An extention of the 5/2-approximation algorithm using oracle

Marin bougeret[‡], Pierre-François Dutot, and Denis Trystram

LIG, Grenoble University, France
bougeret,dutot,trystram@imag.fr

[‡]This work is supported by DGA-CNRS

Research report

Abstract

In this paper we consider the Multiple Cluster Scheduling Problem (MCSP). The objective is to schedule parallel jobs on cluster having different sizes (*i.e.* number of processors) or different speeds. We provide a $\frac{5}{2}$-approximation algorithm (using an oracle guess), improving thus the result of [Bougeret et al., 2010a] that requires the assumption that all the jobs fit on all the clusters. Moreover, this result also hold for clusters having same size but different speeds. Notice that our algorithm even apply for "contiguous scheduling", where jobs must be allocated on contiguous indexes of processors (*i.e.* jobs are rectangles).

1 Introduction

In the grid computing paradigm, several clusters share their computing resources in order to distribute the workload. Each cluster is a set of identical processors connected by a local interconnection network. Jobs are submitted in successive packets called batches. The objective is to minimize the time when all the jobs of a batch are completed, then, the next batch of jobs can be processed. Many such computational grid systems are available all over the world, and the efficient management of the resources is a crucial problem.

Let us now introduce the Multiple Cluster Scheduling Problem (MCSP) more formally. We are given n parallel jobs $J = \{J_1, \ldots, J_n\}$ and N clusters $Cl_1, \ldots, Cl_N$. Each job J_j is described by a processing time p_j and a width q_j (the number of required processors). The area of a job J_j is $q_j p_j$, consequently the total area of a set of jobs X is defined as $A(X) := \sum_{J_j \in X} p_j q_j$. In the same way we define $Q(X) := \sum_{J_j \in X} q_j$ and $P(X) := \sum_{J_j \in X} p_j$. A cluster Cl_ℓ has m_ℓ identical processors, each of them running with speed s_ℓ. A job J_j is only allowed to be scheduled within one cluster, its processing time in cluster Cl_ℓ is $p_j^\ell := \frac{p_j}{s_\ell}$ if $q_j \leq m_\ell$ else $p_j^\ell = \infty$. We assume the clusters to be sorted by non-decreasing order of their number of processors (or machines), i.e. $m_1 \leq m_2 \leq \ldots \leq m_N$. Furthermore we assume $\min_\ell s_\ell = 1$ and define $p_{\max} := \frac{\max_j p_j}{\min_\ell s_\ell} = \max_j p_j$. The objective is to find a non-preemptive schedule of the jobs into the clusters minimizing the makespan, i.e. the latest finishing time of a job.

MCSP is closely related to Multiple Strip Packing (MSP) problem where a cluster can be seen as a strip and a job as a rectangle. However, there is an additional constraint in MSP, since packing a rectangle corresponds to schedule a job in MCSP using consecutive addresses of processors (in other words, the allocation must be contiguous). Thus, results for MCSP do not necessarily apply to MSP as the schedule may be not contiguous. Obviously, a solution for MSP is a (feasible) solution for MCSP. However, approximation ratios are not preserved because the optimal value for MSP is an upper bound of the optimal value for MCSP.

Related work

In the case if $N = 1$ the problem is identical to scheduling n parallel jobs on m identical machines. Here the contiguous case corresponds directly to strip packing. For the case that the number of machines is polynomially bounded in the number of jobs a $(1.5 + \epsilon)$-approximation for the contiguous case and a $(1 + \epsilon)$-approximation for the non-contiguous case where given in [Jansen and Thöle, 2008]. For strip packing Coffman *et al.* gave in [Coffman Jr et al., 1980] an overview about performance bounds for shelf-orientated algorithms as $NFDH$ (Next Fit Decreasing Height) and $FFDH$ (First Fit Decreasing Height), that have an absolute ratio of 3, and 2.7, respectively. Schiermeyer [Schiermeyer, 1994] and Steinberg [Steinberg, 1997] presented independently an algorithm for strip packing with absolute ratio 2. This result was recently improved by Harren *et al.*, in [Harren et al., 2010] they presented an algorithm with absolute ratio $5/3 + \epsilon$. A further important result is an AFPTAS for strip packing with additive constant $\mathcal{O}(1/\epsilon^2 h_{\max})$ of Kenyon and Rémila [Kenyon and Rémila, 2000], where $h_{\max}$ denotes the height of the tallest rectangle (i.e. the length of the longest job). This constant was improved by Jansen and Solis-Oba, who presented in [Jansen and Solis-Oba, 2007] an APTAS with additive constant $h_{\max}$.

For MCSP with clusters of identical sizes and speeds, i.e. $s_\ell = 1$ and $m_\ell = m$ for all $\ell \in \{1, \ldots, N\}$, Zhuk [Zhuk, 2006] showed that MSP has no polynomial time approximation algorithm (unless $P = NP$) with abolute ratio better than 2. The remark of [Ye et al., 2009] that consists in applying a $PTAS$ to balance the area of the jobs among the clusters, provide a $2 + 2\epsilon$-appoximation algorithm whose complexity is in $O(f(\epsilon)g)$, where f is the complexity of a $PTAS$ for the classical $P||C_{max}$ problem with precision ϵ, and g the complexity of Steinberg's algorithm [Steinberg, 1997]. For the non-contiguous case, we proposed recently a low cost $5/2$-approximation in [Bougeret et al., 2010b].

For MCSP with clusters of different sizes but identical speeds, Schwiegelshohn *et al.* [Schwiegelshohn et al., 2008] achieved ratio 3 for a version of parallel job scheduling in grids without release times, and ratio 5 with release times. We recently get a fast $5/2$-approximation in [Bougeret et al., 2010a] that only apply when all the jobs fit in all the clusters, i.e. when $\max_j q_j \leq \min_\ell m_\ell$. As explained in [Bougeret et al., 2010a], the previous remark to get a $2+2\epsilon$ ratio can be extended (using a $PTAS$ for $Q||C_{max}$) for MCSP with clusters of different sizes but identical speeds *only* under this hypothesis $\max_j q_j \leq \min_\ell m_\ell$. For the general problem where a job may not fit into a cluster, we would have to use a $PTAS$ for the problem of scheduling jobs with *inclusive processing set restrictions* where machines have different speeds. However there is up to now (see the survey [Leung and Li, 2008]) only $PTAS$ for scheduling nested jobs when the machines have the same speed [Li and Wang, 2010] (or $FPTAS$ for the $Rm||C_{max}$ problem [Horowitz and Sahni, 1976]). Thus, there is up to now no polynomial algorithm with ratio better than 3 for the MCSP problem where clusters have the same speed.

To the best of our knowledge, there are no specific results for MCSP with clusters of same sizes and different speeds. Notice however that, again, the remark of [Ye et al., 2009] applies for the MCSP with clusters of same sizes and different speeds.

Our results

We present in Section 2 a pure "combinatorial" (without linear programming) algorithm of ratio $5/2$ that applies for MCSP when all the processors have the same speed, but the m_ℓ may differ. This improves the previous 3-approximation algorithms cited before (which moreover only applies for non-contiguous scheduling). That algorithm can be adapted to MCSP with clusters of the same size but with different speed values (see the appendix). The algorithm needs an oracle guess which will require (when enumerating all the possible answers of the oracle) to enumerate $O(\min(n, \frac{2\sum_{\ell=1}^{N} m_\ell}{N m_1})^N)$ possibilities in the worst case. From the methodological point of view, such an combinatorial algorithm with oracle may emphasize what is critical in the problem and provide insight for the considered problem. From the point of view of practical applications, even if the previous complexity is only polynomial for fixed N, this algorithm is faster than using approximation schemes for $Rm||C_{max}$ with $\epsilon = \frac{1}{4}$. Moreover, this running time can be improved (see Section 2.4) using classical rounding techniques. Since we assign each jobs to processors of consecutive addresses, all these results also apply for MSP (*i.e* for contiguous version).

2 A 5/2-Approximation for MCSP where clusters have the same speed

2.1 Main Ideas and Algorithm

In this section we study the problem of scheduling rigid jobs on clusters that have different numbers of processors, supposing that all clusters run at the same speed. We provide a 5/2-approximation that both applies for job scheduling and multiple strip packing.

The main idea of the algorithm is to schedule a set π_ℓ in each cluster Cl_ℓ, starting from Cl_1, such that $\sum_{\ell=1}^{\ell_0} A(\pi_\ell) \geq \sum_{\ell=1}^{\ell_0} A(\pi^*_\ell)$ for any ℓ_0, where π^*_ℓ is the set scheduled in Cl_ℓ in a fixed optimal solution. As the optimal value is not known, we use the classical dual approximation technique [Hochbaum and Shmoys, 1988] and denote by $T \in [p_{\max}, np_{\max}]$ the current value of the guess (of the non-contiguous optimal). Since the clusters may have different numbers of processors we define $Fit^\ell := \{J_j | q_j \leq m_\ell\}$, the set of jobs that fit in Cl_ℓ. Moreover we define $Lg := \{J_j | p_j \geq \frac{T}{2}\}$ the set of long jobs and $Wd^\ell := \{J_j | m_\ell \geq q_j \geq \frac{m_\ell}{2}\}$ the set jobs that are wide in Cl_ℓ. A way to guarantee the area domination is to select for each ℓ a set of jobs X such that $A(X) \geq m_\ell T$, and to schedule it below $\frac{5T}{2}$.
If $m_\ell T \leq A(X) \leq \frac{5}{4} m_\ell T$, we already know according to Steinberg's theorem [Steinberg, 1997] that X can be scheduled below $\frac{5T}{2}$ in polynomial time. For the cases where $A(X) > \frac{5}{4} m_\ell T$, we have to proceed differently. By cleverly choosing the set X as a union of a subset *wide* $\subset Wd^\ell$ of the wide rectangles and a subset *select* $\subset (Fit^\ell \setminus Wd^\ell)$ we make sure that we have only a very small number of critical jobs to handle in this case,and that X can be scheduled in $\frac{5T}{2}$. For example, with $X = \{J_1, J_2, J_3, J_4\}$, with $q_1 = q_2 = q_3 = \frac{m_\ell}{2} + \epsilon$, $p_1 = \frac{T}{2} + \epsilon$, $p_2 = p_3 = \frac{T}{2}$, $p_4 = T$ and $q_4 = \frac{m_\ell}{2}$, we have $A(X') < T$ for any $X' \subsetneq X$ and X cannot be scheduled in $\frac{5T}{2}$.

It appears that the only restriction we need for defining X is to have $P(X \cap Wd^\ell \leq \frac{3T}{2})$. Thus, it is possible that for a given ℓ_0 we have $A(X) < m_{\ell_0} T$ with $Fit^{\ell_0} \neq \emptyset, Fit^{\ell_0} \subset Wd^{\ell_0}$. In this case, to ensure the area domination we also need an area domination for the wide jobs, that is $\sum_{\ell=1}^{\ell_0} A(\pi_\ell \cap Wd^{\ell_0}) \geq \sum_{\ell=1}^{\ell_0} A(\pi^*_\ell \cap Wd^{\ell_0})$.

This area domination for the wide jobs could be guaranteed by scheduling for each cluster the widest possible job, until reaching T. However, by using only a widest first policy we could overlap $\frac{3T}{2}$ because of "big" jobs of $Wd^{\ell_0} \cap Lg$. Thus, by guessing for each cluster the (potential) unique big job scheduled in this cluster in the optimal, we can use the widest first policy with jobs of $Wd^{\ell_0} \setminus Lg$ and avoid the previous problem.

So briefly described our algorithm works as follows. We first enumerate the unique big job for each cluster. Then, for each cluster (starting with Cl_1), we select during phase 1 some wide jobs with widest first policy from $(Wd^\ell \setminus Lg)$ and add them to *wide* until we have a total of length $P(wide)$ at least T and at most $\frac{3T}{2}$. The only way to schedule the wide jobs is one after another. So we sort the jobs in non-increasing order of their widths and schedule them bottom-left justified starting with the widest.
In phase 2 we add jobs with largest area from $(Fit^\ell \setminus Wd^\ell)$ to *select* as long as $A(wide \cup select) < Tm_\ell$. As mentioned before in phase 3 we reschedule $wide \cup select$ with Steinberg if possible or use the fact that we have selected only few critical jobs.

Algorithm 1

guess $J_{j^*_\ell} \in Lg \cap Wd^\ell \cap \pi^*_l$ for all $\ell \in \{1, \ldots, N\}$ and remove them from the initial set of jobs
for $\ell = 1$ to N **do**
————————- phase 1 ————————-
$wide \leftarrow \emptyset$
add $J_{j^*_\ell}$ to $wide$
while $((P(wide) < T)$ and $(Wd^\ell \setminus Lg \neq \emptyset))$ **do**
$J_{j_0} \leftarrow$ widest job of $Wd^\ell \setminus Lg$
add J_{j_0} to $wide$
end while
Reschedule jobs of *wide* sequentially in non-increasing order of their width starting with the widest bottom-left justified (see Figure 2).
————————- phase 2 ————————-
$select \leftarrow \emptyset$
while $((A(wide) + A(select) < m_\ell T)$ and $(Fit^\ell \setminus Wd^\ell \neq \emptyset))$ **do**
$J_{j_0} \leftarrow$ job of $Fit^\ell \setminus Wd^\ell$ with largest area
add J_{j_0} to $select$
end while
————————- phase 3 ————————-
if $A(wide) + A(select) \leq \frac{5}{4} m_\ell T$ **then**
reschedule $wide \cup select$ using Steinberg [Steinberg, 1997] algorithm
else
schedule *select* using lemma 1
end if
end for
if there is an unscheduled job **then**
reject T
end if

2.2 Analysis

Given that we use the dual approximation technique, we have to prove that either Algorithm 1 produces a schedule of makespan lower than $\frac{5T}{2}$, or that $T < Opt$ (in this case we say that T is rejected), where *Opt* denotes the non-contiguous optimal value . For the sake of simplicity, we do not mention everywhere the "reject" instruction in the algorithm. Thus we assume throughout the section that $T \geq Opt$, and it is implicit that if during execution one of the claimed properties is wrong then T should be rejected.

We start by proving that the set of selected jobs assigned by our algorithm to a cluster Cl_ℓ can always be scheduled in $\frac{5T}{2}$.

Lemma 1. *Let $\ell \in \{1, \ldots, N\}$, $p \in \mathbb{N}$ and let wide and select $= \{J_1, \ldots, J_p\}$ be the set of jobs selected for Cl_ℓ in phase 1 and 2. There exists a feasible schedule of $wide \cup select$ into Cl_ℓ with a makespan lower than $\frac{5T}{2}$.*

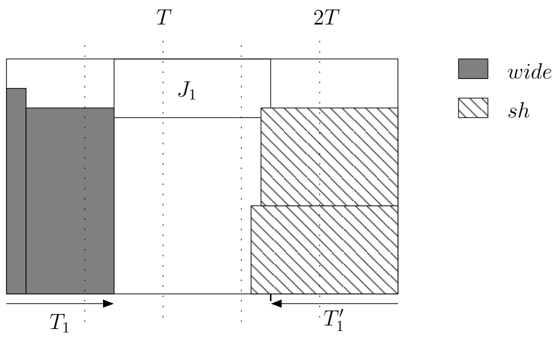

Figure 2: Example of schedule built in Lemma 1

Proof Since it is always possible to schedule $wide \cup select$ with the algorithm of Steinberg [Steinberg, 1997] into the designated space if $A(wide) \cup A(select) \le \frac{5T}{4}$, we only consider cases with $A(wide) \cup A(select) > \frac{5T}{4}$ in phase 3. If $A(wide) \ge m_\ell T$, the algorithm skips phase 2 and consequently $A(select) = \emptyset$ and since $P(wide) \le 3T/2$ by construction, all jobs are scheduled below $\frac{5T}{2}$.
Let us assume $A(wide) < m_\ell T$. Since $A(wide) + A(select) > \frac{5}{4} m_\ell T$ the last job J_p added to *select* has total area strictly larger than $\frac{m_\ell T}{4}$ (otherwise the algorithm would have stopped before). This implies $A(J_j) > \frac{m_\ell T}{4}$ for all $j \in \{1, \ldots, p\}$ and thus $p \le 4$. Since $J_j \notin Wd^\ell$ we furthermore conclude $q_j > \frac{m_\ell}{4}$ and $p_j > \frac{T}{2}$. We add now the jobs in *select* in the following way (see Figure 2):

- Sort the jobs in *select* by decreasing width.
- Starting at time $\frac{5T}{2}$ schedule as many jobs of *select* as possible in the reverse direction using widest first policy bottom-right justified. Let *sh* denote this set of jobs, and let α denote the number of jobs in *sh*.
- If $\alpha < p$, schedule J_j ($\alpha < j \le p$) top justified into Cl_ℓ as soon as possible (i.e. at time $t_j := \min\{t | q_j$ consecutive processors are idle in $Cl_\ell\}$).

Since $Wd^\ell \cap select = \emptyset$, we have $\alpha \ge 2$. Since $P(wide) \le \frac{3T}{2}$ the schedule is feasible for $p \le 2$. Consequently we only study two other cases, namely $p = 3$ or $p = 4$.
Let $p = 3$ and let $J_1 \in select \backslash sh$. Assume that the previous algorithm fails when scheduling J_1, implying that J_1 scheduled at time t_1 intersects *sh*. With $t_1' := \frac{5T}{2} - t_1 - p_1$ we conclude

$$A(wide \cup sh) > t_1(m_\ell - q_1) + t_1'(m_\ell - q_1) + (Q(sh) - (m_\ell - q_1))\frac{T}{2}$$
$$\overset{Q(sh)\ge 2q_1}{>} t_1(m_\ell - q_1) + (\frac{3T}{2} - t_1)(m_\ell - q_1) + (3q_1 - m_\ell)\frac{T}{2} \ge m_\ell T,$$

which is a contradiction, since we have $\forall X \subset select : A(wide) + A(select \backslash X) < m_\ell T$.
Now let $p = 4$. Without loss of generality we assume that there are jobs $J_1, J_2 \in select \backslash sh$ with $p_1 \ge p_2$. Notice that since the algorithm selected 4 jobs of area strictly larger than $\frac{m_\ell T}{4}$ in phase 2 we have $A(wide) < \frac{m_\ell T}{4}$ and thus $P(wide) \le \frac{T}{2}$. Thus we have empty space of widths one between level $\frac{T}{2}$ and $\frac{3T}{2}$ where we can directly schedule J_1 and J_2 at time $\frac{T}{2}$. □

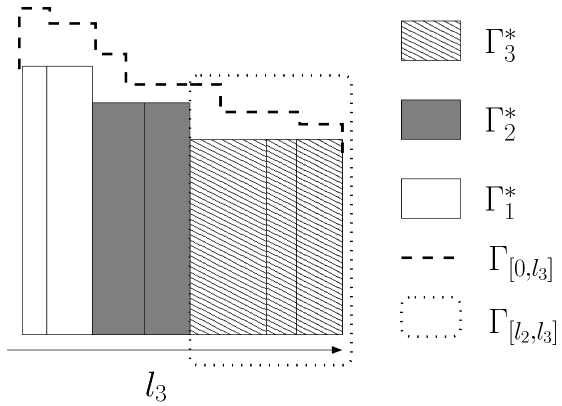

Figure 4: Example of stacks used in Lemma 3

Now we prove that the area of "wide" jobs we scheduled by the algorithm is larger than the one in the optimal. Recall that for all ℓ we denote by π_ℓ the set of jobs scheduled in Cl_ℓ by the algorithm, and π^*_ℓ the set of jobs scheduled in Cl_ℓ in a fixed optimal solution.

Lemma 3. *For $\ell \in \{1, \ldots, N\}$ let $\Pi = \bigcup_{t=1}^{\ell} \pi_t$ be the set of jobs scheduled by the algorithm after finishing the ℓth iteration and $\Pi^* = \bigcup_{t=1}^{\ell} \pi^*_t$ the corresponding optimal set of jobs. Then $A(\Pi \cap Wd^\ell) \geq A(\Pi^* \cap Wd^\ell)$.*

Proof Roughly speaking, this area domination for wide jobs is true since for each cluster Cl_ℓ, we schedule (without counting the guessed jobs that are common to our schedule and the optimal) a length of at least $T - p_{j^*_\ell}$ of the widest possible jobs, and the available length for schedule wide jobs in the optimal is at most $T - p_{j^*_\ell}$. We now start the formal proof.

Assume that $Wd_\ell \neq \emptyset$ after iteration ℓ, otherwise the claim follows directly. We first consider jobs of $B^\ell = Wd^\ell \cap Lg$. We have $\Pi \cap B^\ell = \Pi^* \cap B^\ell$. Indeed, the only jobs of B^ℓ that we scheduled are the guessed one, as no job of B^ℓ can be scheduled in phase one, two or three in clusters $Cl_1 \ldots Cl_\ell$. In addition, the only jobs of B^ℓ scheduled in $\pi^*_1 \ldots \pi^*_\ell$ are also the guessed one, as it is not possible to schedule more than one job of B^ℓ in any cluster $Cl_1 \ldots Cl_\ell$. Thus we only consider now jobs of $Wd^\ell \setminus Lg$. Let $\Gamma = (\Pi \cap Wd^\ell) \setminus Lg$ and $\Gamma^* = (\Pi^* \cap Wd^\ell) \setminus Lg$. W.l.o.g., we assume that there are $a \leq |\Gamma^*|$ different widths $q_1 \geq \ldots \geq q_a$ in Γ^*. Let $\Gamma^*_j \subset \Gamma^*$ be the subset of jobs of width q_j and $n_j := |\Gamma^*_j|$. Let $l_j := \sum_{x=1}^{j} P(\Gamma^*_x)$. We consider the shapes of the rows built by stacking the jobs in Γ and Γ^*, respectively, next to each other sorted by non-increasing width (see Figure 4). We introduce a partial order over stacks of rectangles denoted with "$\leq$". Given two stacks Γ_1 and Γ_2, we say $\Gamma_1 \leq \Gamma_2$ if the shape representing Γ_1 is contained in the one representing Γ_2. For levels l, l' let $\Gamma_{[l,l']}$ denote the row of Γ between l and l'. Remark that $\Gamma^*_{[l_{j-1},l_j]}$ corresponds exactly to Γ^*_j. We show by induction over the number of different widths in Γ^* that $\Gamma^* \leq \Gamma$. Suppose that $\Gamma^*_{[0,l_{j-1}]} \leq \Gamma_{[0,l_{j-1}]}$ and let us prove that $\Gamma^*_{[0,l_j]} \leq \Gamma_{[0,l_j]}$. If all the jobs of $\Gamma^*_{[l_{j-1},l_j]}$ are scheduled by the algorithm we get the desired result. Indeed, no job of Γ^*_j is contained in $\Gamma_{[0,l_{j-1}]}$ otherwise there would be a job J_x and a level $0 \leq l' \leq l_{j-1}$

with $\Gamma^*_{[l',l'+p_x]} > \Gamma_{[0,l'+p_x]}$, as all jobs of $\Gamma^*_{[0,l_{j-1}]}$ are strictly wider than q_j. Thus, $\Gamma^*_{[l_{j-1},l_j]}$ is included in $\Gamma_{[l_{j-1},l_j]}$ and we conclude using the induction hypothesis.
Assume now that there is a job $J_{x_0} \in \Gamma^*_j \backslash \Gamma$. The total processing time of jobs that are wider than J_{x_0} scheduled in the optimal (into clusters $\{Cl_1, \dots, Cl_\ell\}$) is l_j. Let l' denote the total processing time of jobs wider than J_{x_0} that the algorithm packed. We prove that $l' \geq l_j$.
Let N' be the number of clusters where J_{x_0} fits (J_{x_0} fits in clusters $Cl_{\ell-N'+1} \dots, Cl_\ell$). If J_{x_0} is not scheduled in any of these N' clusters, it means that the algorithm scheduled other jobs, that are wider than J_{x_0}. Thus, for any $t \in \{\ell - N' + 1, \dots, \ell\}$, the total processing time of jobs wider than J_{x_0} scheduled by the algorithm on Cl_t is larger than $T - p_{j^*_t}$, which is also an upper bound for the total processing time of schedulable jobs of width larger than q_{x_0} in the optimum. Consequently, we get $l' \geq \Sigma^{\ell}_{t=i-N'+1}(T - p_{j^*_t}) \geq l_j$, which implies $\Gamma[l_{j-1}, l_j] \geq \Gamma^*_{[l_{j-1},l_j]}$. □

We can now prove that all the jobs are scheduled in the end.

Lemma 5. *All the jobs are scheduled when the algorithm stops.*

Proof We prove by induction on ℓ that for all $i \in \{1, \dots, N\}$ we have $A(\bigcup_{t=1}^{\ell} \pi_t) \geq A(\bigcup_{t=1}^{\ell} \pi^*_t)$ after finishing scheduling Cl_ℓ. Let $\Pi = \bigcup_{t=1}^{\ell} \pi_t$ and $\Pi^* = \bigcup_{t=1}^{\ell} \pi^*_t$. Two cases are possible according to what happens in phase 3. If $A(wide) + A(select) \geq m_\ell T$, then we conclude directly. Let us assume that $A(wide) + A(select) < m_\ell T$. This implies that $Fit^\ell \backslash Wd^\ell = \emptyset$ when scheduling Cl_ℓ. Thus, if we write $A(\Pi) = A(\Pi \bigcap Wd^\ell) + A(\Pi \bigcap (I \setminus Wd^\ell))$ we get the desired result as $A(\Pi \bigcap Wd^\ell) \geq A(\Pi^* \bigcap Wd^\ell)$ (according to lemma 3) and $\Pi^* \bigcap (I \setminus Wd^\ell) \subset Fit^\ell \setminus Wd^\ell = \Pi \bigcap (I \setminus Wd^\ell)$. □

2.3 Complexity

Algorithm 1 needs an oracle that provides for every cluster the index of the big (meaning wide and long) job scheduled on this cluster (if such a job is scheduled in the optimum). Thus, the cost of the enumeration is in $O(\prod_{\ell=1}^{N} x_\ell)$, where $x_\ell = |Wd^\ell \cap Lg| + 1$ (we need to add one to encode the possibility where no job of $Wd^\ell \cap Lg$ is scheduled on Cl_ℓ). The problem is that the rough upper bound on this cost (n^N) is almost tight for instances where there are n jobs of width and processing time 1, $N - 1$ clusters of size $2 - \epsilon$ and 1 very large cluster (let us say of size n). In this case there are indeed n possible big jobs for the first $N - 1$ clusters. Another possible bound can be obtained using the fact that $\sum_{\ell=1}^{N} x_\ell \frac{m_\ell}{2} < Q(Lg) \leq \sum_{\ell=1}^{N} m_\ell$, implying $\sum_{\ell=1}^{N} x_\ell < \frac{2\sum_{\ell=1}^{N} m_\ell}{m_1} = \lambda$. Thus, $\prod_{\ell=1}^{N} x_\ell$ is maximized when all the x_ℓ are equal to $\frac{\lambda}{N}$, leading to an overall complexity for the algorithm in $O(N \frac{n \log^2 n}{\log(\log(n))} \log(np_{max}) \min(n, \frac{2\sum_{\ell=1}^{N} m_\ell}{Nm_1})^N)$ (the $\frac{n \log^2 n}{\log(\log(n))}$ factor is the complexity of Steinberg's algorithm, and the $\log(np_{max})$ factor is the running-time of the dichotomic search). Thus, even if this algorithm could be reasonable for "ordinary" instances (where $\prod_{\ell=1}^{N} x_\ell$ is not too large), its worst case complexity remains exponential in N.

We propose in the appendix an improvement of Algorithm 1 using a classical input rounding to replace the $\min\{n, \frac{2\sum_{\ell=1}^{N} m_\ell}{Nm_1}\}$ factor by a constant.

2.4 Improvement using rounding

The idea is that we could still guarantuee the "area domination for wide jobs" (see Lemma 3) by only guessing the processing time of of the (potential) big job scheduled on each cluster, and schedule the widest job that has this processing time. Thus, this new guess could become smaller if the number of different processing times of long jobs is small. We will prove the following theorem.

Theroem 6. *There is a $\frac{5}{2}(1+\epsilon)$-approximation for MCSP where clusters have the same speed that runs in $O(N\frac{n\log^2 n}{\log(\log(n))}\log(np_{max})(\frac{1}{2\epsilon}+1)^N)$.*

Let us first define the rounding.

Lemma 7. *Let I be the original instance, T a guess of $Opt(I)$ and $\epsilon > 0$. We can construct I'_T such that*

- *there are at most $\frac{1}{2\epsilon}+1$ different processing times for all the jobs of Lg' (where $Lg' = \{J_j | p_j > T/2\} \cap I'_T$)*
- *if $T \geq OPT(I)$ then $Opt(I'_T) \leq T(1+\epsilon)$*

Proof We generate I'_T by rounding up the processing time p_j of every long job $J_j \in Lg$ to a value $p'_j := \frac{T}{2} + (a_j+1)\epsilon T$ with $\frac{T}{2} + a_j\epsilon T \leq p_j \leq p'_j$. Of course there are at most $\frac{1}{2\epsilon}+1$ different processing times in Lg'. Since the jobs in Lg are executed in parallel in the optimal solution (since $\frac{T}{2} \geq \frac{OPT(I)}{2}$), replacing those jobs by the ones in Lg' increases the makespan by at most ϵT. Thus $Opt(I'_T) \leq T(1+\epsilon)$. □

Let $T' = T(1+\epsilon)$. Let us now describe the Algorithm 2, that given an instance I'_T (as defined in Lemma 7) either schedules all the jobs with a makespan lower than $\frac{5}{2}T'$, or rejects T' implying that $T' < OPT(I'_T)$ (and thus $T < OPT(I)$). We consider of course that $Wd^\ell = \{J_j \in I'_T | q_j > \frac{m_\ell}{2}\}$.

Algorithm 2

---

for all $\ell \in \{1,\dots,N\}$, guess $p_{j\ell^*}$ the processing time of the (potential) job of $Lg' \cap Wd^\ell \cap \pi^*_l$
for $\ell = 1$ to N **do**
  $J_{x_\ell} \leftarrow$ widest (that have the biggest q_j) job of of processing time $p_{j^*_\ell}$
  if $q_{x_\ell} > \frac{m_\ell}{2}$ **then**
    schedule J_{x_ℓ} on Cl_ℓ //otherwise we say that J_{x_ℓ} is **discarded** by Cl_ℓ
  end if
end for
run Algorithm 1 replacing T by T' (and replacing of course $J_{j^*_l}$ by J_{x_ℓ})

We only have to prove the equivalent of Lemma 3.

Lemma 8. *Let $\ell \in \{1,\dots,N\}$, let $\Pi = \bigcup_{t=1}^{\ell}\pi_t$ after finishing scheduling Cl_ℓ, and let $\Pi^* = \bigcup_{t=1}^{\ell}\pi^*_t$. Then we have $A(\Pi \cap Wd^\ell) \geq A(\Pi^* \cap Wd^\ell)$.*

Proof Let $\ell \in \{1, \ldots, N\}$ and $Wd^\ell \neq \emptyset$ after iteration ℓ of the algorithm. Otherwise the claim follows directly.

We first consider jobs of $B^\ell = Wd^\ell \cap Lg'$ and $B_x^\ell = B^\ell \cap \{J_j | p_j = \frac{T}{2} + x\epsilon T\}$. Let $\Gamma_x = \Pi \cap B_x^\ell$ and $\Gamma_x^* = \Pi^* \cap B_x^\ell$. We will prove that $A(\Pi \cap B^\ell) \geq A(\Pi^* \cap B^\ell)$ by proving that for every x, $A(\Gamma_x) \geq A(\Gamma_x^*)$. Let x be fixed. We proceed as in Lemma 3 by stacking the jobs of Γ_x^* and Γ_x. Let us assume that there are $a \leq |\Gamma_x^*|$ different widths $q_1 \geq \ldots \geq q_a$ in Γ_x^*. Let $\Gamma_{x,j}^* \subset \Gamma_x^*$ be the subset of jobs of width q_j and $n_j := |\Gamma_{x,j}^*|$. Let $l_j := \sum_{t=1}^{j} P(\Gamma_{x,t}^*)$. For levels l, l' let $\Gamma_{[l,l']}$ denote the jobs scheduled in the stack of Γ between l and l'. We show by induction over the number of different widths in Γ_x^* that $\Gamma_x^* \leq \Gamma_x$, where $\leq$ denotes the same partial order as in Section 3.

Suppose that $\Gamma_{x\ [0,l_{j-1}]}^* \leq \Gamma_{x\ [0,l_{j-1}]}$ and let us prove that $\Gamma_{x\ [0,l_j]}^* \leq \Gamma_{x[0,l_j]}$. If all the jobs of $\Gamma_{x,j}^*$ are scheduled by the algorithm we get the desired result.

Assume now that there is a job $J_{x_0} \in \Gamma_{x,j}^* \backslash \Gamma$ (we have $q_{x_0} = q_j$). Let $X_{x_0} = \{J_s | q_s \geq q_{x_0}\}$. Due to the induction hypothesis, we only need to prove that $\Gamma_{x\ [l_{j-1},l_j]} \geq \Gamma_{x\ [l_{j-1},l_j]}^*$, and thus we will only prove that $|\Gamma_x \cap X_{x_0}| \geq |\Gamma_x^* \cap X_{x_0}|$.

We have $|\Gamma_x^* \cap X_{x_0}| = \sum_{t=1}^{j} n_t$, implying that there are at least $\sum_{t=1}^{j} n_t$ clusters where J_{x_0} fits. Moreover, $q_{x_0} > \frac{m_\ell}{2}$ implies that $q_{x_0} > \frac{m_{\ell'}}{2}$ for all $\ell' < \ell$. Thus, J_{x_0} has not been discarded by any cluster between 1 and ℓ. Thus, J_{x_0} has not been scheduled in any of the $\sum_{t=1}^{j} n_t$ clusters where it fits because the algorithm scheduled wider job instead, which proves that $|\Gamma_x \cap X_{x_0}| \geq \sum_{t=1}^{j} n_t = |\Gamma_x^* \cap X_{x_0}|$.

The proof of $A(\Pi \cap Wd^\ell \setminus Lg') \geq A(\Pi^* \cap Wd^\ell \setminus Lg')$ is exactly the same as in Lemma 3 as for any ℓ', the processing time of the jobs $J_{x_{\ell'}}$ is either the same as the potential big job scheduled by the optimal in $Cl_{i'}$, or zero if $J_{x_{\ell'}}$ is discarded. $\square$

Thus, Algorithm 2 is a $\frac{5}{2}(1+\epsilon)$-approximation, and runs in $O(N \frac{n \log^2 n}{\log(\log(n))} \log(np_{max})(\frac{1}{2\epsilon}+1)^N)$.

2.5 A 5/2-approximation for clusters of same size but different speed

It is possible to adapt Algorithm 1 to get a 5/2-approximation for the case where each cluster Cl_ℓ has m identical processors of speed s_ℓ. Again this result also applies for the contiguous case.

We also proceed by dual approximation, and denote by T the current guess. Moreover, let $Fit^\ell = \{J_j | \frac{p_j}{s_\ell} \leq T\}$ be the set of jobs that fit in Cl_ℓ, $Lg^\ell = \{J_j | T \geq \frac{p_j}{s_\ell} > \frac{T}{2}\}$ and $Wd = \{J_j | q_j > \frac{m}{2}\}$.

The idea is to use exactly Algorithm 1 replacing of course Lg by Lg^ℓ and Wd^ℓ by Wd, and scheduling the clusters from the slowest (Cl_1) to the fastest one (Cl_N). Consequently the guess for each cluster ℓ is now the potential job in $Lg^\ell \cap Wd \cap \pi_l^*$ where π_l^* is the set of jobs scheduled on Cl_ℓ in the optimal solution.

The proof of the feasibility of phase 3 is exactly the same as in Lemma 3. The only adaptation needed is to prove the following lemma.

Lemma 9. *For any $x \in \{1, \ldots, N\}$, let $\Pi_x = \bigcup_{t=1}^{x} \pi_t$ be the set of jobs scheduled by the algorithm after finishing the xth iteration and $\Pi_x^* = \cup_{t=1}^{x} \pi_t^*$ the corresponding optimal set of jobs.*

Then we have, for any $\ell \in \{1, \dots, N\}$, $(\Pi_\ell \cap Wd) \geq (\Pi^*_\ell \cap Wd)$, where $\geq$ denotes the same partial order for rows of jobs as in Section 3 for stacks of rectangles.

Proof Let us prove the desired result by induction on ℓ. Let us suppose that $(\Pi_{\ell-1} \cap Wd) \geq (\Pi^*_{\ell-1} \cap Wd)$ (the proof for $\ell = 1$ can be done using the same ideas).

Let $\Gamma = \Pi_l \cap Wd$ and $\Gamma^* = \Pi^*_l \cap Wd$. As in Lemma 1 we prove that $\Gamma^* \leq \Gamma$ by induction on the different numbers of widths of jobs in Γ^*. We use the same notations as in Lemma 3. We suppose that $\Gamma^*_{[0,l_{j-1}]} \leq \Gamma_{[0,l_{j-1}]}$ and we prove that $\Gamma^*_{[0,l_j]} \leq \Gamma_{[0,l_j]}$ by showing that $\Gamma^*_{[l_{j-1},l_j]} \leq \Gamma_{[l_{j-1},l_j]}$. Let us only consider the case where there is J_{x_0} in $\Gamma^*_j \backslash \Gamma$. This implies $J_{x_0} \notin Lg^\ell$, otherwise J_{x_0} would belong to Lg^t for $1 \leq t \leq \ell$, and thus would be a guessed job as the optimal scheduled it in one of the first ℓ clusters. Let $X_\alpha = \{J_s \in J | q_s \geq \alpha\}$ be the set of jobs wider than α. As in Lemma 3 we prove that $\Gamma^*_{[l_{j-1},l_j]} \leq \Gamma_{[l_{j-1},l_j]}$ by showing that $l' \geq l_j$, where $l' = P(X_{q_{x_0}} \cap \Gamma)$ and l_j defined as in Lemma 3 (recall that the definition of l_j implies that $l_j = P(X_{q_{x_0}} \cap \Gamma^*)$).

The hypothesis $(\Pi_{\ell-1} \cap Wd) \geq (\Pi^*_{\ell-1} \cap Wd)$ implies that for any α, $P(X_\alpha \cap \Gamma \cap \Pi_{l-1}) \geq P(X_\alpha \cap \Gamma^* \cap \Pi^*_{l-1})$, thus we use it with $\alpha = q_{x_0}$. Moreover, as J_{x_0} is not scheduled by the algorithm on Cl_ℓ whereas $J_{x_0} \notin Lg^\ell$, it implies that we scheduled wider jobs than J_{x_0} on Cl_ℓ. Thus, we get also $P(X_{q_{x_0}} \cap \Gamma \cap \pi_l) \geq P(X_{q_{x_0}} \cap \Gamma^* \cap \pi^*_l)$, leading to $l' \geq l_j$ and to $\Gamma^*_{[l_{j-1},l_j]} \leq \Gamma_{[l_{j-1},l_j]}$. □

References

[Bougeret et al., 2010a] Bougeret, M., Dutot, P.-F., Jansen, K., Otte, C., and Trystram, D. (2010a). A fast 5/2-approximation for hierarchical scheduling. In *Proceedings of the 16th International European Conference on Parallel and Distributed Computing (EUROPAR)*.

[Bougeret et al., 2010b] Bougeret, M., Dutot, P.-F., Jansen, K., Otte, C., and Trystram, D. (2010b). Approximating the non-contiguous multiple organization packing problem. In *Proceedings of the 6th IFIP International Conference on Theoretical Computer Science (TCS)*.

[Coffman Jr et al., 1980] Coffman Jr, E., Garey, M., Johnson, D., and Tarjan, R. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808.

[Harren et al., 2010] Harren, R., Jansen, K., Prädel, L., and Van Stee, R. (2010). A $5/3 + \epsilon$ approximation for strip packing. submitted.

[Hochbaum and Shmoys, 1988] Hochbaum, D. and Shmoys, D. (1988). A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551.

[Horowitz and Sahni, 1976] Horowitz, E. and Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM (JACM)*, 23(2):317–327.

[Jansen and Solis-Oba, 2007] Jansen, K. and Solis-Oba, R. (2007). New approximability results for 2-dimensional packing problems. *Lecture Notes in Computer Science*, 4708:103.

[Jansen and Thöle, 2008] Jansen, K. and Thöle, R. (2008). Approximation algorithms for scheduling parallel jobs: Breaking the approximation ratio of 2. In *International Colloquium on Automata, Languages and Programming*, pages 234–245.

[Kenyon and Rémila, 2000] Kenyon, C. and Rémila, E. (2000). A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, pages 645–656.

[Leung and Li, 2008] Leung, J. and Li, C. (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2):251–262.

[Li and Wang, 2010] Li, C. and Wang, X. (2010). Scheduling parallel machines with inclusive processing set restrictions and job release times. *European Journal of Operational Research (EJOR)*, 200(3):702–710.

[Schiermeyer, 1994] Schiermeyer, I. (1994). Reverse-fit: A 2-optimal algorithm for packing rectangles. *Lecture Notes in Computer Science*, pages 290–290.

[Schwiegelshohn et al., 2008] Schwiegelshohn, U., Tchernykh, A., and Yahyapour, R. (2008). Online scheduling in grids. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–10.

[Steinberg, 1997] Steinberg, A. (1997). A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26:401.

[Ye et al., 2009] Ye, D., Han, X., and Zhang, G. (2009). On-Line Multiple-Strip Packing. In *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications (COCOA)*, page 165. Springer.

[Zhuk, 2006] Zhuk, S. (2006). Approximate algorithms to pack rectangles into several strips. *Discrete Mathematics and Applications*, 16(1):73–85.