BICRITERIA PARALLEL TASK SCHEDULING

Fabien Baille* 1, Evripidis Bampis 1, Christian Laforest 1, Christophe Rapine 2
1 LaMI, CNRS UMR 8042, Université d’Evry, Tour Evry 2, 523, Place des Terrasses 91000 Evry, France
{fbaille,bampis,laforest}@lami.univ-evry.fr
2 Laboratoire GILCO, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1, France
rapine@gilco.inpg.fr

Abstract
In this paper, we study the problem of scheduling on \( k \) identical machines a set of parallel tasks with release dates and deadlines in order to optimize simultaneously two conflicting criteria, namely the Size (number of scheduled tasks) and the Weight (sum of the weight of scheduled tasks). We distinguish two variants of the problem: In the first one, tasks have to be scheduled in a contiguous subset of machines, whereas in the second one, they can be scheduled on any subset of machines. In both variants, we show that if the maximal number of required machines is greater than \( \frac{k}{2} \), there is no \((a, b)\)-approximate schedule for any two constants \( a \) and \( b \). Nevertheless, if the number of required machines is no more than \( \frac{k}{2} \) then we propose a \((3\alpha, 3\beta)\)-approximation algorithm for the contiguous case and a \((6\alpha, 6\beta)\)-approximation for the non-contiguous case, where \( \alpha \) (resp. \( \beta \)) denotes the approximation ratio of a monocriterion algorithm maximizing the Size (resp. the Weight).

Keywords: Multiprocessor Scheduling, Bicriteria approximation

1. Introduction - Notations

In this paper, we study the problem of scheduling parallel tasks on a set of \( k \) identical machines. Tasks may require more than a single machine to be executed. They are also subject to release dates and deadlines, representing a time window during which the task has to be scheduled. Each task is associated to a positive weight \( w_t \). We search for the schedule maximizing simultaneously the number of scheduled tasks and their weights. Clearly, the scheduler may reject tasks if this rejection leads to a better global performance for the considered criteria. This problem models various practical situations in which a set of potential users (clients) requires the resources of a system (e.g. the processors of a parallel computer, the communication channels of a telecommunication system).

* Corresponding author.
network, etc.) for a prespecified amount of time within a given time window. These resources are managed by a provider which has to maximize simultaneously the number of satisfied clients (number of scheduled tasks) and its own profit (sum of the weights). It is clear that there is a need of tradeoff solutions.

More formally, let $\mathcal{M}$ be a set of $k \geq 1$ identical machines $M_1, \ldots, M_k$. Let $\sigma_i = (r_i, p_i, d_i, \gamma_i, w_i)$ be a parallel task where $r_i$ denotes the release date, $p_i$ the processing time, $d_i$ the deadline, $\gamma_i$ the number of required machines, and $w_i$ the weight or profit. All these parameters are positive, $\gamma_i$ is an integer satisfying $1 \leq \gamma_i \leq k$ and $d_i \geq r_i + p_i$. A task is said to be sequential if $\gamma_i = 1$, otherwise it is said to be parallel. If $d_i = r_i + p_i$, the task is called interval. An instance $I$ of our problem is then completely described by the couple $(\mathcal{T}, k)$ where $\mathcal{T}$ denotes the set of parallel tasks and $k$ the number of machines.

A task $\sigma_i$ is said to be accepted or scheduled if $\gamma_i$ machines are devoted simultaneously and without interruption (preemption is not allowed) to its execution in the time window $[r_i, d_i]$ for a time $p_i$ (i.e. $\gamma_i$ machines become simultaneously busy for a time $p_i$). At each date, a machine can execute at most one task. A schedule is said to be valid if the above conditions are satisfied for all tasks appearing on it (not all tasks need to be scheduled). We distinguish two variants of the problem: We call the case where every task has to be scheduled on a bloc of $\gamma_i$ consecutive machines, i.e. the case where the indices of the $\gamma_i$ machines devoted to the execution of $\sigma_i$ are consecutive, the contiguous case. Otherwise, we talk about the non-contiguous case.

Two objectives are considered in this paper: The maximization of the number of scheduled tasks, called \textsc{Number} or Size problem in the sequel, and the maximization of the sum of the weights of scheduled tasks, called \textsc{Weight} or \textsc{Profit} problem in the sequel. If $O$ denotes a valid schedule of an instance given by a set of tasks $\mathcal{T}$ and $k$ identical machines, then we denote by $N(O)$ (resp. by $W(O)$) the number of tasks accepted by $O$ (resp. the sum of the weights of the tasks accepted by $O$) and $N^*(\sigma)$ (resp. $W^*(\sigma)$) the number (resp. the weight) of the tasks accepted by an optimal schedule of $\sigma$ on $k$ machines.

Both problems are NP-hard in the general case [Bar-Noy et al., 2001, Bar-Noy et al., 1999], thus, we try to find good approximate solutions. As usual, we consider the notion of approximation ratio. For a criterion $C \in \{N, W\}$, an algorithm $A$ is a $\rho$-approximation algorithm if the following inequality is satisfied for every instance $\sigma$:

$$\rho C(A(\sigma)) \geq C^*(\sigma),$$

where $A(\sigma)$ denotes the schedule obtained by $A$ on instance $\sigma$. Then, $\rho$ is called the approximation ratio of $A$. A schedule is said to be an $(\alpha, \beta)$-approximated schedule if it is a factor $\alpha$ away from the optimum of the first criterion and a factor of $\beta$ away from the optimum of the second one. If an
algorithm returns such a solution for all instances, it is said to be an \((\alpha, \beta)\)-
approximation algorithm. In what follows, the first element of the couple will
denote the approximation ratio for the \textit{Size} and the second will represent the
approximation ratio for the \textit{Weight} criterion.

It is not hard to see that, whereas the \textit{Size} problem is a subproblem of
the \textit{Weight} problem (by letting all the weights to be equal to 1), the two
criteria are conflicting (for example, take an instance composed by a single
task requiring \(k\) machines for its execution and having a huge weight and a
big number of sequential tasks with small weights that have to be scheduled
in the same time interval as the \(k\) machine task). Clearly a good solution for
one criterion may lead to a poor solution for the other. An alternative is to find
(when possible) a solution that simultaneously guarantees good approximation
ratios for the two criteria. This is the goal of this paper.

Related works

For the sequential interval case \((\gamma_i = 1, \forall i = 1, 2, \cdots, n)\), the two mono-
criteria problems are solvable in polynomial time (for the \textit{Size} see [Carlisle
and Lloyd, 1995, Faigle and Nawijn, 1995, Faigle and Nawijn, 1991], while
for the \textit{Weight} see [Arkin and Silverberg, 1987, Carlisle and Lloyd, 1995]).
For the sequential bicriteria interval case a \((\frac{k}{r}, \frac{k}{r})\)-approximation algorithm
\((1 \leq r < k)\) has been proposed in [Baille et al., 2004b]. Note that these cases
are special cases of what [Spieksma, 1999] called the job interval scheduling
problem, where each job consists in a set of intervals and a job is scheduled
when one of its intervals is scheduled.

For the sequential task case, the corresponding monocriteria problems are
NP-hard and can be approximated within a factor of 1.58 (for the \textit{Size}) and
2.54 (for the \textit{Weight}) ([Bar-Noy et al., 1999]). For the bicriteria case, there
is a \((1.58, \frac{k}{r}, 2.54, \frac{k}{r})\)-approximation algorithm ([Baille et al., 2004b]).

For the contiguous parallel interval case, the two monocriteria problems
are NP-hard and there is a \(1.58, \frac{k}{r}\)-approximation algorithm for the \textit{Weight}
(and thus for the \textit{Size} problem) for any fixed \(0 \leq \varepsilon < 1\) and bounded \(\gamma_i\)'s, see
[Chen et al., 2002].

For the non-contiguous parallel intervals case, the \textit{Weight} problem is NP-
hard and can be approximated within a factor of 3 (see [Calinescu et al., 2002]).
In the same paper, it is also shown that the special case where \(\max\{\gamma_i\} \leq \frac{k}{r}\)
is approximable within a factor 2. As far as we know, the NP-hardness of the
\textit{Size} problem is still open. Nevertheless, since the \textit{Size} is a subproblem of
the \textit{Weight} problem, the previous result shows that the \textit{Size} problem is also
approximable within a factor of 3 (and within 2 when \(\max\{\gamma_i\} \leq \frac{k}{r}\)).

For the contiguous parallel tasks case, the monocriteria problems are NP-
hard and approximable within a factor of 35 (see [Bar-Noy et al., 2001]).
For the non-contiguous parallel tasks case, the monocriteria problems are NP-hard and there is a 4-approximation proposed in [Bar-Noy et al., 2001] for the \textsc{Weight} (and thus for the \textsc{Size} problem).

The contribution of this paper concerns the bicriteria approach of the two last variants mentioned above. When \( \max \{ \gamma_i \} \leq \frac{k}{5} \), we give existence theorems of approximated bicriteria schedules implying \((\alpha, \beta)\)-approximation algorithms for the contiguous (in Section 2.1) and for the non-contiguous (in Section 2.2) cases. In Section 3 we exhibit non reachable couples of approximation ratios, when \( \max \{ \gamma_i \} > \frac{k}{5} \).

2. Existence of approximate schedules and approximation algorithms

In this section, we prove that there exists approximated schedules for all the instances where \( \max \{ \gamma_i \} \leq \frac{k}{5} \). We consider the contiguous case in Section 2.1, and the non-contiguous case in Section 2.2. In both cases the general idea is the following: We start with two schedules on \( k \) machines, each one optimal for one of the two criteria. Then, we extract from these schedules \( 3 \) sub-schedules on \( \frac{k}{5} \) machines (assume for simplicity that \( k \) is even), and we choose among them two sub-schedules namely the sub-schedule with maximal \textsc{Size} and the sub-schedule with maximal \textsc{Weight}. Finally, we merge in an appropriate way the chosen sub-schedules in order to obtain a valid schedule on \( k \) machines.

2.1 Contiguous case

Existence of a \((3, 3)\)-approximate schedule.

\textsc{Theorem 1} For every instances \( I = (\sigma, k) \) where \( \max \{ \gamma_i \} \leq \left\lfloor \frac{k}{5} \right\rfloor \) and \( k \geq 3 \), there is a \((3, 3)\)-approximated contiguous schedule.

\textsc{Proof.} Let \( O_N^* \) be a contiguous schedule of \( \sigma \) accepting an optimal number of tasks \( N(O_N^*) = N^*(\sigma) \). Let \( O_W^* \) be a contiguous schedule of \( \sigma \) with an optimal sum of weights of accepted tasks \( W(O_W^*) = W^*(\sigma) \).

In \( O_N^* \), consider the machine \( M_{\left\lfloor \frac{k}{5} \right\rfloor} \). Three valid contiguous sub-schedules of the tasks accepted by \( O_N^* \) on \( \left\lfloor \frac{k}{5} \right\rfloor \) machines can be distinguished:

- The first one consists in the set of tasks exclusively scheduled on the \( \left\lfloor \frac{k}{5} \right\rfloor - 1 \) first machines. Since \( \left\lfloor \frac{k}{5} \right\rfloor - 1 \leq \left\lfloor \frac{k}{5} \right\rfloor \), all these contiguous tasks can be scheduled on \( \left\lfloor \frac{k}{5} \right\rfloor \) machines at the same dates and at the same machines as in \( O_N^* \).

- The second one consists in the set of tasks exclusively scheduled in \( O_N^* \) on the set of machines from \( M_{\left\lfloor \frac{k}{5} \right\rfloor + 1} \) to \( M_k \). Using similar arguments
as those used for the first set, we can show that all these tasks can be scheduled on $\left\lfloor \frac{k}{2} \right\rfloor$ machines.

- The third one consists in the set of tasks using machine $M_{\left\lfloor \frac{k}{2} \right\rfloor}$. All these tasks can be scheduled on $\left\lfloor \frac{k}{2} \right\rfloor$ machines. Indeed, it is sufficient to schedule each task at the same dates as in $O_N^*$. The obtained schedule is feasible since $\max\{\gamma_i\} \leq \left\lfloor \frac{k}{2} \right\rfloor$ and since there is no two tasks executed at the same date ($O_N^*$ is valid thus $M_{\left\lfloor \frac{k}{2} \right\rfloor}$ - and so the sub-schedule - accepts at most one task per date).

Let $N$ be the schedule on $\left\lfloor \frac{k}{2} \right\rfloor$ machines among the 3 above schedules containing the maximum number of tasks. Since all these three schedules form a partition of $O_N^*$, we get $3N(N) \geq N(O_N^*)$.

By partitioning $O_W^*$ in the same manner, we get a schedule $W$ on $\left\lfloor \frac{k}{2} \right\rfloor$ machines satisfying $3W(W) \geq W(O_W^*)$.

Let $O$ be the schedule composed by $N$ on the $\left\lfloor \frac{k}{2} \right\rfloor$ first machines, in which all tasks belonging to $N \cap W$ have been deleted, and by $W$ in the $\left\lfloor \frac{k}{2} \right\rfloor$ next machines. Note that $O$ is a contiguous schedule of $\sigma$ and that when $k$ is odd, there is at least an idle machine during the whole schedule.

Since all the tasks deleted from $N$ are in $W$, we get $3N(O) \geq N(O_N^*)$, and $3W(O) \geq W(O_W^*)$ follows from $W(O) \geq W(W)$. The proposed schedule $O$ is then a $(3, 3)$-approximate contiguous schedule of $\sigma$ on $k$ machines.

**Approximation algorithm.** As a consequence of Theorem 1, we show in this section how to construct in polynomial time a $(3\alpha, 3\beta)$-approximation schedule, mixing an $\alpha$-approximation algorithm for the Size problem and a $\beta$-approximation algorithm for the Weight problem. As a corollary, we obtain an algorithm with a couple of approximation ratios of $(4.74\frac{1}{1-\alpha}, 4.74\frac{1}{1-\beta})$ for the contiguous parallel intervals variant and a $(105, 105)$-approximation for the contiguous parallel tasks variant.

**Corollary 1** Given an $\alpha$-approximation algorithm for the contiguous Size problem and a $\beta$-approximation algorithm for the contiguous Weight problem, there is a $(3\alpha, 3\beta)$-approximation algorithm for the contiguous bicriteria problem on a system of $k \geq 3$ machines.

**Proof.** Let $A_N$ be the $\alpha$-approximation algorithm for the Size and $A_W$ be the $\beta$-approximation algorithm for the Weight (both for the contiguous case) problem.

Let $O$ be the schedule returned by the algorithm presented in the proof of Theorem 1 with the only difference that now we start with $A_N(\sigma)$ and $A_W(\sigma)$, the schedules respectively returned by $A_N$ and $A_W$. Clearly, the algorithm
can be executed in polynomial time. $O$ is a valid and contiguous schedule. Thus, with similar arguments as those used in the proof of Theorem 1, we get: $3N(O) \geq N(A_N(\sigma))$ and $3W(O) \geq W(A_W(\sigma))$. Since $A_N$ is an $\alpha$-approximation algorithm and $A_W$ is a $\beta$-approximation algorithm, we have by definition: $\alpha N(A_N(\sigma)) \geq N^*(\sigma)$ and $\beta W(A_W(\sigma)) \geq W^*(\sigma)$. Therefore, $O$ verifies simultaneously: $3\alpha N(O) \geq N^*(\sigma)$ and $3\beta W(A_W(\sigma)) \geq W^*(\sigma)$. □

Corollary 2 For the contiguous interval scheduling problem there is an algorithm yielding a couple of approximation ratios of $(4.74\frac{1}{1-\epsilon}, 4.74\frac{1}{1-\epsilon})$, for all $\epsilon > 0$ on a system of $k \geq 3$ machines.

Proof. It is a consequence of Corollary 1 using the algorithm of [Chen et al., 2002] ($\alpha = \beta = 1.58\frac{1}{1-\epsilon}$).

Corollary 3 There exists a $(105, 105)$-approximation algorithm, for the contiguous task scheduling problem on a system of $k \geq 3$ machines.

Proof. It is a consequence of Corollary 1 using the algorithm of [Bar-Noy et al., 2001] ($\alpha = \beta = 35$).

2.2 Non-contiguous case

In the previous section, we have shown that all contiguous schedules on $k$ machines can be partitionned into three sub-schedules on $\left\lfloor \frac{k}{3} \right\rfloor$ machines. In this section, we want to use the same idea as the previous section in order to obtain a bicriteria approximation algorithm for the non-contiguous case. However, we cannot apply directly the proof of theorem 1. If we try to do it, since a task can be scheduled on any set of machines, a situation in which a “piece” of task appears in each sub-schedules can occur. That is why we describe, in this section, a specific algorithm which takes into account the non-contiguity of the tasks.

Existence of a $(6, 6)$-approximate schedule. In this section, we show the existence of a $(6, 6)$-approximate schedule for the non-contiguous variant of the problem. To obtain this result, we show how to partition a schedule on $k$ machines into $6$ sub-schedules on $\left\lfloor \frac{k}{6} \right\rfloor$ machines. We consider two cases: in the first one we treat instances in which every task requires at most $\frac{k}{4}$ machines for its execution, while in the second case every task requires more than $\frac{k}{4}$ machines but less than $\left\lfloor \frac{k}{2} \right\rfloor$ machines.

Lemma 1 For every instance $I = (\sigma, k)$ where $\max\{\gamma_i\} \leq \frac{k}{4}$ and $k \geq 4$, the set of accepted tasks by every schedule $O$ of $\sigma$ on $k$ machines can be partitioned into 3 valid sub-schedules on $\left\lfloor \frac{k}{3} \right\rfloor$ machines.
Proof. In order to prove this lemma, we present the following greedy algorithm that returns a partition of $O$ into sub-schedules each one using at most $\left\lfloor \frac{k}{l} \right\rfloor$ machines. Then, we show that the number of sub-schedules is at most three.

We need the following notations. We denote by $O^j_{\left\lfloor \frac{k}{l} \right\rfloor}$ the $j$-th sub-schedule of $O$ on $\left\lfloor \frac{k}{l} \right\rfloor$ machines constructed by our algorithm. We also denote by $l$ the total number of constructed sub-schedules. Given a schedule $O$, we construct an instance $\sigma'$ of intervals in the following way: We replace each task $\sigma_i = (r_i, p_i, d_i, \gamma_i, w_i)$ scheduled between its starting date $s_i$ and its ending date $e_i$ in $O$ by the interval $\sigma'_i = (r'_i = s_i, p'_i = p_i, d'_i = e_i, \gamma'_i = \gamma_i, w'_i = w_i)$.

Here is the description of our greedy algorithm.

- $l = 1$.
- For each $\sigma'_i$ of $\sigma'$ taken in the non-decreasing order of the starting dates $s_i$, do:
  - Find (if it exists) the smallest $j$ ($j \in \{1, \ldots, l\}$) such that $\sigma_i$ can be scheduled into $O^j_{\left\lfloor \frac{k}{l} \right\rfloor}$.
  - If $\sigma'_i$ cannot be included into one of the $l$ existing sub-schedules, then schedule $\sigma'_i$ on a new schedule $O^{l+1}_{\left\lfloor \frac{k}{l} \right\rfloor}$ and set $l = l + 1$.

Clearly, this algorithm outputs $l$ sub-schedules of $O$ on $\left\lfloor \frac{k}{l} \right\rfloor$ machines. By construction, all these sub-schedules are valid and form a partition of $O$. Note also that, by construction of $\sigma'$, each task is executed at the same time interval as in $O$.

Claim: For every schedule $O$ of non-contiguous tasks, the greedy algorithm partitions $O$ into 3 sub-schedules, i.e., $l \leq 3$.

Proof of the claim: By contradiction, suppose that $O^l_{\left\lfloor \frac{k}{l} \right\rfloor}$ is non-empty. Let $\sigma_i$ be the first task scheduled in $O^l_{\left\lfloor \frac{k}{l} \right\rfloor}$. Let $s_i$ and $e_i$ be respectively the starting and ending dates of task $\sigma_i$ in $O$. Let us consider $O^1_{\left\lfloor \frac{k}{l} \right\rfloor}$, $O^2_{\left\lfloor \frac{k}{l} \right\rfloor}$, and $O^3_{\left\lfloor \frac{k}{l} \right\rfloor}$ at the time $s_i$ where the algorithm considers $\sigma_i$ for inclusion.

Since $\sigma_i$ cannot be scheduled neither in $O^1_{\left\lfloor \frac{k}{l} \right\rfloor}$ nor in $O^2_{\left\lfloor \frac{k}{l} \right\rfloor}$ nor in $O^3_{\left\lfloor \frac{k}{l} \right\rfloor}$ and since the tasks are treated in non-decreasing order of their release dates, $s_i$ is the date at which the number of busy machines is maximal in $O^1_{\left\lfloor \frac{k}{l} \right\rfloor}$, $O^2_{\left\lfloor \frac{k}{l} \right\rfloor}$ and $O^3_{\left\lfloor \frac{k}{l} \right\rfloor}$.
At date $s_t$, the number of busy machines in each of the schedules $O^1_t$, $O^2_t$, and $O^3_t$ is at least $\left\lfloor \frac{k}{2} \right\rfloor - \gamma_i + 1$ (otherwise, $\sigma_i$ could be added to one of the above schedules by the algorithm). Since $O^1_t$, $O^2_t$, and $O^3_t$ are disjoint by construction, and since the total number of busy machines in $O$ is at most $k$, we must have at date $s_t$:

$$3 \left( \left\lfloor \frac{k}{2} \right\rfloor - \gamma_i + 1 \right) + \gamma_i \leq k$$

thus

$$\gamma_i \geq \frac{1}{2} \left( 3 \left\lfloor \frac{k}{2} \right\rfloor - k + 3 \right).$$

Since $k \leq \left\lfloor \frac{k}{2} \right\rfloor + \left\lfloor \frac{k}{2} \right\rfloor + 1$, we get

$$\gamma_i \geq \frac{1}{2} \left( 3 \left\lfloor \frac{k}{2} \right\rfloor - \left\lfloor \frac{k}{2} \right\rfloor - \left\lfloor \frac{k}{2} \right\rfloor - 1 + 3 \right)$$

so,

$$\gamma_i \geq \frac{1}{2} \left\lfloor \frac{k}{2} \right\rfloor + 1 \quad (1)$$

If $k$ is even, then there is an integer $p$ such that $k = 2p$ and the inequality (1) becomes:

$$\gamma_i \geq \frac{1}{2} \left\lfloor \frac{2p}{2} \right\rfloor + 1 = \frac{p}{2} + 1 = \frac{k}{4} + 1$$

This is in contradiction with the hypothesis: $\max\{\gamma_i\} \leq \frac{k}{4}$.

If $k$ is odd, then there is an integer $p$ such that $k = 2p + 1$ and (1) becomes:

$$\gamma_i \geq \frac{1}{2} \left\lfloor \frac{2p+1}{2} \right\rfloor + 1 = \frac{1}{2} \left( p + \left\lfloor \frac{1}{2} \right\rfloor \right) + 1 \geq \frac{p}{2} + 1$$

Since $\frac{k}{4} = \frac{2p+1}{4} = \frac{p}{2} + \frac{1}{4}$, we have $\frac{p}{2} + 1 > \frac{p}{2} + \frac{1}{4}$ thus

$$\gamma_i > \frac{k}{4}$$

This also contradicts the fact that $\max\{\gamma_i\} \leq \frac{k}{4}$. This proves the claim that proves the Lemma. \hfill \square

\textbf{Lemma 2} For every instance $I = (\sigma, k)$ for which $\left\lfloor \frac{k}{2} \right\rfloor \geq \max\{\gamma_i\}$ and $\min\{\gamma_i\} > \frac{k}{4}$ and $k \geq 2$, the set of accepted tasks by every schedule $O$ of $\sigma$ on $k$ machines can be partitioned into 3 sub-schedules on $\left\lfloor \frac{k}{2} \right\rfloor$ machines.
Bicriteria parallel task scheduling

Proof. Since $\frac{k}{4} + 1$ is the smallest value for a $\gamma_i$, at most 3 tasks are scheduled by the $k$ machines at any date.

Thus, by partitioning $O$ with the greedy algorithm of the proof of Lemma 1, we can show that the fourth sub-schedule is necessarily empty, otherwise, the hypothesis of the validity of $O$ is violated.

**Theorem 2.** For every instance $I = (\sigma, k)$ satisfying $\max\{\gamma_i\} \leq \left\lfloor \frac{k}{2} \right\rfloor$ and $k \geq 4$, there exists a $(6,6)$-approximated schedule.

Proof. Let $O_N^\ast$ (resp. $O_W^\ast$) be an optimal schedule of $\sigma$ on $k$ machines for the *Number* (resp. the *Weight*) criterion. We partition the tasks scheduled by $O_N^\ast$ (and $O_W^\ast$) into *big* and *small* tasks. A task is called big (resp. small) if $\gamma_i > \frac{k}{4}$ (resp. $\gamma_i \leq \frac{k}{4}$).

Let $S_N^\ast$ (resp. $S_W^\ast$) be the schedule on $k$ machines resulting from the deletion of big tasks from $O_N^\ast$ (resp. $O_W^\ast$) and let $B_N^\ast$ (resp. $B_W^\ast$) be the schedule on $k$ machines resulting from the deletion of small tasks from $O_N^\ast$ (resp. $O_W^\ast$).

By construction, $B_N^\ast \cup S_N^\ast = O_N^\ast$ and $B_N^\ast \cap S_N^\ast = \emptyset$. For the same reasons $B_W^\ast$ and $S_W^\ast$ form a partition of $O_W^\ast$.

Since $B_N^\ast$ and $B_W^\ast$ contain, by construction, only tasks such that $\gamma_i > \frac{k}{4}$ and since $\max\{\gamma_i\} \leq \left\lfloor \frac{k}{2} \right\rfloor$, by Lemma 2, we can partition them into 3 sub-schedules on $\left\lfloor \frac{k}{4} \right\rfloor$ machines.

Since $S_N^\ast$ and $S_W^\ast$ only contain tasks such that $\gamma_i \leq \frac{k}{4}$, by Lemma 1 we can partition them into 3 sub-schedules on $\left\lfloor \frac{k}{4} \right\rfloor$ machines.

Thus, we have partitioned both $O_N^\ast$ and $O_W^\ast$ into 6 sub-schedules on $\left\lfloor \frac{k}{2} \right\rfloor$ machines. Let $M_N$ be the sub-schedule of the partition of $O_N^\ast$ with the maximum *Size*. Let $M_W$ be the sub-schedule of the partition of $O_W^\ast$ with the maximum *Weight*.

We have $6N(M_N) \geq N(O_N^\ast)$ and $6W(M_W) \geq W(O_W^\ast)$.

Let $O$ be the (valid) schedule on $k$ machines executing on the $\left\lfloor \frac{k}{4} \right\rfloor$ first machines the tasks of $M_W$ at the same dates as in $O_W^\ast$ and the tasks of $M_N - M_W$ on the $\left\lfloor \frac{k}{2} \right\rfloor$ next machines.

Clearly, we have $6W(O) \geq W(O_W^\ast)$ because $W(M_W) \leq W(O)$.

Since the tasks contained in $M_W \cap M_N$ appear also in $M_W$, each task of $M_N$ appears in $O$. Thus $N(M_N) \leq N(O)$ and $6N(O) \geq N(O_N^\ast)$.

**Approximation algorithm for the non-contiguous case.** We show here that applying the same partitioning as the one described in the proof of Theorem 2 starting from two approximate schedules (instead of two optimal ones) implies that a bicriteria approximate schedule can be computed in polynomial time.
Given an instance \( I = (\sigma, k) \) with \( k \geq 4 \),
an \( \alpha \)-approximation algorithm for the non-contiguous \textsc{size} problem and a \( \beta \)-approximation algorithm for the non-contiguous \textsc{weight} problem, a polynomial time \( (6\alpha, 6\beta) \)-approximation algorithm for the bicriteria non-contiguous problem exists.

\textbf{Proof.} Let \( A_N \) be the \( \alpha \)-approximation algorithm for the \textsc{size} problem and \( A_W \) be the \( \beta \)-approximation algorithm for the \textsc{weight} problem.

1. Partition (in the same manner as mentioned in Lemmas 1 and 2) both \( A_N(\sigma) \) and \( A_W(\sigma) \) into 6 sub-schedules on \( \left\lfloor \frac{k}{2} \right\rfloor \) machines.

2. Choose the schedule having the maximum \textit{Size} among the sub-schedules of the partition of \( A_N(\sigma) \) and the schedule having the maximum \textit{Weight} among the sub-schedules in the partition of \( A_W(\sigma) \).

3. Merge (i.e. make the union) these two schedules deleting redundancies.

Let \( O \) be the schedule returned by the above algorithm. Using similar arguments as those used in the proof of Theorem 2, we easily get: \( 6N(O) \geq N(A_N(\sigma)) \) and \( 6W(O) \geq W(A_W(\sigma)) \). Thus, since algorithm \( A_N \) is an \( \alpha \)-approximation algorithm and \( A_W \) is a \( \beta \)-approximation algorithm, we have by definition: \( \alpha N(A_N(\sigma)) \geq N^*(\sigma) \) and \( \beta W(A_W(\sigma)) \geq W^*(\sigma) \). Therefore, \( O \) verifies simultaneously: \( 6\alpha N(O) \geq N^*(\sigma) \) and \( 6\beta W(A_W(\sigma)) \geq W^*(\sigma) \). Note that the algorithm can be executed in polynomial time.

\textbf{Corollary 5} For the non-contiguous parallel interval scheduling problem with \( k \geq 4 \), there is an algorithm yielding a couple of approximation ratios of (12, 12).

\textbf{Proof.} It is an application of Corollary 4, using the 2-approximation algorithm for the \textsc{weight} problem described in \cite{Calinescu2002} \((\alpha = \beta = 2)\).

\textbf{Corollary 6} There exists an algorithm yielding a couple of approximation ratios of (24, 24) for the non-contiguous parallel task scheduling problem with \( k \geq 4 \).

\textbf{Proof.} It is an application of Corollary 4, using the 4-approximation algorithm for the \textsc{weight} problem described in \cite{Bar-Noy2001} \((\alpha = \beta = 4)\).
3. Negative results

In the previous section, we have analyzed the case where a task could not require more than half of the total number of machines. In this section, we show that when there is no restriction on the number of required machines, some couples of approximation ratios are unreachable.

**Theorem 3** If the number of required machines $\gamma_i$ is greater than $\frac{k}{2}$ then for all constants $\alpha$ and $\beta$, there exists instances $I = (\sigma, k)$ with $k \geq 3$ for which there is no $(\alpha, \beta)$-approximation schedule.

**Proof.** Let us consider $k > 2$ machines and the following set of intervals depending on an integer $W$. All the intervals of the instance require $\frac{k}{2}$ machines. They are of two types:

- One long interval $(r_1 = 0, p_1 = W, d_1 = W, w_1 = W)$.
- $W$ short intervals $(r_i = i, p_i = 1, d_i = i + 1, w_i = 1)$ for $i = 2, \cdots, W + 1$.

We have $S^*(\sigma) = W$ (schedule all the shorts intervals), and $W^*(\sigma) = W$ (schedule the unique long interval). Thus, an optimal schedule for the SIZE problem has a weight $W$ away from the optimal. Conversely, an optimal schedule for the WEIGHT problem has a size $W$ away from the optimal. Since $W$ is unbounded, and since the short intervals intersect the long ones, any constant approximated schedule for one metric can be as far as possible from the optimum for the other metric. This proves the theorem.

4. Conclusion and future work

In this paper, we have studied the problem of scheduling parallel tasks in order to optimize two conflicting criteria namely the Size and the Weight. When the maximum number of required machines is no more than $\frac{k}{2}$, we proposed approximation results for the contiguous and the non-contiguous variants of the problem. When $\gamma_i > \frac{k}{2}$, we show that there is no $(\alpha, \beta)$-approximate schedule for any couple of fixed constants $\alpha$ and $\beta$. It is interesting to know whether it is possible to improve the result on the non-contiguous case by obtaining the same couple of approximation ratios as in the contiguous case. Furthermore, the extension of our approach to other task models like the degradable processing time model (see [Baille et al., 2004a]) and malleable task model (see [Mounie et al., 1999]) should be examined in the considered bicriteria model.

References

12


