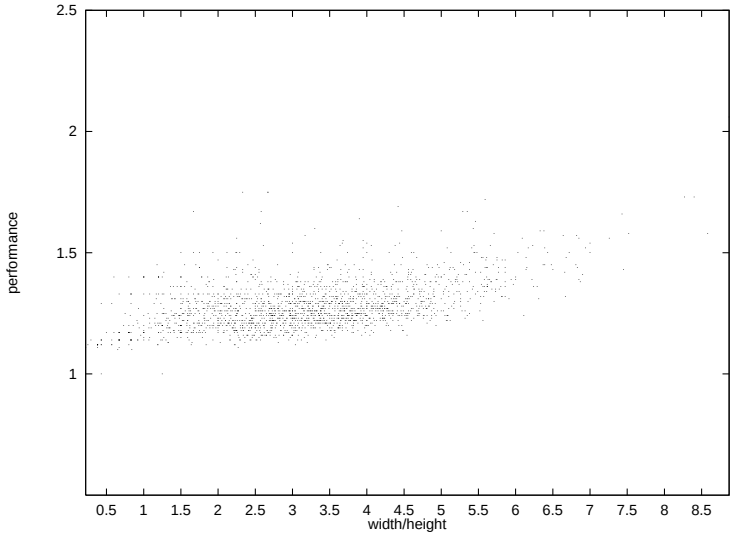

Figure 18: MINCOM (capacity 1)

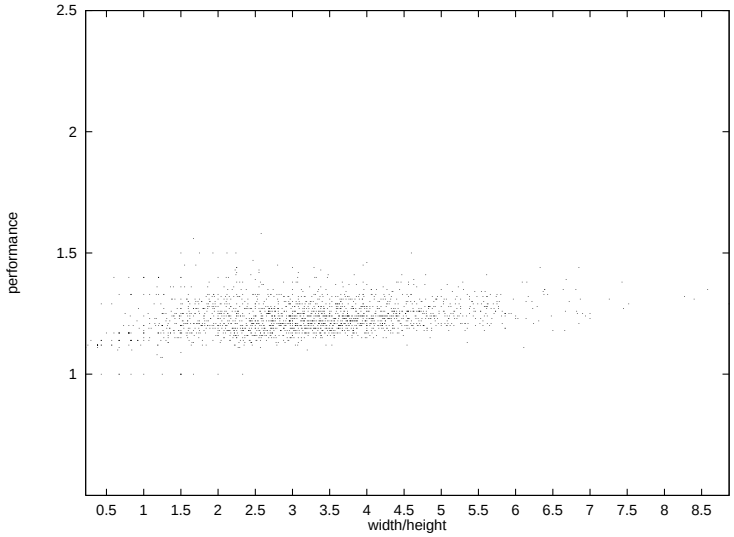

Figure 19: MINCOM (capacity 5)

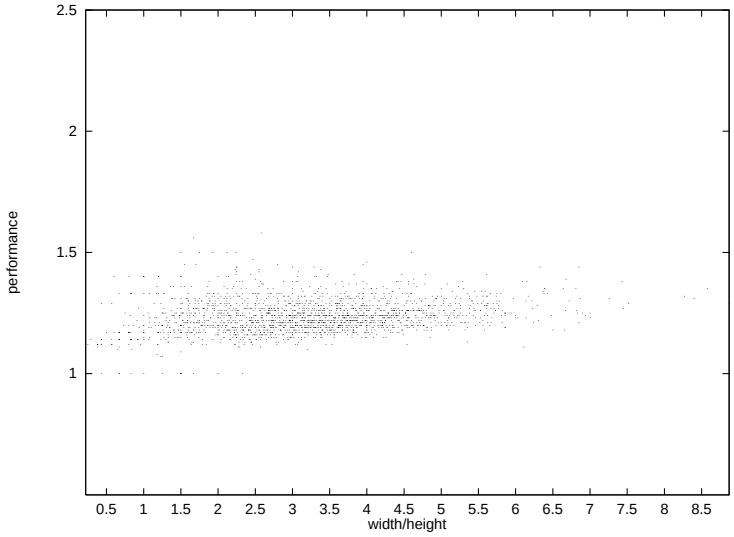

Figure 20: MINCOM (capacity 10)

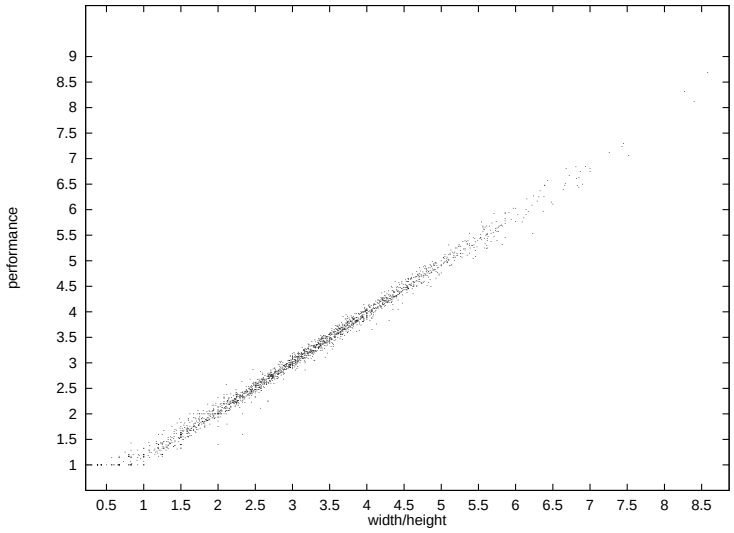

Figure 21: Heuristic based on Chrétienne's algorithm (capacity 1)

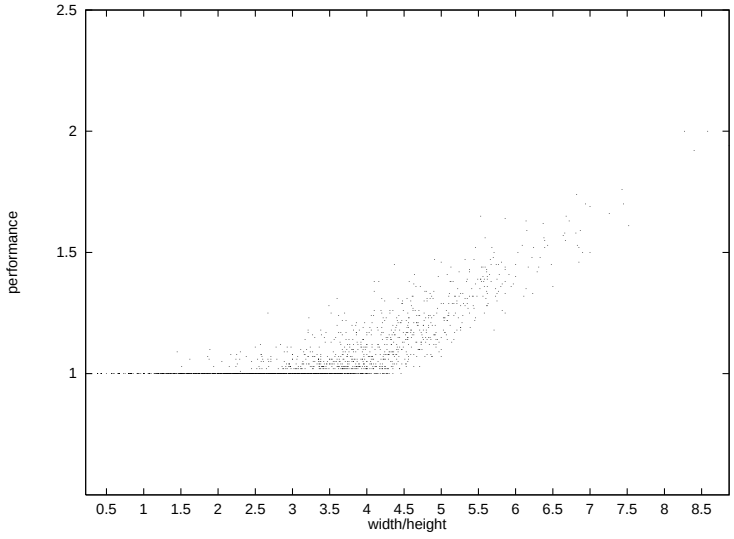

Figure 22: Heuristic based on Chrétienne's algorithm (capacity 5)

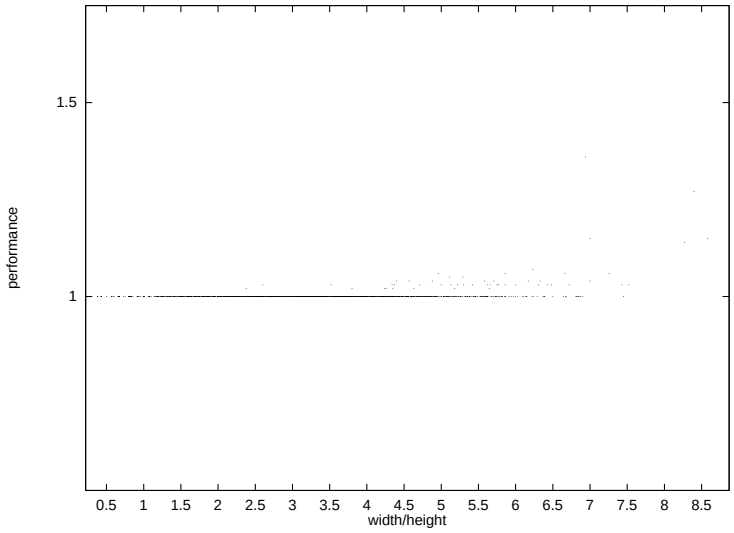

Figure 23: Heuristic based on Chrétienne's algorithm (capacity 10)

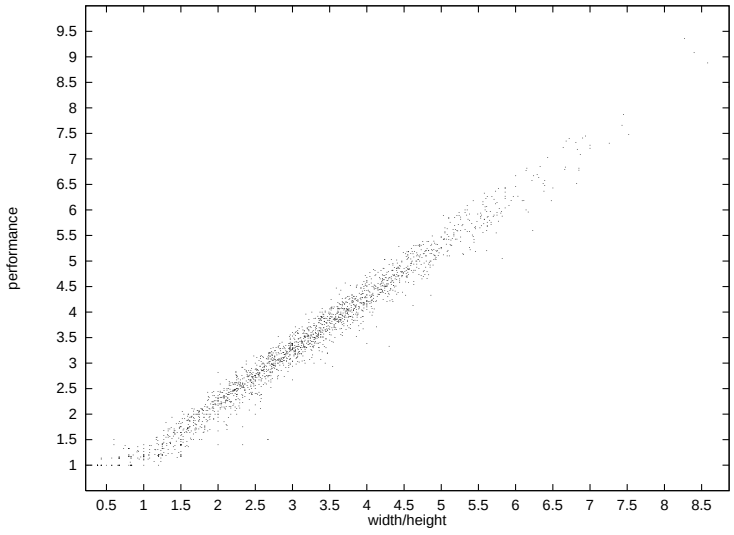

Figure 24: List algorithm according to the ratio $\frac{width}{height}$ of the tree (capacity 1)

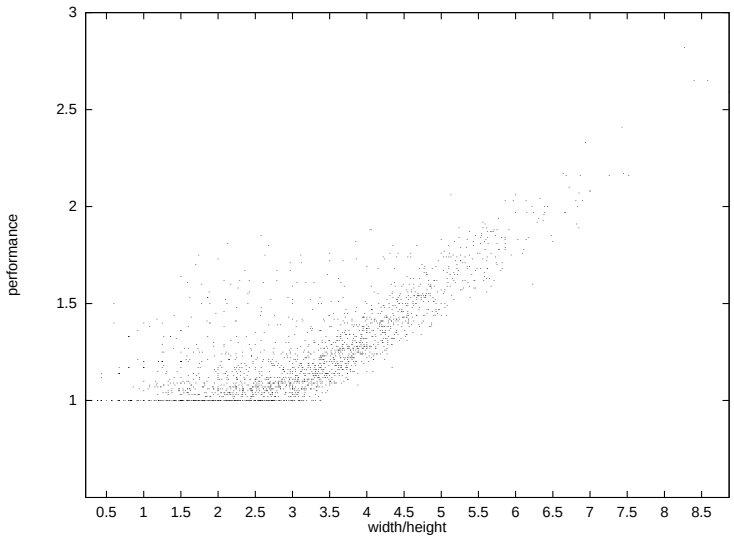

Figure 25: List algorithm according to the ratio $\frac{width}{height}$ of the tree (capacity 5)

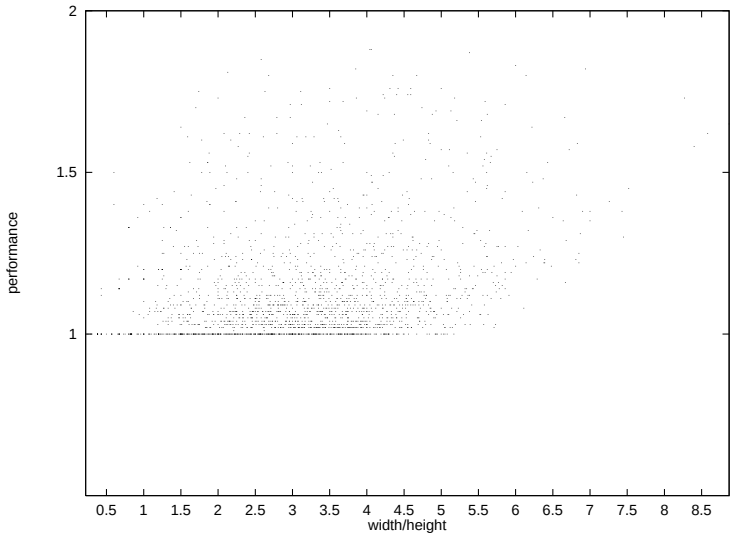

Figure 26: List algorithm according to the ratio $\frac{width}{height}$ of the tree (capacity 10)

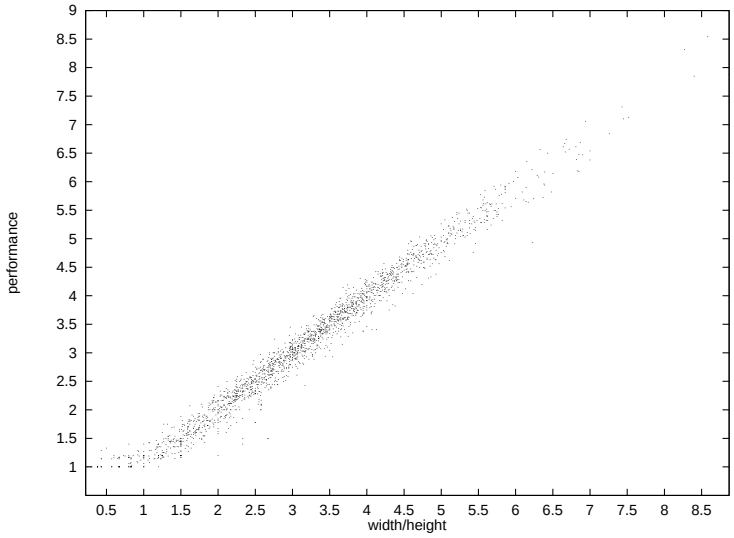

Figure 27: List algorithm according to the width of the tree (capacity 1)

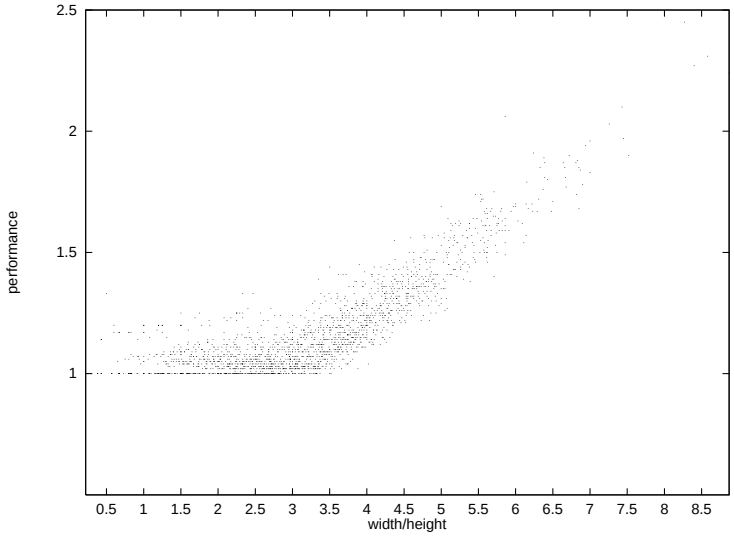

Figure 28: List algorithm according to the width of the tree (capacity 5)

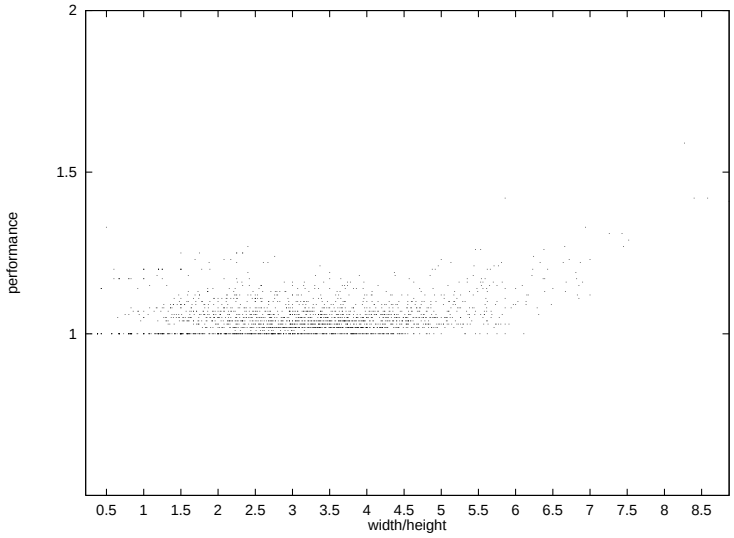

Figure 29: List algorithm according to the width of the tree (capacity 10)

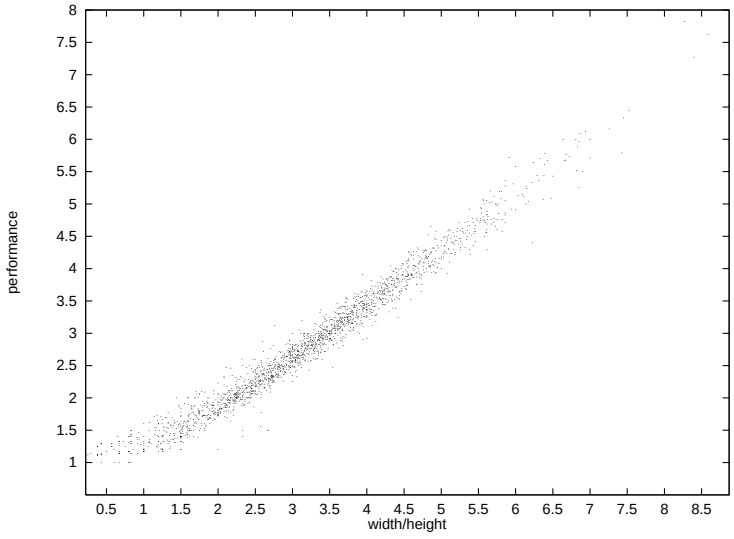

Figure 30: List algorithm according to the height of the tree (capacity 1)

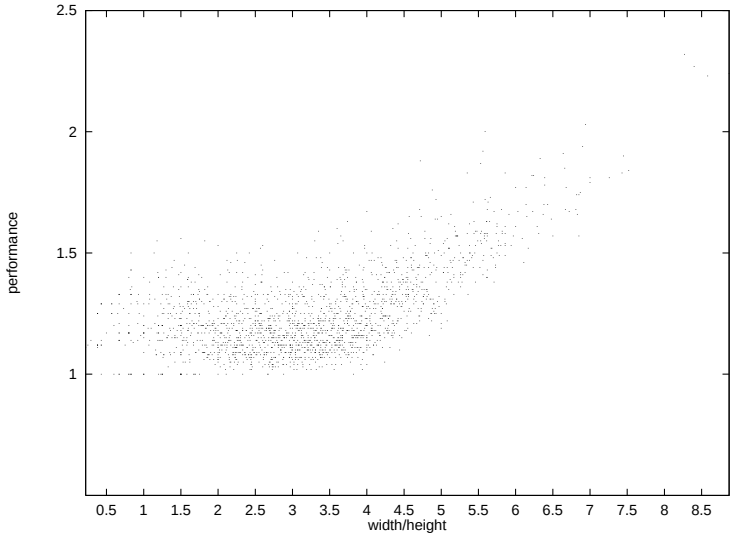

Figure 31: List algorithm according to the height of the tree (capacity 5)

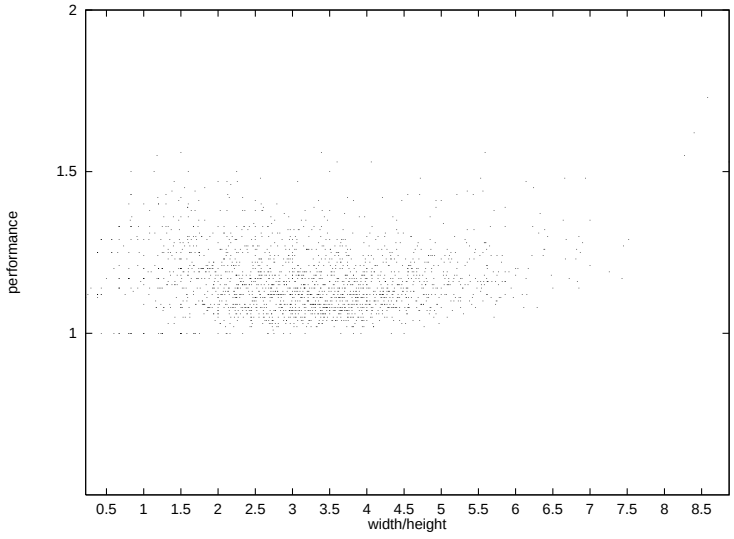

Figure 32: List algorithm according to the height of the tree (capacity 10)

5.2.5 The Mean Number of processors as a Function of the Number of Nodes

Now, we compare the heuristics according to the number of processors they use. First, we have to mention that this number does not depend on the capacity when the method based on Chrétienne's algorithm is used. Indeed, the number of processors required in that case is exactly the one found by Chrétienne's algorithm since we only schedule the communications of the relaxed solution to get a feasible solution (remember that the allocation of tasks to the processors remains the same during that step).

The first observation to make is that this number increase with the number of nodes, whatever the heuristic we consider. Next, whereas the method based on Chrétienne's algorithm requires from 2 or 3 processors (for a tree with 10 nodes) to around 210 (for a tree with 485 nodes) whatever the capacity is, our algorithm never uses more than around 25 processors and the three list algorithm never use more than around 5, 15 and 20 processors in case of a capacity equal to 1, 5 and 10 respectively.

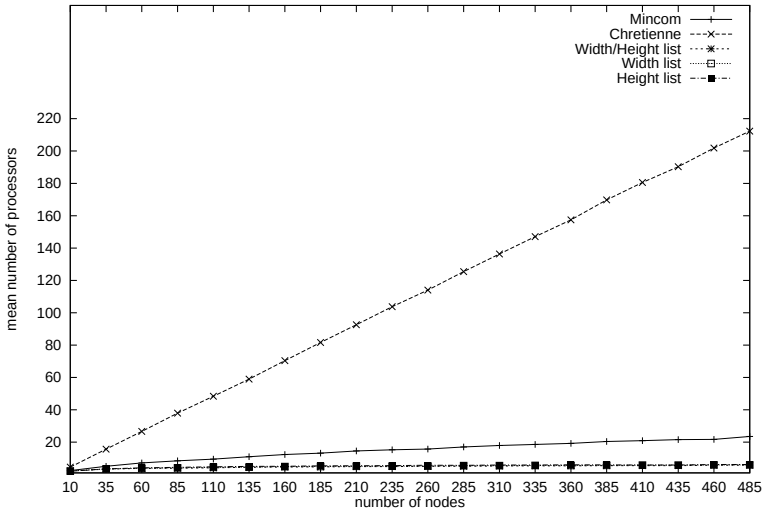

Figure 33: Capacity 1

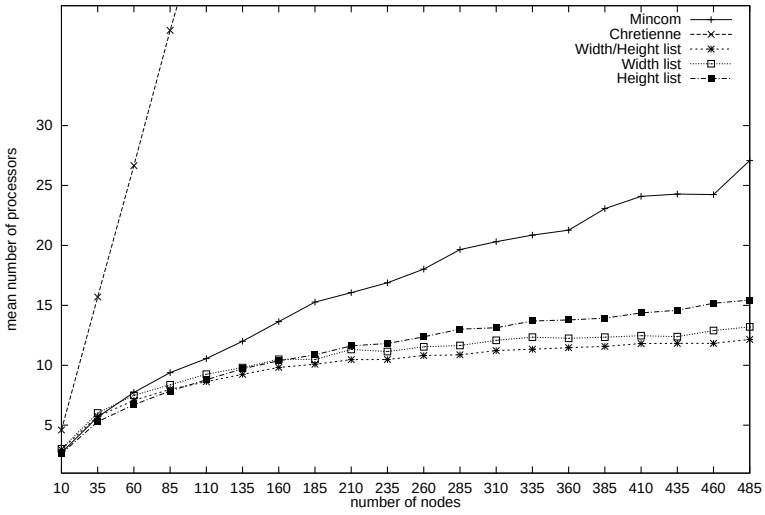

Figure 34: Capacity 5

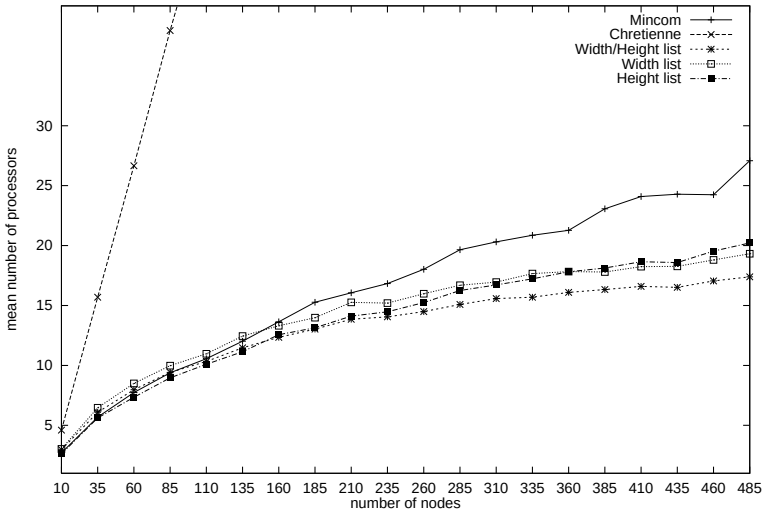

Figure 35: Capacity 10

6 Conclusion

We have shown that it is possible to design approximation algorithms with performance guarantee for two problems where the number of communications is limited. Moreover, We have proved that the two problem are related from the approximation point of view. In this way, we have used the approximation of a the former problem to approximate the latter.

For the first problem (MINCOM-BOUND) we have proposed a 2-approximation algorithm. This algorithm has been proved for the case of an out-tree with the independent-data semantics, but is still valid for an in-tree (the two semantics are in this case the same) and for an in-tree with the independent-data semantics, since a node has at most only one immediate successor. However, in case of an out-tree with the common-data semantics, the algorithm is no longer valid, and the approximation seems harder. For instance, if we consider a 1-type tree with several 0-type subtrees the problem of minimizing the number of communications reduces to the BINPACKING problem, which is NP-hard (recall that the same problem is polynomial in case of the independent-data semantics).

For the second problem (MINTIME-BUS) we have proposed a 6-approximation algorithm. We do not know, contrary to the previous algorithm, if the bound is tight. However we have compared its performance with that of other possible heuristics, by randomly generating an important number of instances. These tests show that the performance ratio of our algorithm is stable around the value 1.25 while the worst case ratio is at most equal to 6. They also show that 1) our algorithm is the most efficient if the capacity is small, 2) list algorithms are the most interesting if we want to use a number of processors as small as possible and 3) the method based on Chrétienne's algorithm is the most efficient if the capacity is great, but it requires a great number of processors.

The approximation of more general problems (arbitrary precedence graph, arbitrary number of processors, ...) seems to be much harder. However, we could test some heuristics. For example, it would be interesting to compare list algorithm with a heuristic based on the relaxation of the problem, as we have done. For the case of scheduling an arbitrary precedence graph on an unlimited number of processors, it is possible to use the $\frac{4}{3}$-approximation of König and Munier [6] for the relaxed problem in the model UET-UCT.

To conclude, we think that the model of Afrati, Papadimitriou, Papageorgiou and Prastein could be a basic model for scheduling problems with a limited number of communications. Despite the difficulty of proving performance guarantee for general problems, we think that their model could be useful for designing good heuristics.

Acknowledgment

I wish to express my sincere thanks to Prof. Claire Hanen for her careful

review and for many improvements in the proof of theorem 3.

References

[1] F. Afrati , C. H. Papadimitriou, and G. Papageorgiou (1988) *Scheduling dags to minimize time and communication*, in Proceedings of 3rd Conference VLSI Algorithms and Architectures AWOC, Lecture Notes in Computer Science, Vol. 319, pp. 134–138.

[2] L. Alonso and R. Schott (1995) *Random generation of trees*, Kluwer Academic Publishers.

[3] Ph. Chrétienne (1989) *A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints*, European Journal of Operational Research, Vol. 43, pp. 225–230.

[4] Ph. Chrétienne, E. G. Coffman Jr, J. K. Lenstra and Z. Liu, editors (1995) *Scheduling theory and its applications*, J. Wiley.

[5] L. Finta and Z. Liu (1997) *Complexity of Task graph Scheduling with Fixed Communication Capacity*, International journal of Foundations of Computer science, Vol. 8, No. 1, pp. 43–36.

[6] J-C König and A. Munier (1993) *A heuristic for a scheduling problem with communication delays*, Rapport technique LRI no. 871, Université de Paris-sud, Orsay, France.

[7] E. L. Lawler (1973) *Optimal sequencing of a single machine subject to precedence constraints*, Management Science, Vol. 19, pp. 544–546.

[8] M. G. Norman, S. Pelagatti and P. Thanish (1995) *On the complexity of scheduling with communication delay and contention*, Parallel Processing Letters, Vol. 5, No. 3, pp. 331–341.

[9] M. L. Prastein (1987) *Precedence-constrained scheduling with minimum time and communication*, Technical Report UILU-ENG-87-2207, ACT-75, Coordinated Science Lab., Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, ILL. USA.

[10] V. J. Rayward-Smith (1987) *UET scheduling with interprocessor communication delays*, Discrete Applied Mathematics, Vol. 18, pp. 51-71.

[11] A. Vessereau (1996) *La statistique*, Presses Universitaires de France, collection "Que sais-je ?", Paris, France.