Worst-Case Analysis of Scheduling Heuristics of Parallel Systems

Zhen Liu

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Worst-Case Analysis of Scheduling Heuristics of Parallel Systems

Zhen Liu

N° 2710

Novembre 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Worst-Case Analysis of Scheduling Heuristics of Parallel Systems*

Zhen Liu

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués

Projet MISTRAL

Rapport de recherche n°2710 — Novembre 1995 — 24 pages

Abstract: It is well-known that most scheduling problems arising from parallel systems are NP-hard, even under very special assumptions. Thus various suboptimal algorithms, in particular heuristics, were proposed in the literature. Worst-case error bounds are established in this note for heuristics of makespan minimization of parallel computations. Different parallel computation models are investigated, including interprocessor communication, task duplication, multiprocessor tasks and parallel tasks. Due to the heterogeneity of these systems, scheduling heuristics can be far away from the optimal solutions. The bounds presented here provide insights to the design of scheduling heuristics in order to obtain good performance guarantee.

Key-words: heuristics for multiprocessor scheduling, makespan, schedule length, worst-case error bound, parallel computation, interprocessor communication, task duplication, multiprocessor task, parallel task.

(Résumé : tsvp)

*__Correspondence__: Zhen LIU, INRIA Centre Sophia Antipolis, 2004 Route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France. E-mail: liu@sophia.inria.fr

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Analyse des heuristiques d'ordonnancement des systèmes parallèles

Résumé : Il est bien connu que la plus part des problèmes d'ordonnancement des systèmes parallèles sont NP-difficiles même sous des hypothèses très spécifiques. Ainsi beaucoup d'algorithmes sous optimaux, en particulier les heuristiques, ont été proposés dans la littérature. Des bornes sont établies dans cette note sur les performances des heuristiques d'ordonnancement pour la minimisation de la durée totale d'exécution des programmes parallèles. Différents modèles de calcul parallèle sont envisagés : délai de communication entre processeurs, duplication de tâches, tâches multiprocesseurs et tâches parallèles. A cause de l'hétérogénéité de ces systèmes, les heuristiques d'ordonnancement peuvent s'éloigner de beaucoup des solutions optimales. Les bornes présentées ici fournissent des indications sur la manière dont on peut concevoir des heuristiques d'ordonnancement ayant une garantie de performances.

Mots-clé : heuristiques d'ordonnancement, durée d'ordonnancement, borne de performance, calcul parallèle, communication, duplication de tâches, tâches multiprocesseurs et tâches parallèles.

1 Introduction

Scheduling of parallel computations has received an increasing interest during the last two decades. Researchers from both the communities of parallel computing and of combinatorial optimization have obtained a number of results on the complexity of the problems and optimal solutions. The reader is referred to the books [2, 3] for the recent development in the field.

In this note we are interested in the *makespan* (or *schedule length*) minimization for parallel computations which are represented by *task graphs*. It is now well-known that most such scheduling problems are NP-hard, even under very special assumptions. Thus many suboptimal algorithms, in particular heuristics, were proposed in the literature, see e.g. [4, 8, 9, 10, 13]. In [4] and [9] various heuristics were presented and empirically compared for randomly generated task graphs.

Worst-case error bounds of heuristics were first provided by Graham [6, 7] for the basic task graph model (see below for the definition), and extended to the models with interprocessor communications in [11]. Some other worst case analyses were carried out for more specific models, see remarks and references in [11].

In this short paper, we consider new parallel computation models with task duplications, multiprocessor tasks and parallel tasks. We extend the results of [7, 11] further to these parallel computation models and provide worst case error bounds for heuristics of makespan minimization. Due to the heterogeneity of the systems, scheduling heuristics can be far away from the optimal solutions. The bounds presented in the paper provide insights to the design of scheduling heuristics in order to obtain good performance guarantee. Guided by these analyses, we propose several classes of scheduling heuristics which yield better performance bounds.

The paper is organized as follows. In the next section we define three computation models in detail. In Section 3 we establish the worst case error bounds for these models. We also propose scheduling heuristics which are shown to have better performance guarantee. In Section 4 we present conclusions of our study and remarks on further extensions.

2 Problem Description

We consider the scheduling problem in multiprocessor systems with n identical parallel processors. A job consists of a set of tasks whose executions are governed by given precedence constraints. Denote by $G = (V, E)$ the task graph, where the vertices in $V = \{1, 2, \cdots, |V|\}$ represent the tasks, and the directed edges indicate the precedence constraints: $(i, j) \in E$ implies that task i is an immediate predecessor of task j.

The task processing times are specified by the function $P : V \rightarrow IR^+$, with the meaning that P_i is the processing time of task $i \in V$. In the case of parallel tasks (see definition below), this is the execution time when one processor is used.

In the *basic task graph scheduling problem*, a scheduling policy of the parallel processing system is the one that assigns the tasks to the processors and schedules the executions of these tasks so that the precedence relations are satisfied: for any pair of tasks i, j such that task i is the immediate predecessor of task j, the processing starting times, denoted by S_i and S_j, respectively, have the following relation:

$$S_j \geq S_i + P_i. \tag{1}$$

The schedule is such that a task can be assigned to only one processor, and one processor can execute at most one task at any time.

We shall consider scheduling problems of the following three parallel computation models.

Interprocessor communications and task duplication :

The communications between tasks are described by the function $T : E \rightarrow IR^+$, where $T_{i,j}$ is the amount of information to be transferred by task i to task j when i finishes, $(i, j) \in E$. The communication times, referred to as *interprocessor communication delays*, depend on the system configuration and on the processors where the tasks are processed. Let $C(T_{i,j}, p, q)$ denote the communication time between tasks i and j if they are processed by processors p and q respectively. In the literature, it is sometimes assumed that $C(T_{i,j}, p, q) =$

$T_{i,j}d(p,q)$, where $d(p,q)$ is the communication delay for a unit message from processor p to processor q.

Scheduling policies should satisfy

$$S_j \geq S_i + P_i + C(T_{i,j}, p, q),$$

provided tasks i and j are assigned to processors p and q, respectively.

We assume as usual that the communication time between two tasks assigned to the same processor is negligible. In this case, *task duplication* can be useful to reduce interprocessor communications so as to shorten the makespan. For example, if task i has two successors, say j and k, then, by duplicating i to i and i', the arcs (i,j) and (i,k) are replaced by (i,j) and (i',k) so that, if j (resp. k) is assigned to the same processor as i (resp. i'), there is no communication cost for (i,j) (resp. (i',k)). In general duplication of a task is useful only when the task has at least two successors.

Formally, duplication of task i of task graph $G = (V,E)$ results in a graph $G' = (V',E')$ with

$$V' = V \cup \{i'\}, \quad \text{and} \quad E' = E \cup \{(j,i')|(j,i) \in E\} \cup \{(i',j)|(i,j) \in E_i^\circ\},$$

where E_i° is a nontrivial subset of $E_i \equiv \{(i,j)|(i,j) \in E\}$. Moreover, $P_{i'} = P_i$, $T_{j,i'} = T_{j,i}$ and $T_{i',j} = T_{i,j}$. Note that when $|E_i| \geq 2$, there can be more than one possibility of defining E_i° (in fact, there are exactly $2^{|E_i|} - 2$ possibilities). The choice of a subset of successors of i as the successors of i' takes part in the scheduling decision.

We assume that for each $i \in V$, task i can be duplicated for at most d_i times. Let $d = \max_{i \in V} d_i$, with $1 \leq d \leq n$ by definition.

Multiprocessor tasks :

In this case, the processing of a task requires one or more processors simultaneously. For each $i \in V$, let $m_i \geq 1$ be the number of processors required to execute task i simultaneously. All the m_i processors execute task i for P_i time units. Let $m = \max_{i \in V} m_i$, $m_0 = \min_{i \in V} m_i$, with $1 \leq m_0 \leq m \leq n$ by definition.

We assume in this case that there is no interprocessor communication. The incorporation of interprocessor communication times in the model will be considered in Section 4.

Note that when $m = 1$, the problem reduces to the basic scheduling problem. Scheduling policies should satisfy (1).

Parallel tasks :

In this case, a task can be executed by one or more processors in parallel with linear speed up, i.e., if task $i \in V$ is executed by q processors, the execution time of each of the q processors on the task is P_i/q. For each $i \in V$, let $p_i \geq 1$ be the maximal number of processors that task i can use for parallel execution. Let $p = \max_{i \in V} p_i$, $p_0 = \min_{i \in V} p_i$, with $1 \leq p_0 \leq p \leq n$ by definition.

Again, we assume in this case that there is no interprocessor communication. The incorporation of interprocessor communication times in the model will be considered in Section 4.

Note that when $p = 1$, the problem reduces to the basic scheduling problem. Scheduling policies should satisfy (1) with P_i replaced by P_i/q_i if q_i processors are used to execute task i.

These computation models have recently been studied in the literature. The reader is referred to [3, Chapter 4] for scheduling with task duplications, to [1, 2] for scheduling of multiprocessor tasks, and [5] for scheduling of parallel tasks. These scheduling problems were shown to be NP-hard in general. They are NP-hard even under quite specific assumptions on timing, such as Unit-Execution Time (UET), Unit-Communication Time (UCT), and specific assumptions on task graph structures.

In this paper we will be interested in the performance bounds of heuristics of these scheduling problems. We shall analyze in particular *greedy scheduling policies*. In general, a policy is of greedy type if at any time, some processors are allowed idle only when non of the subset of idle processors can be used to execute a task waiting for execution at that time.

Within the context of multiprocessor tasks, a scheduling policy is of greedy type if as soon as there are enough available processors for at least one *enabled task* (a task is said to be enabled in this case if it has no unfinished predecessor tasks), one of the enabled tasks is assigned to the available processors. For parallel tasks, a scheduling policy is of greedy type if at any time there is processor idling only when there is no enabled tasks. In the case of interprocessor communications, a scheduling policy is of greedy type if whenever a task is selected for the assignment, it is assigned to the processor which starts its execution earliest, and that the task starts as soon as possible.

Many heuristics studied in the literature are of greedy type, e.g., for the scheduling with interprocessor communications, the Earliest Task First policy [8], the Least Schedule Flexibility First policy [13], and the Earliest Ready First policy [10]. List schedules (see e.g. [14, 15]) are also examples of greedy policies.

The scheduling policies described above are *nonpreemptive policies*. In this paper, we shall consider *preemptive policies* as well. When preemptions are allowed, scheduling decisions should also decide when to preempt and resume task executions. Thus, a task is cut into pieces (or more precisely, replaced by a chain of subtasks) which are executed in different time intervals. The subtasks have the same multiprocessor requirement (in the model of multiprocessor tasks) or the same parallelism (in the model of parallel tasks), and the sum of their processing times are equal to the processing time of the corresponding task. In case interprocessor communications are not considered, a task (or more precisely, its subtasks) can also be executed on different processors without penalty. However, if interprocessor communication times are not negligible, specific assumptions have to be made on the communication times among the subtasks in the chain representing a task. In this paper, we will not consider preemptive policies for such a case.

Let $M_\pi(G)$ denote the makespan of task graph G under the (preemptive or nonpreemptive) scheduling policy π on n processors, and $M_*(G)$ be the makespan of an optimal preemptive policy (in the same parallel processing model). Note that, in any of the two parallel processing models without interprocessor communications,

for any given task graph, an optimal preemptive policy has always a (nonstrictly) smaller makespan than an optimal nonpreemptive policy.

The goal of this paper is twofold: to establish upper bounds for the makespan of greedy policies with respect to the optimal makespan, and to propose heuristics having better performance guarantees.

3 Worst-Case Error Bounds

3.1 Basic Task Graph Scheduling

We start our discussion with the results of Graham [6, 7] who analyzed the worst-case error bounds of list schedules for the basic scheduling problem.

Let π be an arbitrary greedy policy for scheduling task graph G subject to the precedence constraint (1). Then,

$$M_\pi(G) \leq \left(2 - \frac{1}{n}\right) M_*(G). \tag{2}$$

This bound will be extended to the new parallel processing models in the next subsections.

3.2 Task Duplications

Consider first the scheduling problem with interprocessor communications. Task duplications are allowed. Let $G = (V, E)$ be the task graph associated with the communication times $C(T_{i,j}, p, q)$ and duplication degrees d_i, with $d = \max_{i \in V} d_i$. We first establish the following result for the general class of greedy policies.

Theorem 1 *Let π be an arbitrary nonpreemptive greedy policy with possibly task duplications for task graph G. Then,*

$$M_\pi(G) \leq \left(d + 1 - \frac{1}{n}\right) M_*^0(G) + C_{\max}, \tag{3}$$

where $M_*^0(G)$ is the makespan by an optimal preemptive policy without considering the interprocessor communication delays, and $C_{\max}$ is the maximal interprocessor communication delay of the chains in G:

$$C_{\max} = \max_{\{(i_1,\cdots,i_k) \mid \forall j=1,\cdots,k-1,\ (i_j,i_{j+1})\in E\}} \sum_{j=1}^{k-1} C\left(T_{i_j,i_{j+1}}, \pi(i_j), \pi(i_{j+1})\right), \quad (4)$$

where $\pi(i)$ denotes the index of the processor on which task i is processed under scheduling policy π.

Proof. The arguments used in the proof are analogous to those of [11]. We provide a detailed proof here, however, for sake of completeness and clearness of presentation of the other results in the paper.

By convention, we consider the processing time of task i to be a half-open interval $[t, t+P_i)$ on the time axis, provided task i starts at time t.

Consider the greedy policy π on n processors for task graph G. Let $G' = (V', E')$ be the task graph resulted from task duplications under π. Let $\pi(i)$ be the index of the processor to which task $i \in V'$ is assigned by π. Denote by S_i (respectively F_i) the time at which task $i \in V'$ starts (respectively finishes) under π. It is trivial that $F_i = S_i + P_i$.

Let $M = M_\pi(G)$. Define a partition $A \bigcup B$ of the set of all points of time in $[0, M)$, where the subset A is the set of all points of time for which all processors are executing some task of G', B is the set of all points of time for which at least one processor is idle (all the processors may be idle, but in this case some interprocessor communication must occur for π is greedy). Observe that A and B are the disjoint union of half-open intervals.

Let task $k_1 \in V'$ be one of the tasks that finish at time M under π. There are two possibilities for the point S_{k_1}:

Case 1: $S_{k_1} \in B$ and S_{k_1} is not a boundary point of B.

Then by the definition of B, there is some processor, say q_1, which for some $\epsilon > 0$ is idle during the time interval $[S_{k_1} - \epsilon, S_{k_1})$. Due to the fact that π is

greedy, there exists an immediate predecessor task, say k_2, of k_1 in G', such that

$$F_{k_2} + C(T_{k_2,k_1}, \pi(k_2), q_1) \geq S_{k_1}. \tag{5}$$

In fact, if such a predecessor task did not exist, task k_1 should have been started on processor q_1 earlier than S_{k_1}.

Case 2: $S_{k_1} \in A$ or S_{k_1} is a boundary point of B.

Suppose that the set $H = \{x \mid 0 \leq x < S_{k_1}, \quad x \in B\} \neq \emptyset$. Let u be the least upper bound of H. It follows from the fact that A and B are the disjoint union of half-open intervals that $u \in A$, and that there is some processor, say q_1, such that for some $\epsilon > 0$, processor q is idle during the time $[u - \epsilon, u)$. It then follows that there exists a task $h(k_1)$, which is identical to k_1, or a predecessor of k_1 in G', such that $S_{h(k_1)} \geq u$ and that there is an immediate predecessor task k_2 of $h(k_1)$ satisfying the relations:

$$S_{k_2} < u, \qquad \text{and} \qquad F_{k_2} + C(T_{k_2,h(k_1)}, \pi(k_2), q_1) \geq u. \tag{6}$$

If this were not true, task $h(k_1)$ (i.e., task k_1 or one of its predecessors in G') should have been started on processor q_1 before time u, according to the greedy rule.

In both cases, we see that either the set $\{x \mid 0 \leq x < S_{k_1}, \quad x \in B\}$ is empty, or there are a processor q_1, a task $h(k_1)$ which is either identical to k_1 or a predecessor of k_1, and an immediate predecessor task k_2 of $h(k_1)$ such that

$$x \in \left[F_{k_2} + C(T_{k_2,h(k_1)}, \pi(k_2), q_1),\ S_{k_1}\right)$$

implies that $x \in A$.

Repeating this construction, we can find a sequence of triplet $\{(k_{j+1}, h(k_j), q_j)\}_{j=1}^{r-1}$ such that

- $h(k_j)$ is identical to k_j or is a predecessor of k_j in G', $j = 1, 2, \cdots, r-1$;

- k_{j+1} is an immediate predecessor of $h(k_j)$ in G', $j = 1, 2, \cdots, r-1$;
- $x \in \left[F_{k_{j+1}} + C(T_{k_{j+1},h(k_j)}, \pi(k_{j+1}), q_j),\ S_{k_j}\right)$ implies that $x \in A$, $j = 1, 2, \cdots, r-1$;
- $\{x \mid 0 \leq x < S_{k_r},\ \ x \in B\} = \emptyset$.

This fact implies that the set

$$\bigcup_{j=1}^{r} \left[S_{k_j},\ F_{k_j} + C(T_{k_j,h(k_{j-1})}, \pi(k_j), q_{j-1})\right)$$

covers B in the sense that for all $x \in B$, there is $1 \leq j \leq r$ such that

$$x \in \left[S_{k_j},\ F_{k_j} + C(T_{k_j,h(k_{j-1})}, \pi(k_j), q_{j-1})\right),$$

where, by convention, $C(T_{k_1,h(k_0)}, \pi(k_1), q_0) = 0$.

Denote by Φ the sum of idling times of π on the n processors during the time interval $[0, M)$. It then follows that

$$\Phi \leq (n-1)\sum_{j=1}^{r} P_{k_j} + n\sum_{j=1}^{r} C(T_{k_j,h(k_{j-1})}, \pi(k_j), q_{j-1}) \leq (n-1)\sum_{j=1}^{r} P_{k_j} + nC_{\max} \quad (7)$$

For all $j = 1, \ldots, r$, let $k_j \in V'$ be either identical to task $k_j^\circ \in V$ or a duplication of it. It is then clear that k_{j+1}° is a predecessor of k_j° in G, $j = 1, 2, \cdots, r-1$. Hence,

$$M_*^0(G) \geq \sum_{j=1}^{r} P_{k_j^\circ} \quad (8)$$

and

$$M_*^0(G) \geq \frac{1}{n}\sum_{i \in V} P_i \quad (9)$$

Relations (7), (8) and (9) readily entail that

$$\begin{aligned} M_\pi(G) &= \frac{1}{n}\left(\Phi + \sum_{i\in V'} P_i\right) \\ &\le \frac{1}{n}\left((n-1)\sum_{j=1}^{r} P_{k_j} + nC_{\max} + \sum_{i\in V'} P_i\right) \\ &\le \frac{1}{n}\left((n-1)\sum_{j=1}^{r} P_{k_j^\circ} + nC_{\max} + \sum_{i\in V} d_i \cdot P_i\right) \\ &\le \frac{1}{n}\left((n-1)M_*^0(G) + nC_{\max} + d\cdot n\cdot M_*^0(G)\right) \\ &\le \left(d+1-\frac{1}{n}\right) M_*^0(G) + C_{\max} \end{aligned}$$

This completes the proof. ∎

As a corollary, when task duplication is not allowed (i.e. $d = 1$), we obtain

Corollary 2 *Let π be an arbitrary greedy policy with no task duplications for task graph G. Then,*

$$M_\pi(G) \le \left(2-\frac{1}{n}\right) M_*^0(G) + C_{\max}, \tag{10}$$

where $M_*^0(G)$ and $C_{\max}$ are as defined in Theorem 1.

Note that $M_*^0(G)$ in (10) is still the optimal preemptive makespan with possible task duplications. Thus, Corollary 2 is in fact the corollary of the proof of Theorem 1. Note also that (10) is a slight improvement of the result of [11] which showed that $M_\pi(G) \le \left(2-\frac{1}{n}\right)\widetilde{M}_*^0(G) + C_{\max}$, where $\widetilde{M}_*^0(G)$ is the optimal preemptive makespan without task duplications.

Comparing Theorem 1 with Corollary 2, one observes that permitting task duplications results in a worse performance guarantee of heuristics. A better approach

of designing heuristics, as far as the performance guarantee is concerned, is to design heuristics with *greedy duplications*, i.e. schedule the tasks first according to a heuristic policy without task duplications, and then modify the scheduling decisions by duplicating some tasks so as to decrease the makespan at each task duplication. Such an approach has clearly the same performance guarantee as that of greedy policy with no task duplications:

Corollary 3 *Let π be an arbitrary greedy policy with greedy task duplications for task graph G. Then,*

$$M_\pi(G) \leq \left(2 - \frac{1}{n}\right) M_*^0(G) + C_{\max}, \tag{11}$$

where $M_*^0(G)$ and $C_{\max}$ are defined as in Theorem 1.

As was pointed out in [11], different bounds are found in the literature for specific scheduling heuristics under more restrictive assumptions for the model with interprocessor communication times. For example, [16] provided an upper bound for the UET-UCT case:

$$M_\pi(G) \leq (3 - 2/n)M_*(G) - 1 + 1/n \tag{12}$$

A slightly less tight bound can be derived from (10) by noting that $C_{\max} \leq M_*(G) - 1$:

$$M_\pi(G) \leq (3 - 1/n)M_*(G) - 1.$$

In [8, 10], bounds similar to (10) were derived for some particular heuristics.

While tighter bounds are available for specific scheduling algorithms, the bounds (3) and (10) are best possible over the general class of greedy policies in the sense that these bounds cannot be replaced by any smaller function of the same variables. In fact, such bounds can be achieved as close as possible by varying the parameters π, G, P and T. The interested reader can construct examples showing this fact in an analogous way to [6, 11].

3.3 Multiprocessor Tasks

Consider now the problem of scheduling multiprocessor tasks. Let $G = (V, E)$ be the task graph associated with the processing times P_i and the numbers m_i of processors required to execute tasks, $i \in V$, with $m = \max_{i \in V} m_i$ and $m_0 = \min_{i \in V} m_i$. We start with the following result.

Lemma 4 *Let π be an arbitrary (preemptive or nonpreemptive) greedy policy for task graph G consisting of multiprocessor tasks. If $m + m_0 \leq n + 1$, then,*

$$M_\pi(G) \leq \left(2 + \frac{m - m_0 - 1}{n - m + 1}\right) M_*(G). \tag{13}$$

Proof. The proof is similar to that of Theorem 1. We will only provide a sketch of the proof. Consider the greedy policy π on n processors for task graph $G = (V, E)$. Let $G' = (V', E')$ be the task graph resulted from $G = (V, E)$ in replacing tasks of G by chains of subtasks corresponding to preemptions (G is identical to G if π is nonpreemptive). Thus, π is a nonpreemptive policy for $G' = (V', E')$. Denote by S_i (resp. F_i) the time at which task $i \in G'$ starts (resp. finishes) under π.

Let $M = M_\pi(G) = M_\pi(G')$. Let A (resp. B) be the subset of points of time in $[0, M)$ such that at most $m - 1$ processors are idle (resp. at least m processors are idle). Note that at most $n - m_0$ processors can be idle due to the fact that π is greedy. Note also that A and B are the disjoint union of half-open intervals.

By mimicking the arguments of the proof of Theorem 1, we can show that there is a sequence of triplet $\{(k_{j+1}, h(k_j), q_j)\}_{j=1}^{r-1}$ such that

- $h(k_j)$ is identical to k_j or is a predecessor of k_j in G', $j = 1, 2, \cdots, r-1$;
- k_{j+1} is an immediate predecessor of $h(k_j)$ in G', $j = 1, 2, \cdots, r-1$;
- the set $\bigcup_{j=1}^{r} [S_{k_j},\ F_{k_j})$ covers B in the sense that for all $x \in B$, there is $1 \leq j \leq r$ such that $x \in [S_{k_j},\ F_{k_j})$.

Let U be the total length of intervals of B. It then follows that

$$U \leq \sum_{j=1}^{r} P_{k_j} \leq M_*(G') = M_*(G) \tag{14}$$

Denote by Φ the sum of idling times of π on the n processors during the time interval $[0, M)$. Then

$$\Phi \leq (m-1)(M-U) + (n-m_0)U = (m-1)M + (n-m-m_0+1)U \tag{15}$$

Finally we note that

$$M_*(G) = M_*(G') \geq \frac{1}{n} \sum_{i \in V} m_i P_i \tag{16}$$

Relations (14), (15) and (16) together with the fact that $m + m_0 \leq n+1$ readily imply that

$$M_\pi(G) = \frac{1}{n}\left(\Phi + \sum_{i \in V} m_i P_i\right) \leq \frac{m-1}{n} M + \frac{n-m-m_0+1}{n} M_*(G) + M_*(G),$$

so that

$$(n-m+1) \cdot M \leq (2n-m-m_0+1) \cdot M_*(G),$$

hence the result. ∎

When the condition $m + m_0 \leq n+1$ is not satisfied, we can insert a task with zero processing time to be executed on a single processor. Then, the new task graph has $m_0 = 1$ so that the condition $m + m_0 \leq n+1$ is satisfied. Consequently,

Corollary 5 *Let π be an arbitrary (preemptive or nonpreemptive) greedy policy for task graph G consisting of multiprocessor tasks. Then,*

$$M_\pi(G) \leq \left(\frac{2n-m}{n-m+1}\right) M_*(G). \tag{17}$$

If $m \leq n/2$, then

$$M_\pi(G) \leq \left(\frac{3n}{n+2}\right) M_*(G). \tag{18}$$

Proof. Inequality (17) comes from Theorem 4 by setting $m_0 = 1$. Since the function $(2n-m)/(n-m+1)$ is increasing in m, $1 \leq m \leq n$, we obtain inequality (18) by setting $m = n/2$ in (17). ∎

Observe that the coefficients in (13) and (17) are increasing in m, and reach n (when $m = n$ and $m_0 = 1$). This is clearly quite bad. However, when $m \leq n/2$, (18) indicates that $M_\pi(G) < 3M_*(G)$. This last fact allows us to establish the following theorem which implies that $M_\pi(G) < 4M_*(G)$ in general.

Theorem 6 *Let π be an arbitrary (preemptive or nonpreemptive) greedy policy for task graph G consisting of multiprocessor tasks. Then,*

$$M_\pi(G) \leq \min\left\{\frac{4n+2}{n+2}, \frac{2n-m}{n-m+1}\right\} \cdot M_*(G). \tag{19}$$

Proof. Let $M = M_\pi(G)$. Let $[a_j, b_j)$, $j = 1, 2, \ldots$, be the time intervals in $[0, M)$ when π executes tasks or subtasks requiring strictly more than $n/2$ processors (i.e. $m_i > n/2$), where $0 \leq a_1 < b_1 < a_2 < b_2 < \cdots$. Without loss of generality, we can assume that in π, tasks or subtasks executing at the time instants $a_1, b_1, a_2, b_2, \ldots$ are preempted and resumed immediately. Let $G' = (V', E')$ be the task graph resulted from $G = (V, E)$ in replacing tasks of G by chains of subtasks corresponding to preemptions in π.

Let $V_2' \subseteq V'$ be the set of tasks running in the time intervals $[a_j, b_j)$, $j = 1, 2, \ldots$. Note that V_2' contains not only tasks requiring strictly more than $n/2$ processors, but also those running in parallel with these tasks. Let G_1' be the subgraph of G' obtained by replacing tasks of V_2' with tasks of zero processing time.

Consider the subschedule π_1 of π for tasks of G'_1. It is clear that π_1 is a greedy (preemptive) heuristics for G'_1. Applying (18) to π_1 entails that

$$M_{\pi_1}(G'_1) \leq \left(\frac{3n}{n+2}\right) M_*(G'_1) \leq \left(\frac{3n}{n+2}\right) M_*(G') = \left(\frac{3n}{n+2}\right) M_*(G). \tag{20}$$

Let K be the total duration of the time intervals $[a_j, b_j)$, $j = 1, 2, \ldots$. Due to the fact that the tasks requiring (strictly) more than $n/2$ processors are executed sequentially in any optimal schedule, we obtain that

$$M_*(G) = M_*(G') \geq K. \tag{21}$$

As a consequence of (20) and (21), we obtain

$$M_\pi(G) = M_\pi(G') = M_{\pi_1}(G'_1) + K \leq \left(\frac{4n+2}{n+2}\right) M_*(G). \tag{22}$$

Since both bounds (17) and (22) hold, we obtain (19) as a consequence. ∎

As in the previous scheduling problems, bound (19) is for arbitrary heuristics. Thus, one can again construct examples as in [6, 11] to show that this bound is best possible over the general class of greedy policies.

3.4 Parallel Tasks

Consider now the last scheduling problem: scheduling of parallel tasks. Let $G = (V, E)$ be the task graph associated with the processing times P_i and the parallelization degrees of tasks p_i, $i \in V$, with $p = \max_{i \in V} p_i$. Our first result concerning this model is the following.

Theorem 7 *Let π be an arbitrary (preemptive or nonpreemptive) greedy policy for task graph G consisting of parallel tasks. Then,*

$$M_\pi(G) \leq \left(p + 1 - \frac{p}{n}\right) M_*(G). \tag{23}$$

Proof. We only provide a sketch of the proof. Consider the greedy policy π on n processors for task graph $G = (V, E)$. Let $G' = (V', E')$ be the task graph resulted from $G = (V, E)$ in replacing tasks of G by chains of subtasks corresponding to preemptions. Thus, π is a nonpreemptive policy for G'. Denote by S_i (resp. F_i) the time at which task $i \in G'$ starts (resp. finishes) under π.

Let $M = M_\pi(G) = M_\pi(G')$. Let A (resp. B) be the subset of points of time in $[0, M)$ such that all processors are busy (resp. at least one processor is idle). Observe that A and B are the disjoint union of half-open intervals.

By mimicking the arguments of the proof of Theorem 1, we can show that there is a sequence of triplet $\{(k_{j+1}, h(k_j), q_j)\}_{j=1}^{r-1}$ such that

- $h(k_j)$ is identical to k_j or is a predecessor of k_j in G', $j = 1, 2, \cdots, r-1$;
- k_{j+1} is an immediate predecessor of $h(k_j)$ in G', $j = 1, 2, \cdots, r-1$;
- the set $\bigcup_{j=1}^{r}[S_{k_j},\ F_{k_j})$ covers B.

Denote by Φ the sum of idling times of π on the n processors during the time interval $[0, M)$. It then follows that

$$\Phi \leq (n-1)\sum_{j=1}^{r} P_{k_j} \tag{24}$$

Let q_i be the maximum number of processors used in an optimal schedule to run task i, $i \in V'$. It is easy to see that

$$M_*(G) = M_*(G') \geq \sum_{j=1}^{r} \frac{P_{k_j}}{q_{k_j}} \geq \frac{1}{p}\sum_{j=1}^{r} P_{k_j}. \tag{25}$$

Thus, inequality (24) implies

$$\Phi \leq p(n-1)M_*(G). \tag{26}$$

Note also that

$$M_*(G) = M_*(G') \geq \frac{1}{n}\sum_{i \in V'} P_i \tag{27}$$

Combining relations (26 and (27) allows us to conclude that

$$M_\pi(G) = \frac{1}{n}\left(\Phi + \sum_{i\in V'} P_i\right) \leq \frac{1}{n}\left(p(n-1)M_*(G) + nM_*(G)\right) = \left(p + 1 - \frac{p}{n}\right) M_*(G),$$

which concludes the proof. ∎

Since the bound established in (23) is for arbitrary heuristics, one can again construct examples as in [6, 11] to show that this bound is best possible over the general class of greedy policies.

Observe that the bound in (23) is much worse than that of Graham (2) for the basic task graph scheduling problem. The coefficient in (23) is linear in p (the maximum number of processors useful for a task). When $p = n$, this coefficient equals n, which is clearly unsatisfactory for large n. This is mostly due to the heterogeneity of parallelism degrees p_i.

If, however, we make restriction in the use of parallel processors, we can obtain better bounds. To this end, we propose a subclass of greedy heuristics, referred to as *bounded parallelism* heuristics.

Bounded parallelism heuristics are two-step scheduling policies. Each policy begins with fixing a real number $\rho \in [1/n, 1]$, referred to as *parallelism ratio*, such that $\rho p \leq n/2$. In the first step, it schedules the tasks using q_i processors for task i, where q_i is an arbitrary strictly positive integer such that $\rho p_i \leq q_i \leq n/2$. In the second step, it increases or decreases the numbers of processors used by some tasks if such a modification of parallelism results in a smaller makespan.

Theorem 8 *Let π be an arbitrary bounded parallelism greedy policy with parallelism ratio $\rho \leq n/(2p)$ for task graph G consisting of parallel tasks. Let $\bar{p} = \min(p, n/2)$. Then,*

$$M_\pi(G) \leq \left(1 + \frac{1}{\rho} + \frac{\bar{p} - (1 + \frac{1}{\rho})}{n - \bar{p} + 1}\right) M_*(G) \leq \left(2 + \frac{1}{\rho}\right)\left(1 - \frac{2}{n+2}\right) M_*(G). \quad (28)$$

Proof. We only need to consider the performance of the bounded parallelism heuristics at their first step. Let π' be the schedule corresponding to the first step of π. Let $G' = (V', E')$ be the task graph resulted from $G = (V, E)$ in replacing tasks of G by chains of subtasks corresponding to preemptions in π'. Let $M = M_{\pi'}(G') = M_{\pi'}(G)$.

By the definition of bounded parallelism greedy policy, π' schedules tasks of G' as if G' is composed of multiprocessor tasks. Let U be the total length of time intervals when at least $\bar{p}$ processors are idle. Let Φ be the sum of idling times of π' on the n processors during the time interval $[0, M)$. Then relation (15) holds with $m = \bar{p}$ and $m_0 = 1$, i.e.

$$\Phi \leq (\bar{p} - 1)(M - U) + (n - 1)U = (\bar{p} - 1)M + (n - \bar{p})U \tag{29}$$

As in the proof of Lemma 4, we can find a sequence of triplet $\{(k_{j+1}, h(k_j), q_j)\}_{j=1}^{r-1}$ such that $\bigcup_{j=1}^{r}[S_{k_j},\ F_{k_j})$ covers the time intervals when at least $\bar{p}$ processors are idle. Let q_i be the number of processors used in π' for task i of G'. It then follows that

$$U \leq \sum_{j=1}^{r} \frac{P_{k_j}}{q_{k_j}} \leq \sum_{j=1}^{r} \frac{P_{k_j}}{\rho p_{k_j}} \leq \frac{1}{\rho} M_*(G') = \frac{1}{\rho} M_*(G). \tag{30}$$

Thus, we obtain from (29) and (30), as well as (27), that

$$M = \frac{1}{n}\left(\Phi + \sum_{i \in V} P_i\right) \leq \frac{1}{n}\left((\bar{p} - 1)M + \frac{n - \bar{p}}{\rho} M_*(G) + n M_*(G)\right),$$

so that

$$M \leq \left(1 + \frac{1}{\rho} + \frac{\bar{p} - (1 + \frac{1}{\rho})}{n - \bar{p} + 1}\right) M_*(G), \tag{31}$$

which proves the first inequality of (28). The second inequality of (28) comes from the easily checked fact that the right-hand-side of (31) is increasing in $\bar{p}$. Thus, the proof is completed by taking $\bar{p} = n/2$ in (31). ∎

Note that the condition $\rho p \leq n/2$ is trivially satisfied if $\rho \leq 1/2$. When $p \leq n/2$, this condition is true for all $\rho \leq 1$.

When $p = 1$, which implies $\rho = 1$, the bound (28) coincides with (2).

In view of Theorem 8, the performance guarantee of bounded parallelism heuristics is bounded by a constant (i.e. $2 + 1/\rho$) times the optimal makespan. Comparing with (23), one sees that such heuristics have a significantly better performance guarantee: the bound is decreased from a linear function of p to constant $2 + 1/\rho$. Moreover, these heuristics still allow one to use a large range of parallel processors for different tasks.

4 Concluding Remarks

We have established worst case error bounds for scheduling heuristics in three parallel processing models. The first model is concerned with the mechanism of task duplications which is useful to reduce the interprocessor communication cost. In the second model, some tasks should be run on several processors simultaneously. In the third one, tasks can be run on one or several processors with linear speed up. In all these models, the task graphs are used to describe the precedence relations between tasks.

The bounds obtained here are extensions of the error bound of Graham [7] on the basic scheduling problem (2). Indeed, the bounds (3), (19) and (23) coincide with (2) when there is no interprocessor communication cost, and no task duplication ($d = 1$), no multiprocessor task ($m = 1$), and no parallel task ($p = 1$). By using the similar arguments as those of [6, 11], it can be shown that these bounds are best possible over the general classes of greedy policies.

The parallel processing models consisting of multiprocessor tasks and parallel tasks can further be extended to include interprocessor communication times (but without task duplications and without preemptions).. In this case, one can show that for nonpreemptive greedy policies, the bounds similar to (10) can be established. More precisely, the coefficient in front of $M_*^0(G)$ at the right hand side of (10), i.e. $2 - 1/n$, should be replaced by the corresponding coefficients of $M_*(G)$ in (19) and (23,28). Whereas the second term $C_{\max}$ remains unchanged.

In comparing (3) with (10), one observes that the coefficient in (3) is much worse. Thus, we proposed two-step heuristics with greedy task duplications which yield better the performance guarantee (11).

For the same reason, we proposed two-step heuristics with bounded parallelism for scheduling task graphs consisting of parallel tasks. As far as the worst-case analysis is concerned, this class of heuristics is particularly efficient and convenient. Indeed, the scheduler can control the error bound of the heuristics by adjusting the parallelism ratio ρ.

Now, comparing (19) with (2), we can also see that having multiprocessor tasks in the model also induce worse bounds. Fortunately, the coefficient is still bounded by a constant (which is 4).

Therefore, we should be very careful in the design of heuristics for such scheduling problems. We believe that better bounds could be obtained for specific heuristics.

References

[1] J. Blazewicz, M. Drozdowski, G. Schmidt, D. de Werra, "Scheduling Independent Multiprocessor Tasks on a Uniform k-Processor System", *Parallel Computing*, Vol. 20, pp. 15-28, 1994.

[2] J. Blazewicz, K. Ecker, G. Schmidt, J. Weglarz, *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag, 1993.

[3] P. Chretienne, E. G. Coffman, J. K. Lenstra, Z. Liu, (Eds.) *Scheduling Theory and Its Applications*, J. Wiley, 1995.

[4] C. Coroyer, Z. Liu, "Effectiveness of Heuristics and Simulated Annealing for the Scheduling of Concurrent Tasks — An Empirical Comparison," *Proc. 5-th International Conference on Parallel Architectures and Languages Europe* (Eds. A. Bode, M. Reeve, G. Wolf), Munich, Germany, June 1993, Springer, pp. 452-463.

[5] J. Du, J. Y-T. Leung, "Complexity of Scheduling Parallel Task Systems", *SIAM J. Disc. Math*, Vol. 2, No. 4, pp. 473-487, 1989.

[6] R. L. Graham, "Bounds for Certain Multiprocessing Anomalies", *Bell Syst. Tech. J.*, Vol. 45, pp. 1563-1581, 1966.

[7] R. L. Graham, "Bounds on Multiprocessing Timing Anomalies", *SIAM J. Appl. Math.*, Vol. 17, No. 2, pp. 416-429, 1969.

[8] J. J. Hwang, "Deterministic Scheduling in Systems with Interprocessor Communication Times", Ph.D. Dissertation, Computer and Information Sciences Department, Univ. of Florida, 1987.

[9] Y. K. Kwok, I. Ahmad, "Scheduling task graphs on multiprocessor: issues and a better approach", preprint.

[10] C. Y. Lee, J. J. Hwang, Y. C. Chow, F. D. Anger, "Multiprocessor Scheduling with Interprocessor Delays", *Oper. Res. Letters*, Vol. 7, No. 3, pp. 141-147, 1988.

[11] Z. Liu, "A Note on Graham's Bound." *Information Processing Letters*, Vol. 36, pp. 1-5, 1990.

[12] Z. Liu, "Scheduling of Random Task Graphs on Parallel Processors", *Proc. of Modeling, ANalysis, and Simulation of Computer and Telecommunication Systems* January 1995, Durham (North Carolina), USA. Eds: P. Dowd and E. Gelenbe, IEEE Computer Society Press.

[13] Z. Liu, J. Labetoulle, "A Heuristic Method for Loading and Scheduling Flexible Manufacturing Systems", *Proc. of the Intern. Conf. Control 88*, London, IEE Conference Publication No.285, pp.195-200, 1988.

[14] Z. Liu, E. Sanlaville, "Preemptive Scheduling with Variable Profile, Precedence Constraints and Due Dates". *Disc. Appl. Math.*, Vol. 58, No. 3, April 1995.

[15] Z. Liu, E. Sanlaville, "Profile Scheduling by List Algorithms." In *Scheduling Theory and Its Applications*, P. Chretienne et al. (Eds.), J. Wiley, 1995, pp. 95-114.

[16] V. J. Rayward-Smith, "UET Scheduling with Unit Interprocessor Communication Delays", Internal Report SYSc80-06, School of Information Systems, Univ. of East Anglia, Norwich, 1986.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399