ELSEVIER

Information Processing Letters 58 (1996) 255-263

Information Processing Letters

Scheduling complete intrees on two uniform processors with communication delays

Jacek Błażewicz [a], Pascal Bouvry [b], Frédéric Guinand [c], Denis Trystram [c,*]

[a] *Institute of Computing Science, Poznań University of Technology, Poznań, Poland*

[b] *Interactive Systems, Center for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands*

[c] *LMC-IMAG, 46, avenue Félix Viallet, 38031 Grenoble Cedex, France*

Received 3 September 1993
Communicated by L. Kott

Abstract

We present an optimal algorithm for scheduling a complete k-ary tree on two uniform processors of different speeds in order to minimize schedule length. We consider the basic case of unit standard execution times and unit communication times.

Keywords: Parallel algorithms; Parallelism; Scheduling; Intrees; Uniform processors

1. Introduction

New computer technologies allow a more intensive use of multiprocessor systems speeding up the computations. On the other hand, in order to achieve real increase of the processing speed in such systems, methods should be elaborated that properly schedule tasks on a set of parallel processors.

This is especially true in systems, where different modules (tasks) of the program are allocated to different processors and communications (data transmissions) among modules are required [1,7]. Recently the problem of scheduling tasks on parallel processors, taking into account communication delays, has been considered in [2-6]. In all these papers the schedule length has been chosen as a criterion.

In [6] a version of the problem has been considered in which the number of identical processors is unlimited. The authors provide an algorithm which approximates the optimal schedule length with a worst case ratio of two. This algorithm provides an asymptotically optimal schedule length for complete binary trees: $O((\tau \log n)/\log \tau)$, where τ represents the message-to-instruction ratio.

A very similar problem has been considered by Chrétienne and Picouleau [2,3] under the assumption that the number of processors is still unlimited and that the communication delays as the processing times of the tasks are not fixed. They show that in the case of intrees, the problem of finding an optimal schedule is NP-hard, even if the height of the tree is at most two (the problem is called the *harpoon problem*).

* Corresponding author. Mailing address: LMC-IMAG, Campus Universitaire, Institut Fourier, BP 53x, 100 rue de Mathematique, 38041 Grenoble Cedex 9, France. Email: denis.trystam@imag.fr.

PII S0020-0190(96)00062-2

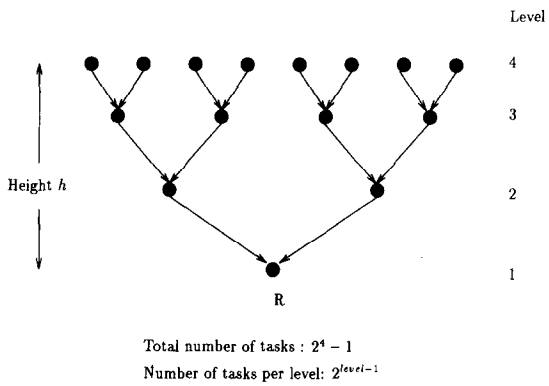


Fig. 1. Example of a complete binary intree with height = 4.

In [4] the authors describe an $O(n^{\tau+1})$ algorithm based on dynamic programming which provides a schedule of minimal length. They improve the asymptotic lower bound presented in [6] for full binary trees by a factor 2: $2(\tau \log n)/\log \tau$.

Lastly in [5] it has been shown that the problem of scheduling unit length tasks forming trees on parallel processors remains NP-hard either for binary trees and uniform communication delays or for complete binary trees, but varying communication delays. A polynomial-time algorithm minimizing schedule length has been also presented for complete k-ary trees and uniform communication delays.

In the present paper we extend the above model by assuming that processors are uniform, that is they differ by their speeds. A polynomial-time algorithm is given for the case of unit standard execution time tasks forming complete intrees to be scheduled on two uniform processors with speeds equal to 2 and 1 respectively. Unit communication times are assumed. Before presentation of the algorithm we set up the subject more precisely.

We consider a set of precedence constrained tasks forming a complete intree. All tasks are assumed to have unit standard execution times. A processor set consists of two processors with speeds equal to 2 and 1, respectively. Thus, the execution of a task takes one unit of time on the fastest processor (denoted by P_f) and two units of time on the slowest processor (denoted by P_s).

Each task is non-preemptable and needs only one processor for its execution.

Two tasks joined by an arc and processed on different processors must always communicate and such a communication between processors takes one unit of time.

Such a communication can be overlapped with the processing of some other tasks on both processors, whenever enough tasks independent of the tree for which the transmission occurs, exist.

In the following we will consider complete k-ary intrees. Each tree is characterized by its height h ($h > 1$) and by arity k. It follows that such a tree contains $n = (k^h - 1)/(k - 1)$ nodes (or tasks). Let us see Fig. 1 for the basic notations (precedence constraints are ignored in most of the figures for the sake of presentation).

The considered criterion is schedule length. Using the notation of Veltman [7], the problem can be written as follows:

$$Q2 \mid \text{complete intree}, p_i = 1, c = 1 \mid C_{\max}$$

In the next two sections, an $O(n)$-time algorithm for the above problem will be presented.

2. Theoretical analysis

2.1. Some preliminaries

Firstly, a lemma is proved, which discusses an optimal assignment of the root of an intree.

Lemma 1. *No optimal schedule can be found, with the root allocated to P_s.*

Proof. Each schedule with the root allocated to P_s belongs to one of the three following classes:

- A processing of the tasks allocated to P_s (except for the root) finishes before a processing of all the tasks allocated to P_f (see Fig. 2, Case 1).
- A processing of the tasks allocated to P_s (except for the root) and a processing of all the tasks allocated to P_f finish at the same time (see Fig. 2, Case 2).
- A processing of the tasks allocated to P_s (except for the root) finishes after a processing of all the tasks allocated to P_f (see Fig. 2, Case 3).

In every case, the allocation of the root to P_f leads to a better schedule (in the third case, to obtain the

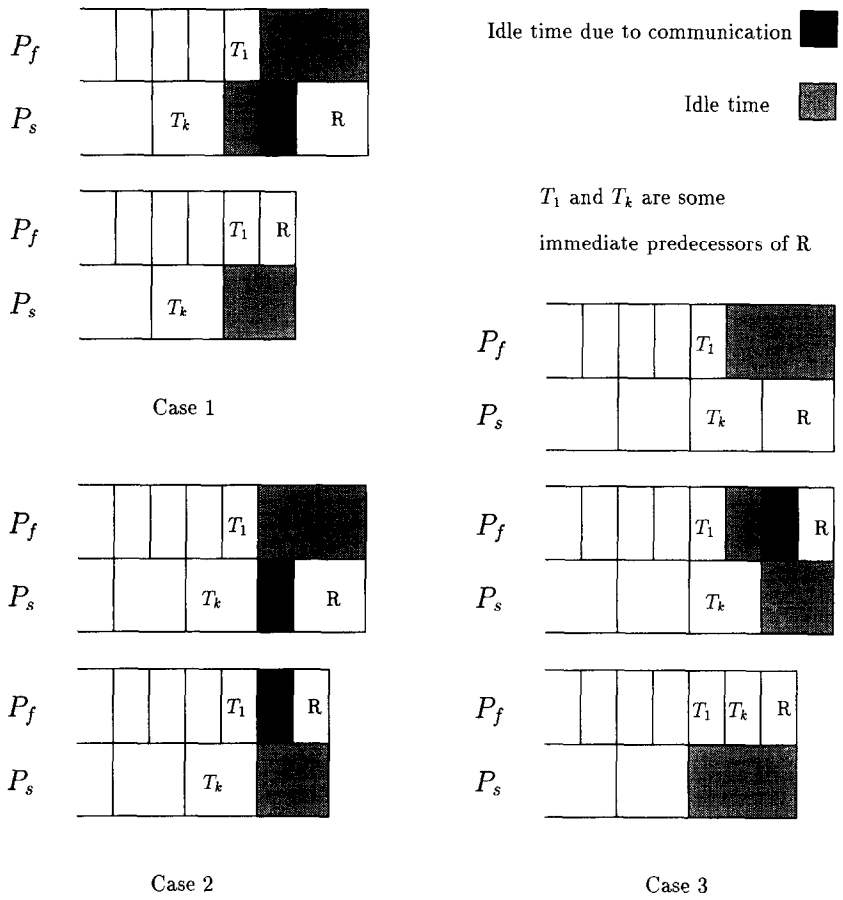


Fig. 2. Allocation of the root for the optimality.

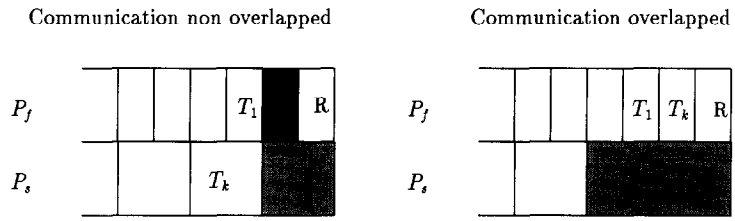
optimal schedule, it is necessary to move another task from P_s to P_f, to overlap the communication). □

2.2. *Basic property*

A general rule which is to be followed here is the so-called load-balancing. Here it means equal—in the sense of a sum of real processing times (plus possibly communication delays)—assignment of tasks to processors. Moreover, this equality is tested level by level. Let n denote the total number of tasks. We calculate first the number of tasks to be executed on P_s in an optimal schedule assuming a removal of the precedence constraints. As P_f is twice faster than P_s, it can execute two tasks, while P_s can only execute one.

Thus, if we allocate $\lfloor n/3 \rfloor$ tasks to P_s, which corresponds to load-balancing of the tasks (taking into account the relative speeds) between both processors (corresponding to the ratio of their processing speeds), we obtain an optimal schedule.

Following Fig. 2, we see that without relaxation, when P_s finishes an execution of all tasks assigned to it, it has to send data from the last task computed to P_f. It needs two units of time (one unit for the communication and one unit for the execution of at least one task (the root)). So, there remain at least two tasks to be executed on P_f, except in the second case of Fig. 2 where the root only remains to be executed. In this case, however, it is possible to move task T_k from P_s to P_f, without increasing the schedule length. So, the allocation of $\lfloor (n-2)/3 \rfloor$ tasks on P_s, keeping P_f always busy (i.e. with all the communications overlapped), leads to the optimality of the schedule (see Fig. 3).



Communications can be overlapped without increasing the schedule length

Fig. 3. Compact schedule.

This is an upper bound for the number of tasks to be allocated to P_s. Thus, a lower bound on the number of tasks executed by P_f for a complete k-ary tree of height h is:

$$n - \left\lfloor \frac{n-2}{3} \right\rfloor, \quad \text{where } n = \frac{k^h - 1}{k-1}.$$

As P_f executes one task per unit of time, and assuming that all the communications are overlapped and there is no idle time for P_f, this is also the lower bound on the schedule length.

3. Scheduling algorithm

We present now a general algorithm for k-ary trees. It is a level by level based algorithm.

3.1. Description of the algorithm

The idea of the algorithm is to load-balance the tasks of a given level as much as possible.

The algorithm can be split into three steps.

Algorithm 1.

Step 1. Allocate the tasks of the highest level (the hth level) in the following way:

- $\lfloor k^{h-1}/3 \rfloor$ tasks to P_s and
- $2\lfloor k^{h-1}/3 \rfloor$ tasks to P_f.

The tasks which are to be allocated to P_s are chosen from the right to the left of the tree (in the following steps, the choice is done in the same way).

The remaining $k^{h-1} - 3\lfloor k^{h-1}/3 \rfloor$ tasks of this level are to be allocated in the way described in Step 2.

Step 2. Add the remaining tasks to those of the following level (the $(h-1)$th level at the first iteration), and, as in the previous step, load-balance as much as possible these tasks.

Repeat this step until the second level of the tree is reached.

Step 3. With this last step, the schedule is completed by taking care of the overlapping of the communications between the tasks at the second and the first levels respectively. For that, if j tasks have to be allocated at the second level (including the remaining tasks of the previous level), we can only allocate $\lfloor (j-1)/3 \rfloor$ tasks on P_s, because we need one unit of time to send data from P_s to P_f between level 2 and 1 (see Fig. 4).

Finally the root is allocated to P_f.

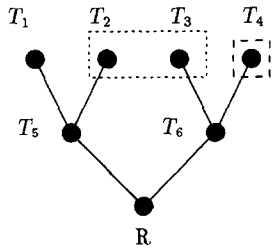
3.2. Optimality

Theorem 2. *Algorithm* 1 *always constructs an optimal schedule.*

Proof. Following the basic property of the scheduling problem (discussed in Section 2.2) we need only to show that every schedule constructed by Algorithm 1 has the two following properties:

- the number of tasks allocated to P_s is $\lfloor (n-2)/3 \rfloor$ and
- there is no idle time on P_f.

In order to prove the optimality of Algorithm 1 for all complete k-ary intrees, we have to verify the above two properties for $k = 3N$, $k = 3N+1$, and $k = 3N-1$ (with $N \in \mathbb{N}^*$).



At the first step:

Task T_4 is allocated to P_s
and T_2, T_3 to P_f, respectively.

At the second step: If we consider all the tasks minus one (i.e $\lfloor \frac{j-1}{3} \rfloor$), the communication can be overlapped.

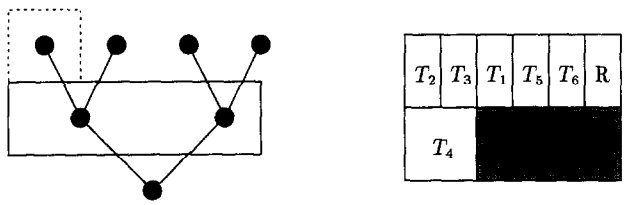


Fig. 4. An example of the end of a schedule in case of binary trees.

3.2.1. Case 1: $k = 3N$ *(with* $N \in \mathbb{N}^*$*)*

Verification of the number of tasks allocated to P_s Since the number of tasks at any level $i+1$ is $(3N)^i$ it can be divided by 3. Thus, the total number of tasks allocated to P_s from level h through level 3 is

$$\frac{1}{3}\left(\frac{k^h-1}{k-1}-(k+1)\right).$$

Moreover, in the two last levels, the total number of tasks allocated to P_s is $N-1$. Summing up, the total number of tasks which are allocated to P_s for the complete tree is:

$$N-1+\frac{1}{3}\left(\frac{k^h-1}{k-1}-(k+1)\right).$$

It remains to verify the following equation:

$$N-1+\frac{1}{3}(n-(k+1)) \stackrel{?}{=} \left\lfloor \frac{n-2}{3} \right\rfloor.$$

The left-hand side of the equation can be transformed in the following way:

$$\frac{k}{3}-1+\frac{n}{3}-\frac{k}{3}-\frac{1}{3}=\frac{n-4}{3}.$$

The right-hand side:

$$\left\lfloor \frac{n-4}{3}+\frac{2}{3} \right\rfloor = \frac{n-4}{3}+\left\lfloor \frac{2}{3} \right\rfloor.$$

Finally we get

$$\frac{n-4}{3}=\frac{n-4}{3}+\left\lfloor \frac{2}{3} \right\rfloor.$$

So the first property is proved for this case.

Verification that P_f *is always busy* As for each level the total number of tasks is divided by 3, between two different levels $i+1$ and i, there is no communication between processors, because the tasks of the ith level have all their predecessors allocated to the same processor.

Moreover, at the end of the schedule (i.e. for levels 2 and 1), if P_s executes some tasks of level 2, it communicates some data to the root (allocated to P_f). P_s has $N-1$ tasks to execute while P_f has $2N+1$ tasks to execute. Then, as P_s needs $2(N-1)$ units of time for the execution of these tasks, and P_f needs $2N+1$, there remain 3 units of time for the overlap of the communication, and no idle time on P_f appears.

In conclusion, for this case ($k = 3N$), both properties have been verified and hence the schedule is optimal.

3.2.2. Case 2: $k = 3N-1$ (with $N \in \mathbb{N}^*$)

Verification of the number of tasks allocated to P_s When $k = 3N-1$, the number of tasks at a given level $i+1$ is $k^i = (3N-1)^i$.

This number cannot be divided by 3, but the sum of the tasks of two consecutive levels can be divided by 3, indeed:

$$(3N-1)^i + (3N-1)^{i-1} = 3N(3N-1)^{i-1}.$$

So, in this case we consider the load-balanced allocation for pairs of levels.

As in the previous case, we can calculate the number of tasks which are allocated to P_s. Two cases occur:

- *h is even.* Using Algorithm 1, we allocate the tasks belonging to levels h through 3, and we assign to P_s exactly $(n-(k+1))/3$ tasks.

Moreover, in the two last levels, the total number of tasks allocated to P_s is $N-1$; then, the total number of tasks which are allocated to P_s for the complete tree is:

$$N-1+\frac{1}{3}\left(\frac{k^h-1}{k-1}-(k+1)\right),$$

hence, $(n-3)/3$. But, h is even, so

$$\left\lfloor\frac{n-2}{3}\right\rfloor = \frac{n-3}{3} + \left\lfloor\frac{1}{3}\right\rfloor = \frac{n-3}{3}.$$

- *h is odd.* Using Algorithm 1, we allocate the tasks belonging to levels h through 4, and we assign to P_s exactly $(n-(k^2+k+1))/3$ tasks.

Moreover, in the three last levels (level 1, 2 and 3), the total number of tasks allocated to P_s is $\lfloor k^2/3\rfloor + N - 1$; thus, the total number of tasks which are allocated to P_s for the complete tree is:

$$\left\lfloor\frac{k^2}{3}\right\rfloor + N - 1 + \frac{1}{3}\left(\frac{k^h-1}{k-1}-(k^2+k+1)\right).$$

Since $k^2 = (3N-1)^2$, we have $k^2 = 3q+1$ ($q \in \mathbb{N}^*$). Then, $\lfloor k^2/3\rfloor - k^2/3 = -1/3$. Hence, the total number of tasks allocated to P_s is $(n-4)/3$. But, h is odd, so

$$\left\lfloor\frac{n-2}{3}\right\rfloor = \frac{n-4}{3} + \left\lfloor\frac{2}{3}\right\rfloor = \frac{n-4}{3}.$$

Thus, the property is true for all k-ary complete trees with $k = 3N-1$.

Verification that P_f is always busy The following proposition will be useful for proving this property.

Proposition 3.

- *If i is even then* $(3N-1)^i = 3\lfloor(3N-1)^i/3\rfloor + 2$.
- *If i is odd then* $(3N-1)^i = 3\lfloor(3N-1)^i/3\rfloor + 1$.

We consider now the allocation of pairs of levels and there are two kinds of communications (see Fig. 5):

- communications from P_f to P_s (for white tasks in Fig. 5) which are called *intra-pair communications*,
- communications from P_s to P_f (for dotted tasks in Fig. 5) which are called *inter-pair communications*.

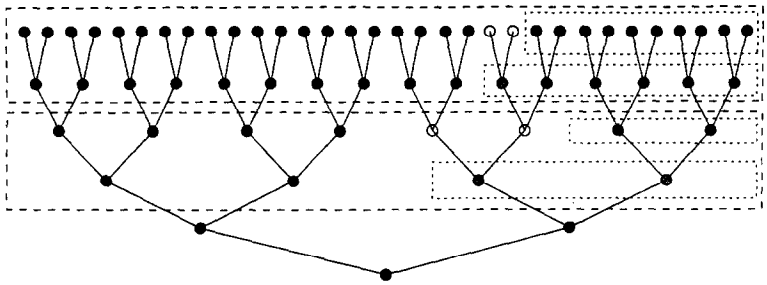
We will prove that these communications can be overlapped in both cases.

- *h is even.* In each pair of levels, the higher level l is always even. So, following Proposition 3, we have

$$(3N-1)^{l-1} = 3\left\lfloor\frac{(3N-1)^{l-1}}{3}\right\rfloor + 2.$$

Then, the number of tasks allocated to P_s is $(k^{l-1}-2)/3$ at level l and $(k^{l-2}+2)/3$ at level $l-1$. A task belonging to level $l-1$ needs data from k tasks belonging to the lth level. To avoid communications from P_f to P_s, P_s would execute $k(k^{l-2}+2)/3$ tasks belonging to level l. But, only $(k^{l-1}-2)/3$ tasks of level l are allocated to P_s. So, there is a lack of $2*(k+1)/3 = 2N$ tasks. It means that there are $2N$ communications of that type which cannot be avoided. For the overlapping of these communications, P_s has to be busy during at least $2N+1$ units of time. Since $(k^{l-1}-2)/3$ tasks are allocated to P_s, we only have to verify

$$\frac{k^{l-1}-2}{3} - N \geqslant 2N+1. \qquad (1)$$



○ Tasks allocated to P_f which communicate data to P_s

● Tasks allocated to P_s which communicate data to P_f

Fig. 5. Intra- and inter-pair communications.

Since h is even, $l \geqslant 4$ (because of Step 3 of Algorithm 1), we see that (1) is always true, and then the communications between both processors inside a pair of levels, are always overlapped.

Between two levels ($l'-1$ and l, where $l' = l+2$) of two consecutive pairs, the communications occur only from P_s to P_f. Indeed, the number of tasks allocated to P_f at level $l+1$ is $k^l - (k^l+2)/3$ and at level l is $k^{l-1} - (k^{l-1}-2)/3$. To avoid communications from P_s to P_f, we need $k^l - (k^l - 2k)/3$ tasks at level $l+1$, but we have only $k^l - (k^l+2)/3$. So, there are $2N$ communications which cannot be avoided.

As in the previous case, these communications can be overlapped, because $k^{l-1} - (k^{l-1}-2)/3$ tasks are allocated to P_f at the lth level. So, we have only to prove that

$$k^{l-1} - \frac{k^{l-1}-2}{3} - N \geqslant 2N+1. \qquad (2)$$

Since $h \geqslant 2$, $l \geqslant 4$ and $k = 3N-1$, this inequality is proved.

Then, if h is even, both kinds of communications are overlapped.

- *h is odd.* This case is exactly the same as the previous one, except that we use the second relation of Proposition 3: If i is odd then

$$(3N-1)^i = 3\left\lfloor \frac{(3N-1)^i}{3} \right\rfloor + 1.$$

We prove in the same way that all the communications which cannot be avoided are overlapped.

Then, both properties are verified. So the schedule given by the algorithm is optimal for the case $k = 3N-1$.

3.2.3. Case 3: $k = 3N+1$ (*with* $N \in \mathbb{N}^*$)

In this case, considering one level ($3N+1$ cannot be divided by 3), or two levels ($(3N+2)(3N+1)^{l-1}$ cannot be divided by 3) is not enough. But, if we consider three consecutive levels, the sum of tasks can be divided by 3. Indeed,

$$(3N+1)^l + (3N+1)^{l-1} + (3N+1)^{l-2} = 3(3N^2+3N+1)(3N+1)^{l-2}.$$

Thus, we can load-balance a schedule for each triple of tree levels. As in both previous cases, we prove the two properties.

Verification of the number of tasks allocated to P_s

For each set of three consecutive levels (levels l, $l-1$, $l-2$), we allocate $(3N^2+3N+1)k^{l-2}$ tasks to P_s. Then, three cases occur:

- $h = 3L-1$ (with $L \in \mathbb{N}^*$). We consider all the levels minus level 1 and 2.
- $h = 3L$. We consider all the levels minus level 1, 2 and 3.
- $h = 3L+1$. We consider all the levels minus level 1, 2, 3 and 4.

Thus, the number of tasks allocated to P_s for each case is:

$$(3N^2+3N+1)\sum_{i=1}^{\lfloor (h-2)/3\rfloor} k^{h-3i}+Z,$$

where

$$Z=\begin{cases}\left\lfloor\dfrac{k-1}{3}\right\rfloor, & \text{for } h=3L-1,\\ \left\lfloor\dfrac{k^2+k-1}{3}\right\rfloor, & \text{for } h=3L,\\ \left\lfloor\dfrac{k^3+k^2+k-1}{3}\right\rfloor, & \text{for } h=3L+1.\end{cases}$$

Thus, we have to prove that

$$\left\lfloor\frac{1}{3}\left(\frac{k^h-1}{k-1}-2\right)\right\rfloor = (3N^2+3N+1)\sum_{i=1}^{\lfloor (h-2)/3\rfloor} k^{h-3i}+Z.$$

As an example the case $h=3L-1$ is shown below.

$$\left\lfloor\frac{1}{3}\left(\frac{k^h-1}{k-1}-2\right)\right\rfloor = \left\lfloor\frac{1}{3}\left(\left(\frac{k^h-1}{k-1}-(k+1)\right)+k+1-2\right)\right\rfloor = \frac{1}{3}\left(\frac{k^h-1}{k-1}-(k+1)\right)+\left\lfloor\frac{k-1}{3}\right\rfloor. \quad (3)$$

Moreover, for $h=3L-1$, $h-3(\lfloor (h-2)/3\rfloor)=2$. Then

$$(3N^2+3N+1)\sum_{i=1}^{\lfloor (h-2)/3\rfloor} k^{h-3i}+\left\lfloor\frac{k-1}{3}\right\rfloor = \frac{1}{3}\left(\frac{k^h-1}{k-1}-(k+1)\right)+\left\lfloor\frac{k-1}{3}\right\rfloor. \quad (4)$$

We see that (3) and (4) are equal. The remaining two cases are proved in a similar way.

Verification that P_f is always busy We remark that

$$k^h=3A+1 \quad (\text{with } A\in\mathbb{N}).$$

Indeed,

$$\sum_{i=1}^{h-1} k^i=\frac{k^h-1}{k-1}\sum_{i=1}^{h-1} k^i = \frac{k^h-1}{3N}k^h=3\left(N\sum_{i=1}^{h-1}k^i\right)+1. \quad (5)$$

Thus, for a level $h-3i>2$ (with $i\in\mathbb{N}$) there remains only one task which is not allocated at Step 1; for a level $h-1-3i>2$ (with $i\in\mathbb{N}$) there remains only two tasks which are not allocated with the other tasks of the same level; for a level $h-2-3i>2$ (with $i\in\mathbb{N}$) all the tasks are allocated. This leads to the following remarks:

- between a level $h-3i$ and a level $h-1-3i$ there are N communications from P_s to P_f which cannot be avoided because
$$\frac{k^{h-3i}-1}{3}-k\frac{k^{h-1-3i}-1}{3}=\frac{k-1}{3}=N;$$
- between a level $h-1-3i$ and a level $h-2-3i$ there are $2N+1$ communications from P_f to P_s which cannot be avoided because
$$\frac{k^{h-1-3i}-1}{3}-k\frac{k^{h-2-3i}+2}{3}=\frac{2k+1}{3}=2N+1;$$
- between a level $h-1-3i$ and a level $h-2-3i$ there are $N+1$ communications from P_s to P_f which cannot be avoided because
$$\frac{k^{h-2-3i}+2}{3}-k\frac{k^{h-3(i+1)}-1}{3}=\frac{k+2}{3}=N+1.$$

We can easily show that all these communications can be overlapped in the same way as in the case $3N-1$.

For the three different cases of arity of a complete k-ary intree, both properties on the number of tasks allocated to P_s and on the unbroken activity of P_f have been proved which leads to the optimality of Algorithm 1.

3.3. Complexity of the algorithm

The time complexity of the algorithm is linear. Indeed, for each level the number T_s of tasks to be allocated to P_s is calculated which is constant in time. The T_sth first tasks on the right of the tree are allocated to P_s, the other tasks of this level are allocated to P_f. So for a given level, the amount of time needed by the algorithm to provide a schedule is constant in

time; therefore, since the height of the tree is h, Algorithm 1 is an $O(h)$ algorithm.

4. Conclusion

This result is close to those of Jakoby and Reishuk [5] because we consider complete k-ary intrees with UET tasks. The differences are in the choice of the communication cost, constant in our case and uniform in their paper, and in the choice of processors, identical in their work and uniform in our case. The case of speeds 1 and s (with $s \in \mathbb{N}$), with the same hypothesis of execution times and of communication costs, could be the next contribution. The case of speeds 1 and 2, with UET tasks and uniform communication delays, could be another one, since Jakoby and Reishuk [5] provide a polynomial algorithm to solve this problem with identical processors.

References

[1] J. Błażewicz, K. Ecker, G. Shmidt and J. Weglarz, *Scheduling in Computer and Manufacturing Systems* (Springer, Berlin, 1993).

[2] P. Chrétienne, A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints, *European J. Oper. Res.* **43** (1989) 225–230.

[3] P. Chrétienne and C. Picouleau, The basic scheduling with interprocessor communication delays, Tech. Rept. MASI 91.6, Paris, 1991.

[4] H. Jung, L. Kirousis and P. Spirakis, Lower bounds and efficient algorithms for multiprocessor scheduling of dags with communication delays, in: *Proc. 1st SPAA* (1989) 254–264.

[5] A. Jakoby and R. Reishuk, The complexity of scheduling problems with communication delays for trees, in: *Algorithm Theory SWAT'92*, 1992.

[6] C.H. Papadimitriou and M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms, in: *Proc. 20th Ann. ACM Symp. on Theory of Computing*, 1988.

[7] B. Beltman, B.J. Lageweg and J.K. Lenstra, Multiprocessor scheduling with communication delays, *Parallel Comput.* **16** (1990) 173–182.