A polynomial-time algorithm for allocating independent tasks on heterogeneous fork-graphs

Olivier Beaumont, Arnaud Legrand, Yves Robert

▶ **To cite this version:**

Olivier Beaumont, Arnaud Legrand, Yves Robert. A polynomial-time algorithm for allocating independent tasks on heterogeneous fork-graphs. [Research Report] LIP RR-2002-07, Laboratoire de l'informatique du parallélisme. 2002, 2+10p. hal-02101974

HAL Id: hal-02101974

https://hal-lara.archives-ouvertes.fr/hal-02101974v1

Submitted on 17 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

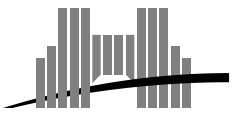
CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

A polynomial-time algorithm for allocating independent tasks on heterogeneous fork-graphs

Olivier Beaumont
Arnaud Legrand
Yves Robert

February 2002

Research Report N° 2002-07



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

INRIA

A polynomial-time algorithm for allocating independent tasks on heterogeneous fork-graphs

Olivier Beaumont
Arnaud Legrand
Yves Robert

February 2002

Abstract

In this paper, we consider the problem of allocating a large number of independent, equal-sized tasks to a heterogeneous processor farm. The master processor P_0 can process a task within w_0 time-units; it communicates a task in d_i time-units to the i-th slave P_i, $1 \leq i \leq p$, which requires w_i time-units to process it. We assume communication-computation overlap capabilities for each slave (and for the master), but the communication medium is exclusive: the master can only communicate with a single slave at each time-step.
We give a polynomial-time algorithm to solve the following scheduling problem: given a time-bound T, what is the maximal number of tasks that can be processed by the master and the p slaves within T time-units?

Keywords: heterogeneous processors, scheduling, mapping, master-slave, complexity

Résumé

Nous nous intéressons à l'ordonnancement de tâches indépendantes sur une plateforme maître-esclave hétérogène, modélisée par un graphe *fork*. Le maître et les p esclaves ont des vitesses de calcul différentes. De même, les liens de communication entre le maître et chaque esclave ont des capacités différentes. Le modèle autorise le recouvrement calcul-communication sur chaque processeur, mais on suppose que toutes les communications s'effectuent en mode 1-port: à tout instant, le maître ne peut envoyer qu'un seul message (à un esclave donné).
Nous présentons un algorithme polynomial pour résoudre le problème d'ordonnancement correspondant: étant donnée une borne temporelle T, maximiser le nombre total de tâches qui peuvent être effectuées par le maître et les p esclaves en temps T.

Mots-clés: processeurs hétérogènes, heuristiques d'ordonnancement, distribution, maître-esclave, complexité

1 Introduction

In this paper, we deal with the problem of allocating a large number of independent, equal-sized tasks to a heterogeneous processor farm. The master processor P_0 can process a task within w_0 time-units: it communicates a task in d_i time-units to its i-th slave P_i, $1 \leq i \leq p$, which requires w_i time-units to process it. We assume communication-computation overlap capabilities for each slave (and for the master), but the communication medium is exclusive. In other words, the master can only communicate with a single slave at each time-step. We state the communication model more precisely:

- a given communication from P_0 to P_i lasts d_i units of time
- P_0 cannot be involved in any other communication during these units of time
- P_i cannot start the execution of the task before the communication is completed.

We give a polynomial-time algorithm to solve the following scheduling problem (to be stated more formally in Section 2: given a time-bound T, what is the maximal number of tasks that can be processed by the master and the p slaves within T time-units? The polynomial complexity of this scheduling problem is established in Sections 3 and 4. This result is rather surprising: indeed, two variants of the problem are shown NP-complete in Section 5. Finally, we survey several related problems from the literature in Section 6.

2 Problem statement

We consider a fork-graph (see Figure 1) with a master-processor P_0 and p slaves P_i, $1 \leq i \leq p$. The master processor P_0 owns an arbitrarily large number of independent, equal-size, non-preemptive tasks; it can process a given task within w_0 time-units; in parallel, it can send tasks to the p slaves, but only to a single slave at a given time-step.

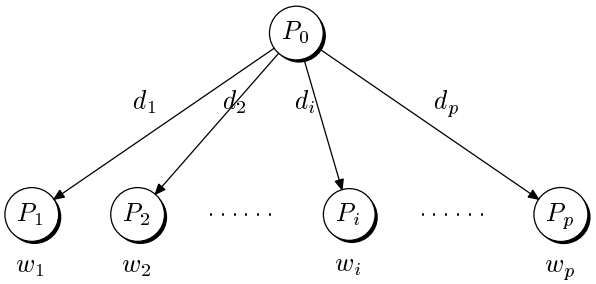


Figure 1: Heterogeneous fork-graph

It takes d_i time-units for slave P_i to receive a task, $1 \leq i \leq p$. More precisely, if P_0 sends a task to P_i at time-step t, P_i cannot start executing this task before time-step $t + d_i$, while P_0 cannot initiate another communication before time-step $t + d_i$. P_i can process a task within w_i time-units; while computing a given task, P_i can receive another task from P_0. See Figure 2 for an example.

We deal with the following scheduling problem: given a time bound T, determine the scheduling and allocation of tasks to processor which maximize the total number of tasks whose execution is completed no later than T. Informally, we have to determine the best ordering of the messages sent by the master, together with the identity of the processors which execute the corresponding tasks, so as to maximize the total number of executed tasks. Note that this scheduling problem is called the NOW-Exploitation problem in [17].

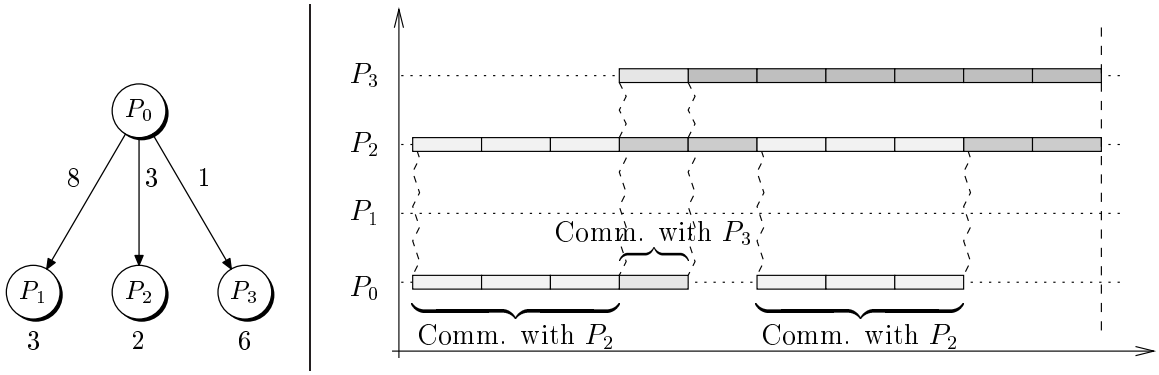


Figure 2: Example of execution: at most three tasks can be scheduled within $T = 10$ time-units.

Remark 1. *Clearly, because the master processor can compute and communicate in parallel, it should be kept busy; within T time-units, P_0 can execute $\lfloor \frac{T}{w_0} \rfloor$ tasks. To deal with the case where the master processor P_0 has no processing capability, simply let $w_0 = +\infty$. Similarly, to deal with the case where the master cannot overlap its own computations with the communications to the slaves, add a new child P_{-1} such that $d_{-1} = w_0$ and $w_{-1} = w_0$, and then let $w_0 = +\infty$.*

To state our complexity result, we have to be more precise about the problem specification. The expected output of the scheduling algorithm includes the list of tasks to be processed by each slave. The size of this list will be linear in the time-bound T, therefore exponential in the problem size if we only let T in the input (because it can be encoded with $O(\log T)$ bits). Rather, we let the input include the original list of all tasks owned by the master. Such an hypothesis makes full sense in practice: even though we assume equal-size tasks, each task will correspond to a different file, and to a different computation, and we must keep track of which task is executed on which computing resource.

We are ready to formally state our scheduling problem (in view of the previous remarks, we omit the computations performed by the master processor):

Definition 1 (MAX-TASKS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T)$). *Given*

- *a set $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of n equal-size independent tasks,*
- *a set $\mathcal{W} = \{w_1, w_2, \ldots, w_p\}$ of p execution times*
- *a set $\mathcal{D} = \{d_1, d_2, \ldots, d_p\}$ of p communication delays,*
- *a time-bound T,*

compute a maximal-cardinal subset of $\mathcal{F}$ of tasks that can be executed within T time-units on a heterogeneous master-slave platform made up with p slaves P_i of computation time w_i and communication time d_i, $1 \leq i \leq p$.

Theorem 1. *MAX-TASKS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T)$ can be solved in polynomial time.*

The proof is constructive. In the following, a (somewhat sophisticated) greedy algorithm is shown to be optimal.

3 Reduction to a problem with at most one task per slave

Proposition 1. *Any instance MAX-TASKS*$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T)$ *can be polynomially reduced to an instance MAX-TASKS-SINGLE*$(n, \mathcal{F}, p', \mathcal{W}', \mathcal{D}', T)$ *where each slave is constrained to execute at most one task.*

Proof. Each slave P_i can be dealt with separately, so let i, $1 \leq i \leq p$, be fixed. We replace slave P_i by a collection of l_i slaves $P_{i,j}$, $1 \leq j \leq l = \min(n, \lfloor \frac{T-w_i}{M_i} \rfloor + 1)$, where $M_i = \max(d_i, w_i)$. See Figure 3: all the new slaves have the same communication capability d_i as P_i, but $P_{i,j}$ processes a task in $w_{i,j} = w_i + (j-1).M_i$ time-units. Obviously, the $j+1$-th task executed by P_i in the original instance will be executed by $P_{i,j}$ in the new instance.

After the transformation, the total number of slaves is bounded by $p \times n$, which is indeed polynomial in the size of the input (because $\mathcal{F}$ is of size at least $O(n)$ and $\mathcal{W}$, or $\mathcal{D}$, of size at least $O(p)$). ■

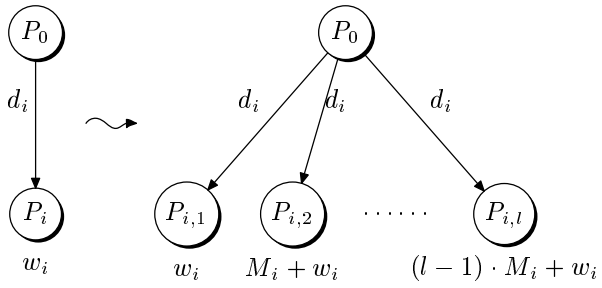


Figure 3: Replacing a slave by a collection of uni-task slaves

4 With at most one task per slave

In this section, we solve MAX-TASKS-SINGLE$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T)$, the problem where each slave is limited to executing at most one-task. The problem is reduced to deciding which slave to communicate with, and in which order. For simplicity, we still use the notations p, $\mathcal{W}$, $\mathcal{D}$ while in fact there are $p' = O(pn)$ slaves after the transformation of Section 3.

We start with a technical lemma:

Lemma 1. *Consider a set of k slaves capable of executing k tasks (one each) within T units with some scheduling. Then, if we sort these slaves by computation times (largest first), and if we schedule the communications from the master P_0 in that order, we can also execute k tasks within T time-units.*

Proof. To simplify notations, let $P_1, P_2, \ldots, P_k$ be the sorted slaves: $w_1 \geq w_2 \geq \ldots \geq w_k$. Let $(P_{i_1}, \ldots, P_{i_k})$ be the ordering of the communications from P_0 in the original schedule. The following set of equations is satisfied:

$$\begin{array}{lllll}
(L_1) & d_{i_1} + w_{i_1} & \leq T & \text{first task} \\
(L_2) & d_{i_1} + d_{i_2} + w_{i_2} & \leq T & \text{second task} \\
\vdots & \ddots & \vdots & \vdots \\
(L_k) & d_{i_1} + d_{i_2} + \cdots + d_{i_k} + w_{i_k} & \leq T & \text{last task}
\end{array}$$

Let $j_1, \ldots, j_k$ be the indices of the tasks of the sorted slaves in the original schedule: $i_{j_1} = 1$, $i_{j_2} = 2$, ... and $i_{j_k} = k$. $i_{j_1} = 1$ means that processor P_1 is assigned the j_1-th task in the original schedule. We prove by induction that tasks can be assigned in the ordering $1, 2, \ldots, k$ of sorted computation times:

i) For the first task, we have $d_1 + w_1 \leq T$. Indeed, see equation L_{j_1}: $d_{i_1} + \cdots + d_{i_{j_1-1}} + d_1 + w_1 \leq T$.

ii) For the second task, we have $d_1 + d_2 + w_2 \leq T$. Indeed, there are two possibilities:

- Either $j_2 > j_1$; then the inequality L_{j_2} contains the quantity $d_{i_{j_1}} + d_{i_{j_2}} + w_{i_{j_2}}$, hence $d_1 + d_2 + w_2 \leq T$
- Or $j_2 < j_1$; then the inequality L_{j_1} contains the quantity $d_{i_{j_1}} + d_{i_{j_2}} + w_{i_{j_1}}$; because $w_2 \leq w_1$, we derive that $d_1 + d_2 + w_2 \leq T$.

iii) For the general case, let $i < k$. Assume that for all $l \leq i$ we have $d_1 + \cdots + d_l + w_l \leq T$. We aim at proving that $d_1 + \cdots + d_{i+1} + w_{i+1} \leq T$. Let $m_i = \max(j_1, \ldots, j_i)$.

- Either $j_{i+1} > m_i$; then we consider the inequality $L_{j_{i+1}}$, which contains the terms $d_1, d_2, \ldots, d_i$ (because $j_{i+1} > m_i$) as well as the term $d_{i+1} + w_{i+1}$. Hence $d_1 + \cdots + d_{i+1} + w_{i+1} \leq T$.
- Or $j_{i+1} < m_i$; then we consider the inequality L_{m_i}, which contains the terms $d_1, d_2, \ldots, d_{i+1}$ as well as the term $d_i + w_i$. Hence $d_1 + \cdots + d_{i+1} + w_i \leq T$. Because $w_i \geq w_{i+1}$, we finally derive that $d_1 + \cdots + d_{i+1} + w_{i+1} \leq T$.

■

We introduce the following greedy algorithm:

```
MASTER-SLAVE((P_1, ..., P_p), T)
1:  Sort slaves such that d_1 ≤ d_2 ... ≤ d_p
2:  L ← ∅
3:  For i = 1 To p Do
4:      If L ∪ {P_i} can be scheduled within T time-units
5:      Then L ← L ∪ {P_i}
6:  Return (L)
```

The condition at line 4 can be checked using Lemma 1: sort the current list L according to the computation times and verify that all inequalities are satisfied.

Proposition 2. *The* Master-Slave *algorithm returns a maximal set of tasks that can be processed within T time-units.*

Proof. To simplify notations, we say that a set of indices $\mathcal{I} = \{i_1, \ldots, i_k\}$ is *schedulable* if there exists a valid scheduling executing k tasks within T time-units, with processors $\{P_{i_1}, \ldots, P_{i_k}\}$.

Assume that the set $\mathcal{I} = \{i_1, \ldots, i_k\}$ is schedulable, and further (without loss of generality) that $d_{i_1} \leq d_{i_2} \leq d_{i_k}$. We show that the MASTER-SLAVE algorithm returns at least k values, i.e. a schedulable set $\mathcal{G} = \{g_1, \ldots, g_k\}$, which will conclude the proof. We proceed by induction.

i) Let $\mathcal{I}_0 = \{i_1, \ldots, i_k\}$. The set $\mathcal{I}_0$ is schedulable. Each subset of $\mathcal{I}_0$ is schedulable. In particular, each singleton $\{i_j\}$ is schedulable. By construction of the MASTER-SLAVE algorithm, g_1 exists and $g_1 \leq \min(i_1, \ldots, i_k)$. This is because the MASTER-SLAVE algorithm selects the

smallest index of a schedulable task. Now we show that there exists a schedulable set $\mathcal{I}_1$ of k elements which contains g_1. If $g_1 \in \mathcal{I}_0$ we are done: $\mathcal{I}_1 = \mathcal{I}_0$. Otherwise consider the ordering $i_{j_1}, i_{j_2}, \ldots, i_{j_k}$ in which $\mathcal{I}_0$ is scheduled: we have the set of inequalities:

$$\begin{array}{lll} (L_1) & d_{i_{j_1}} + w_{i_{j_1}} & \leq T \\ (L_2) & d_{i_{j_1}} + d_{i_{j_2}} + w_{i_{j_2}} & \leq T \\ \vdots & \ddots & \vdots \\ (L_k) & d_{i_{j_1}} + d_{i_{j_2}} + \cdots + d_{i_{j_k}} + w_{i_{j_k}} & \leq T \end{array}$$

We replace the first scheduled index i_{j_1} by g_1 to build I_1: we let $I_1 = I_0 \setminus \{i_{j_1}\} \cup \{g_1\}$. The first inequality $c_{g_1} + w_{g_1} \leq T$ holds because g_1 is schedulable. The other inequalities (L_2) to (L_k) still hold because $d_{g_1} \leq d_{i_{j_1}}$. Hence $\mathcal{I}_1 = \{g_1, i_{j_2}, \ldots, i_{j_k}\}$ is schedulable.

ii) Therefore there exists a schedulable set $\mathcal{I}_1 = \{g_1, i'_1, \ldots, i'_{k-1}\}$. For all j, $1 \leq j \leq k_1$, the pair $\{g_1, i'_j\}$ is schedulable, which establishes that the MASTER-SLAVE algorithm does select an index g_2 after selecting g_1. Furthermore, $g_2 \leq \min(i'_1, \ldots, i'_{k-1})$. Now we show that there exists a schedulable set $\mathcal{I}_2$ of k elements which contains g_1 and g_2. If $g_2 \in \{i'_1, \ldots, i'_{k-1}\}$ let $\mathcal{I}_2 = \mathcal{I}_1$. Otherwise, consider the ordering in which $\mathcal{I}_1$ is scheduled. There is no reason that g_1 should be scheduled first in $\mathcal{I}_1$, hence we have two cases:

- g_1 is indeed the first task in $\mathcal{I}_1$ to be communicated by P_0. Let f be the index of the second task in $\mathcal{I}_1$ to be communicated by P_0. Because the pair $\{g_1, g_2\}$ is schedulable, we can replace f by g_2: the first two inequalities (L_1) and (L_2) are satisfied (maybe we have to swap g_1 and g_2, but anyway this pair can be scheduled), and the following $k-2$ inequalities still hold true because $d_{g_2} \leq d_f$.
- The index f of the first task in $\mathcal{I}_1$ to be communicated by P_0 is different from g_1. Then we replace f by g_2: the first inequality (L_1) is satisfied because $\{g_2\}$ is schedulable, and the following $k-1$ inequalities still hold true because $d_{g_2} \leq d_f$.

We conclude that $\mathcal{I}_2$ is schedulable.

iii) For the general case, assume we have a schedulable set $\mathcal{I}_{j-1} = \{g_1, \ldots, g_{j-1}, i'_1, \ldots, i'_{k-j+1}\}$. For every l, $1 \leq lk - j + 1$, the subset $\{g_1, \ldots, g_{j-1}, i'_l\}$ is schedulable, hence the the MASTER-SLAVE algorithm does select an index g_j after selecting $g_1, \ldots, g_{j-1}$. Furthermore, $g_j \leq \min(i'_1, \ldots, i'_{k-j+1})$. Now we show that there exists a schedulable set $\mathcal{I}_j$ of k elements which contains $\{g_1, g_2, \ldots, g_j\}$. If $g_j \in \{i'_1, \ldots, i'_{k-j+1}\}$ let $\mathcal{I}_j = \mathcal{I}_{j-1}$. Otherwise, consider the ordering in which $\mathcal{I}_{j-1}$ is scheduled. We pick up the first task f to be scheduled that is different from the MASTER-SLAVE indices g_1, g_2, $\ldots$, g_{j-1}. We replace this index f by g_j to derive $\mathcal{I}'_j$. We claim that $\mathcal{I}_j$ is schedulable. To see this, consider the scheduling of $\mathcal{I}_{j-1}$, and let l be such that the first l inequalities (L_1) to (L_l) deal with MASTER-SLAVE indices g_{i_1}, g_{i_2}, $\ldots$, g_{i_l}, while (L_{l+1}) deals with index f:

(a) $\{g_{i_1}, \ldots, g_{i_l}, g_j\}$ is schedulable as a subset of $\{g_1, \ldots, g_j\}$: a valid ordering of these $l+1$ tasks provide the first $l+1$ inequalities.

(b) inequalities (L_{l+2}) to (L_k) still hold because $d_{g_j} \leq d_f$.

This establishes Proposition 2. ■

Proof of Theorem 1. We go back to the proof of Theorem 1. First we note that the complexity of the Master-Slave algorithm is quadratic in the number of slaves: each task is processed once, and the test for its insertion in the returned scheduled is easily implemented in time linear in the number of already selected tasks. To see this, simply maintain the list of already selected tasks together with their current termination time. When a new task F_i is inserted, we increment by d_i the termination times of all tasks F_j whose w_j is smaller than w_i, and we check on the fly that these termination times still do not exceed T.

Going back to the original problem, the complexity is thus bounded by $O(n^2p^2)$: after the transformation of Section 3, the number of uni-task slaves is bounded by $O(n.p)$, and the complexity of the greedy algorithm is at most quadratic in that quantity. This concludes the proof of Theorem 1. ■

5 Extensions

In this section we discuss two variants which are natural extensions of the MAX-TASKS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T)$ problem. The first variant deals with a two-port communication model, where the master can communicate to at most two slaves simultaneously. The second variant deals with two masters sharing slaves. Both variants turn out to be NP-complete.

5.1 With a two-port communication model

We state the decision problem associated with the extension to a two-port communication model as follows:

Definition 2 (MAX-TASKS-2PORTS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T, K)$). *Given a set $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of n equal-size independent tasks, a set $\mathcal{W} = \{w_1, w_2, \ldots, w_p\}$ of p execution times, a set $\mathcal{D} = \{d_1, d_2, \ldots, d_p\}$ of p communication delays, a time-bound T, and a task-bound K, is it possible to execute at least K tasks within T time-units on a heterogeneous master-slave platform made up with p slaves P_i of computation time w_i and communication time d_i, $1 \leq i \leq p$, and a master capable of communicating to at most two slaves simultaneously?*

Proposition 3. *MAX-TASKS-2PORTS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T, K)$ is NP-complete.*

Proof. MAX-TASKS-2PORTS$(n, \mathcal{F}, p, \mathcal{W}, \mathcal{D}, T, K)$ clearly belongs to the class NP. We use a reduction from 2-PARTITION [8] to show its completeness. We are given an instance I_1 of 2-PARTITION, i.e. a set of n integers $\mathcal{A} = \{a_1, \ldots, a_n\}$, and we ask whether there exists a partition of $\{1, \ldots, n\}$ into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that

$$\sum_{i \in \mathcal{A}_1} a_i = \sum_{i \in \mathcal{A}_2} a_i \quad ?$$

Let $S = \frac{1}{2} \sum_{i=1}^{n} a_i$.

Given I_1, we build an instance I_2 of MAX-TASKS-2PORTS as follows. We use a set $\mathcal{F}$ of n tasks and $p = n$ slaves with $w_i = 2S$ and $d_i = a_i$ for $1 \leq i \leq n$. We let $T = 3S$ and $K = n$. The size of I_2 is polynomial (even linear) in the size of I_1.

Assume that I_1 has a solution $\mathcal{A}_1$ and $\mathcal{A}_2$. Let the master communicate along its first communication channel with the slaves whose index is in $\mathcal{A}_1$, and along its second communication channel with with the slaves whose index is in $\mathcal{A}_2$. The ordering of the communications is not important. At time S, the last two processors have completed the reception of their message from the master. At time $3S$, each processor has executed one task, hence a solution to I_2.

Assume now that I_2 has a solution. Let $\mathcal{A}_1$ denote the set of indices of those processors which communicate with the master along its first communication channel, and $\mathcal{A}_2$ denote the set of indices of those processors which communicate with the master along its second communication channel. Each processor can execute at most one task, because $2w_i = 4S > T$ for all i. Hence, because $K = n$, each processor executes exactly one task. Therefore the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ represent a partition of $\{1, \ldots, n\}$.

The last processor to receive a message along the first communication channel does complete the execution of its task within T time units, which translates into

$$\sum_{i \in \mathcal{A}_1} d_i + 2S \leq T$$

Similarly, $\sum_{i \in \mathcal{A}_2} d_i + 2S \leq T$. Since $\sum_{i=1}^{n} d_i = 2S$, we get

$$\sum_{i \in \mathcal{A}_1} d_i = \sum_{i \in \mathcal{A}_1} d_2 = S,$$

hence a solution to I_1. ■

5.2 With two masters

We state the decision problem associated with the extension for two masters (see Figure 4) as follows:

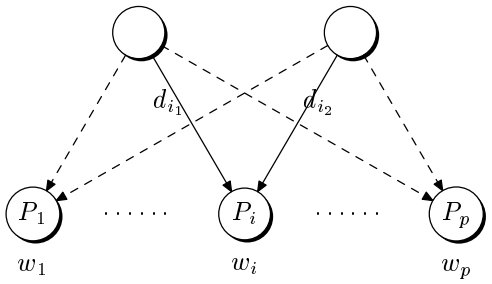


Figure 4: Variant with two masters.

Definition 3 (MAX-TASKS-2MASTERS$(n, \mathcal{F}_1, \mathcal{F}_2, p, \mathcal{W}, \mathcal{D}_1, \mathcal{D}_2, T, K)$**).** *Given a master-slave heterogeneous computing platform made up with*

- *a first master holding a set* $\mathcal{F}_1 = \{F_1, F_2, \ldots, F_n\}$ *of* n *equal-size independent tasks,*
- *a second master possessing a set* $\mathcal{F}_2 = \{F'_1, F'_2, \ldots, F'_n\}$ *of* n *equal-size independent tasks,*
- p *slaves,*
- *a set* $\mathcal{W} = \{w_i, 1 \leq i \leq p\}$ *of execution times of the* p *slaves,*
- *a set* $\mathcal{D}_1 = \{d_{i_1}, 1 \leq i \leq p\}$ *of communication delays from the first master to the* p *slaves,*
- *a set* $\mathcal{D}_2 = \{d_{i_2}, 1 \leq i \leq p\}$ *of communication delays from the second master to the* p *slaves,*

given a time-bound T*, and a task-bound* K*, is it possible to execute at least* K *tasks within* T *time-units on the platform, assuming that each processor is capable of sending to or receiving from at most another processor at any time-step?*

Proposition 4. *MAX-TASKS-2MASTERS*$(n, \mathcal{F}_1, \mathcal{F}_2, p, \mathcal{W}, \mathcal{D}_1, \mathcal{D}_2, T, K)$ *is NP-complete.*

Proof. MAX-TASKS-2MASTERS clearly belongs to NP. Again, we use a reduction from 2-PARTITION to show its completeness. We are given an instance I_1 of 2-PARTITION, i.e. a set of n integers $\mathcal{A} = \{a_1, \ldots, a_n\}$, and we ask whether there exists a partition of $\{1, \ldots, n\}$ into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that

$$\sum_{i \in \mathcal{A}_1} a_i = \sum_{i \in \mathcal{A}_2} a_i \quad ?$$

Let $S = \frac{1}{2} \sum_{i=1}^{n} a_i$.

Given I_1, we build an instance I_2 of MAX-TASKS-2MASTERS as follows. We use $p = n$ slaves with $w_i = 2S$ for all $1 \leq i \leq n$.. We let $d_{i_1} = d_{i_2} = a_i$ for $1 \leq i \leq n$. We let $T = 3S$ and $K = n$. The size of I_2 is polynomial (even linear) in the size of I_1.

The proof that I_1 has a solution if and only if I_2 has a solution is quite similar to that of Proposition 3. ■

6 Related problems

This paper is a follow-on of two previous papers that study the same scheduling problem. In [3], communication links are assumed to be homogeneous. In other words, $d_i = d$ for $1 \leq i \leq p$. We use a matching algorithm to compute the optimal solution in this context. In [2] we deal with master-slave tasking on more general tree-structured networks: in other words, the underlying architecture is a tree, it is no longer restricted to a fork graph. We do not know the complexity of the master-slave scheduling problem on a general tree graph, which seems to be a challenging open problem. Rather than searching a solution to this problem, we characterize in [2] the best steady-state for various operation models (i.e. we neglect the initialization and termination phases).

We classify several related papers along the following three main lines:

Scheduling fork graphs The complexity of scheduling fork graphs has been widely studied in the literature. Given a task graph, the standard macro data-flow model [5, 16] assumes that: (i) processors are homogeneous; (ii) communication delays are paid each time a task and one of its successors are not assigned to the same processor. In this model, scheduling a fork graph with an inifinite number of processors is a polynomial problem [9], while scheduling a fork-join graph is NP-complete [6].

Extensions of the standard macro data-flow model includes the one-port model [13, 14] (just as in this paper, each processor can communicate with at most another processor at a given time-step) and the LogP model [7]. Scheduling fork graphs in the one-port model [1] or in the LogP model [20] remains NP-complete, even for an infinity of resources. Extensions of the result under the LogP model to allow message forwarding are dealt in [21].

Collective communications on heterogeneous platforms Several papers deal with the complexity of collective communications on heterogeneous platforms: broadcast and multicast operations are addressed in [4, 15], gather operations are studied in [11].

Master-slave on the computational grid Master-slave scheduling on the grid can be based on a network-flow approach [19, 18] or on an adaptive strategy [12]. Enabling frameworks to facilitate the implementation of master-slave tasking are described in [10, 22].

7 Conclusion

In this paper, we have shown that master-slave tasking using heterogeneous fork-graphs can be solved in polynomial time, using a non-trivial greedy algorithm. The complexity of this scheduling problem for arbitrary tree graphs is a challenging open problem. A first step to solving this problem would be to study the complexity for a linear chain of processors.

References

[1] O. Beaumont, V. Boudet, and Y. Robert. A realistic model and an efficient heuristic for scheduling with heterogeneous processors. In *HCW'2002, the 11th Heterogeneous Computing Workshop*. IEEE Computer Society Press, 2002.

[2] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In *International Parallel and Distributed Processing Symposium IPDPS'2002*. IEEE Computer Society Press, 2002. Extended version available as LIP Research Report 2001-25.

[3] O. Beaumont, A. Legrand, and Y. Robert. The master-slave paradigm with heterogeneous processors. In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors, *Cluster'2001*, pages 419–426. IEEE Computer Society Press, 2001. Extended version available as LIP Research Report 2001-13.

[4] P.B. Bhat, V.K. Prasanna, and C.S. Raghavendra. Efficient collective communication in distributed heterogeneous systems. In *19th IEEE International Conference on Distributed Computing Systems (ICDCS'99)*. IEEE Computer Society Press, 1999.

[5] P. Chrétienne, E.G. Coffman Jr., J.K. Lenstra, and Z. Liu, editors. *Scheduling Theory and its Applications*. John Wiley and Sons, 1995.

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

[7] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: A practical model of parallel computation. *Communications of the ACM*, 39(11):78–85, 1996.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1991.

[9] Apostolos Gerasoulis and Tao Yang. On the granularity and clustering of directed acyclic task graphs. *IEEE Trans. Parallel and Distributed Systems*, 4(6):686–701, 1993.

[10] J.P Goux, S.Kulkarni, J. Linderoth, and M. Yoder. An enabling framework for master-worker applications on the computational grid. In *Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC'00)*. IEEE Computer Society Press, 2000.

[11] J.-I. Hatta and S. Shibusawa. Scheduling algorithms for efficient gather operations in distributed heterogeneous systems. In *2000 International Conference on Parallel Processing (ICPP'2000)*. IEEE Computer Society Press, 2000.

[12] E. Heymann, M.A. Senar, E. Luque, and M. Livny. Adaptive scheduling for master-worker applications on the computational grid. In R. Buyya and M. Baker, editors, *Grid Computing - GRID 2000*, pages 214–227. Springer-Verlag LNCS 1971, 2000.

[13] S.L. Johnsson and C.-T. Ho. Spanning graphs for optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Computers*, 38(9):1249–1268, 1989.

[14] D.W. Krumme, G. Cybenko, and K.N. Venkataraman. Gossiping in minimal time. *SIAM J. Computing*, 21:111–139, 1992.

[15] R. Libeskind-Hadas, J.R.K. Hartline, P. Boothe, G. Rae, and J. Swisher. On multicast algorithms for heterogeneous networks of workstations. *Journal of Parallel and Distributed Computing*, 61(11):1665–1679, 2001.

[16] C.H. Papadimitriou and M. Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM Journal on Computing*, 19(2):322–328, 1990.

[17] A.L. Rosenberg. Sharing partitionable workloads in heterogeneous NOWs: greedier is not better. In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors, *Cluster Computing 2001*, pages 124–131. IEEE Computer Society Press, 2001.

[18] G. Shao. *Adaptive scheduling of master/worker applications on distributed computational resources.* PhD thesis, Dept. of Computer Science, University Of California at San Diego, 2001.

[19] G. Shao, F. Berman, and R. Wolski. Master/slave computing on the grid. In *Heterogeneous Computing Workshop HCW'00*. IEEE Computer Society Press, 2000.

[20] J. Verriet. *Scheduling with communication for multiprocessor computation.* PhD thesis, Dept. of Computer Science, Utrecht University, 1998.

[21] K.N. Venkataraman W. Zimmermann, W. Löwe. On scheduling send-graphs and receive-graphs under the LogP model. *Information Processing Letters*, to appear.

[22] J.B. Weissman. Scheduling multi-component applications in heterogeneous wide-area networks. In *Heterogeneous Computing Workshop HCW'00*. IEEE Computer Society Press, 2000.