

## Approximating the Non-contiguous Multiple Organization Packing Problem

Marin Bougeret, Pierre François Dutot, Klaus Jansen, Christina Otte, Denis  
Trystram

► **To cite this version:**

Marin Bougeret, Pierre François Dutot, Klaus Jansen, Christina Otte, Denis Trystram. Approximating the Non-contiguous Multiple Organization Packing Problem. 6th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science (TCS) / Held as Part of World Computer Congress (WCC), Sep 2010, Brisbane, Australia. pp.316-327, 10.1007/978-3-642-15240-5\_23 . hal-01054450

**HAL Id: hal-01054450**

**<https://hal.inria.fr/hal-01054450>**

Submitted on 6 Aug 2014

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

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



# Approximating the non-contiguous Multiple Organization Packing Problem

Marin Bougeret<sup>\*1</sup>, Pierre François Dutot<sup>1</sup>, Klaus Jansen<sup>2</sup>, Christina Otte<sup>2</sup>,  
and Denis Trystram<sup>1</sup>

<sup>1</sup> Grenoble University

ZIRST 51, avenue Jean Kuntzmann  
38330 Montbonnot Saint Martin, France  
{bougeret,pfdutot,trystram@imag.fr}

<sup>2</sup> Department of Computer Science

Christian-Albrechts-University to Kiel  
Christian-Albrechts-Platz 4, 24098 Kiel, Germany.  
{kj,cot}@informatik.uni-kiel.de

**Abstract.** We present in this paper a  $5/2$ -approximation algorithm for scheduling rigid jobs on multi-organizations. For a given set of  $n$  jobs, the goal is to construct a schedule for  $N$  organizations (composed each of  $m$  identical processors) minimizing the maximum completion time (makespan). This algorithm runs in  $O(n(N + \log(n)) \log(np_{max}))$ , where  $p_{max}$  is the maximum processing time of the jobs. It improves the best existing low cost approximation algorithms. Moreover, the proposed analysis can be extended to a more generic approach which suggests different job partitions that could lead to low cost approximation algorithms of ratio better than  $5/2$ .

## 1 Problem statement

In this paper we consider the problem of scheduling rigid jobs on Multi-organizations. An organization is a set of  $m$  identical available processors. A job  $j$  must be executed on  $q_j$  processors (sometimes called the *degree of parallelism*) during  $p_j$  units of time. The  $q_j$  processors must be allocated on the same organization. The makespan of the schedule is defined as the maximum finishing time over all the jobs. Given a set of  $n$  jobs, the goal is to find a non-overlapping schedule of all the jobs on  $N$  organizations while minimizing the makespan.

This problem is closely related to strip packing problems. Indeed, if we add the constraint of using contiguous processors, then scheduling a job  $j$  on  $q_j$  contiguous processors during  $p_j$  units of time is equivalent to packing a rectangle of width  $q_j$  and height  $p_j$ .

---

\* This work has been supported by DGA-CNRS

*Related works.* Strip packing, rigid jobs scheduling and Multi-organizations scheduling problems are all strongly  $\mathcal{NP}$ -hard, and Zhuk [1] showed that there is no polynomial time approximation algorithm with absolute ratio better than 2 for strip packing.

For Strip Packing problem, Coffman et al. gave in [2] an overview about performance bounds for shelf-oriented algorithms as *NFDH* (Next Fit Decreasing Height) and *FFDH* (First Fit Decreasing Height). These algorithms have a approximation ratio of 3 and 2.7, respectively. Schiermeyer [3] and Steinberg [4] presented independently an algorithm for Strip Packing with absolute ratio 2. A further important result for the Strip Packing problem is an AFPTAS with additive constant  $\mathcal{O}(1/\epsilon^2)$  of Kenyon and Rémila [5]. This constant was improved by Jansen and Solis-Oba, who presented in [6] an APTAS with additive constant 1. Concerning the multi-strip packing problem, there is a  $2 + \epsilon$  approximation in [7] whose algorithmic cost is doubly exponential in  $\frac{1}{\epsilon}$ . In [8] we gave a 2 approximation with a large algorithmic cost and an AFPPTAS for this problem.

Let us now review the related work about rigid job scheduling. For one organization, the famous List Algorithm for scheduling with resource constraints of Garey and Graham [9] can be applied (when there is only one resource to share) to schedule rigid jobs, and is then a 2 approximation. The rigid job scheduling problem on multi-organization has been studied with an on-line setting in [10]. The authors achieved a ratio of 3 without release times (and 5 with release times). Notice that these results do not require the knowledge of the processing times of the jobs. Moreover, the organizations may have a different number of processors. The rigid job scheduling problem on multi-organizations has been extended in [11] for the case where the jobs are submitted to local queues on each cluster with the extra constraint that the initial local schedules must not be worsened. The authors provide a 3-approximation.

Generally, the results about rigid job scheduling cannot be adapted to the more constrained contiguous version. To the best of our best knowledge, there is still no (reasonable)  $\alpha$  such that for any instance  $I$ ,  $Optc(I) \leq \alpha Optnc(I)$  (where  $Optc$  denotes the contiguous optimal value and  $Optnc$  the non-contiguous one). The authors of [12] show that  $\alpha > 1$  by constructing a (rather) simple instance with 8 jobs and 4 machines.

*Our contribution.* In this paper, we present a  $\frac{5}{2}$  approximation algorithm for the rigid job scheduling problem on multi-organizations that runs in  $O(n(N + \log(n)) \log(np_{max}))$ , where  $p_{max}$  is the maximum processing time of the jobs. Moreover, we suggest how the approach used for the  $\frac{5}{2}$ -algorithm could be extended to get approximation algorithms with better ratio and a low algorithmic cost.

*Organization of the Paper.* The preliminaries for the  $\frac{5}{2}$ -approximation are in Section 2. In Section 3.1 to 3.4 we describe how to construct a preallocation of the “big” jobs that fits in the targeted makespan. In Section 4 we show how to turn this preallocation into a compact schedule, and in Section 5 we analyze the complexity of the algorithm. The discussions on the approach are in Section 6.

## 2 Principle and definitions

Let us now give some definitions that are used throughout the proofs and the description of the algorithm. We first extend the previous  $p_j$  and  $q_j$  notations to  $Q(X)$  and  $P(X)$  where  $X$  is a set of jobs. We also define the surface (sometimes also called the area) of a set of jobs as  $S(X) = \sum_{j \in X} q_j p_j$ . A *layer* is a set of jobs which are scheduled sequentially on the same organization. The length of a layer  $Lay$  is  $P(Lay)$ , the sum of the processing time of all the jobs in  $Lay$ . A *shelf* is a set of jobs which are scheduled on the same organization, and which start at the same time. Given a shelf  $sh$ , the value  $Q(sh)$  is called the *height* of  $sh$ . What we call a *bin* can be seen as a reservation of a certain number of processors (generally  $m$ ) during a certain amount of time. The algorithm will add some jobs to bins, and given a bin  $b$ , we denote by  $Q(b)$  the value  $\sum_{\{j \in b\}} q_j$ . Given a sequence of bins  $seq$ , we denote by  $Q(seq)$  the value  $\sum_{b \in seq} Q(b)$ . These notations are extended in the same way for  $P$  and  $S$ . In the whole paper, we consider that the sets of jobs used as parameters in the algorithms are modified after the calls.

Let us sketch how  $5/2$  algorithm is constructed. Let  $OPT$  denote the value of an optimal solution. We target a  $\frac{5}{2}$  ratio by both ensuring that, for each organization at least half of the processors are used at any time before the starting time of the last job, and that the small jobs (whose processing time is lower than  $OPT/2$  and height lower than  $m/2$ ) are scheduled at the end. Thus, if the makespan of the final schedule is due to a small job, it is lower than the processing time of the small job plus the starting time of this job, implying a makespan lower than  $OPT/2 + 2OPT = 5OPT/2$ . As the optimal value is not known, we use the well known dual approximation technique [13]. Let  $w$  denote the current guess of  $OPT$ . The schedule is built in three steps. In the first one we compute a preallocation  $\pi_0$  of the “big” ( $p_j > w/2$  or  $q_j > m/2$ ) jobs. Then we apply a list algorithm which turns  $\pi_0$  into a “compact” schedule  $\pi_1$  (see Section 4). Finally, the final schedule  $\pi$  is constructed by adding to  $\pi_1$  the small remaining jobs using again a list algorithm (see also Section 4).

Let us define the following sets:

- let  $L_H = \{j | q_j > m/2\}$  be the set of high jobs
- let  $L_{XL} = \{j | p_j > 3w/4\}$  be the set of extra long jobs
- let  $L_L = \{j | 3w/4 \geq p_j > w/2\}$  be the set of long jobs
- let  $L_B = (L_{XL} \cup L_L) \cap L_H$  be the set of huge jobs
- let  $I' = L_H \cup L_{XL} \cup L_L$

We will prove that either we schedule  $I$  with a resulting makespan lower than  $5w/2$ , or  $w < OPT$ . Notice that for the sake of simplicity we did not add the “reject” instructions in the algorithm. Thus we consider in all the proof that  $w \geq OPT$ , and it is implicit that if one of the claimed properties is wrong during the execution, the considered  $w$  should be rejected. Notice that we only consider the  $w$  values such that  $Q(L_{XL} \cup L_L) \leq Nm$  and  $P(L_H) \leq Nw$ .

We start by providing in Section 3 the three phase algorithm *Build\_Prealloc* that builds the preallocation  $\pi_0$  of the jobs of  $I'$ . We will denote by  $\pi_0^i$  the set of

preallocated jobs in organization  $O_i$ . In phase 1 we preallocate the high jobs. In phase 2 and phase 3 we preallocate the long and extra long jobs by first packing shelves of jobs into bins, and then putting these bins into organizations. An example of a preallocation is depicted Figure 1.

### 3 Construction of the preallocation

#### 3.1 Phase 1

Let  $N_1$  be the number of organizations used in phase 1. In phase 1, the jobs of  $L_H$  are packed in  $N_1$  organizations. The  $Create\_Layer(X, l)$  procedure creates a layer  $Lay$  of length at most  $l$ , using a Best Fit (according to the processing times) policy (BFP). Thus,  $Create\_Layer(X, l)$  add at each step the longest job that fits. Thus, phase 1 calls for each organization (until  $L_H$  is empty)  $Create\_Layer(L_H, 5w/2)$ .

Let us introduce some notations. Let  $Lay_i$  denote the set of jobs scheduled in the layer created in organization  $O_i$ . Let  $L_{XL}^1$  and  $L_L^1$  denote the remaining jobs of  $L_{XL}$  and  $L_L$  after phase 1. Thus, for the moment we have  $\pi_0^i = Lay_i$  for all  $i \leq N_1$ .

**Lemma 1 (phase 1).** *If  $\exists i_0 < N_1$  such that  $P(\pi_0^{i_0}) \leq 2w$  then it is straightforward to pack all the jobs of  $I'$ . Otherwise, we get  $\forall i \in \{1, \dots, N_1 - 1\}$ ,  $S(\pi_0^i) > wm$  and  $N_1 \leq \lceil N/2 \rceil$ .*

*Proof.* First let us notice that phase 1 ends, as  $P(L_H) \leq Nw$  and  $P(\pi_0^i) > w$  for every organization where we do not run out of jobs to schedule. We first suppose that  $\exists i_0 < N_1$  such that  $P(\pi_0^{i_0}) \leq 2w$ . In this case we just have to prove that it is straightforward to preallocate  $L_{XL} \cup L_L$ . We proceed by contradiction by supposing that we never ran out of jobs of  $L_{XL} \cup L_L$ . When the algorithm creates a layer for a organization  $i$ , we know due to the BFP order that it will pack at least two jobs of  $L_B$ , if  $L_B$  is not empty. The hypothesis implies that during the execution of phase 1,  $L_H \setminus L_B$  was empty before  $L_B$ . Thus, for  $i < N_1$ , there is at least two jobs of  $L_B$  in  $\pi_0^i$ , meaning that  $\forall i$  with  $1 \leq i < N_1$ ,  $Q((L_{XL} \cup L_L) \cap \pi_0^i) > m$ .

Concerning the  $N - N_1$  other organizations, we can create shelves of jobs of  $L_L \cup L_{XL}$  using a best fit according to the height (BFH), implying that each shelf has a height of at least  $2m/3$  according to Lemma 2. Packing two shelves in each organization, we get  $\forall i > N_1$ ,  $Q((L_{XL} \cup L_L) \cap \pi_0^i) > 4m/3 > m$ .

Finally, let us check what is scheduled in organization  $N_1$ . If two jobs of  $L_B$  are scheduled in this organization, then  $Q((L_{XL} \cup L_L) \cap \pi_0^{N_1}) > m$ . If one job of  $L_B$  is scheduled, then we create one shelf of jobs of  $L_{XL} \cup L_L$ , and  $Q((L_{XL} \cup L_L) \cap \pi_0^{N_1}) > m/2 + 2m/3$ . If no huge job is scheduled in organization  $N_1$ , we pack as before two shelves of jobs of  $L_{XL} \cup L_L$ . Thus, if in every case we have  $Q((L_{XL} \cup L_L) \cap \pi_0^{N_1}) > m$ . Thus, we get  $Q((L_{XL} \cup L_L)) > Nm$ , which is impossible.

Let us prove the second part of the lemma. First notice that for any  $i < N_1$ ,  $S(\pi_0^i) > 2w^{m/2} = mw$ . Moreover, we have  $2(N_1 - 1)w < \sum_{i=1}^{N_1} P(\pi_0^i) = P(L_H) \leq Nw$ , implying  $N_1 \leq \lceil N/2 \rceil$ .  $\square$

Thus, we now assume until the end of the proof that we are in the second case of Lemma 1 where  $\forall i \in \{1, \dots, N_1 - 1\}$ ,  $S(\pi_0^i) > mw$  and  $N_1 \leq \lceil N/2 \rceil$ .

### 3.2 Phase 2

In phase 2 the jobs of  $L_{XL}^1 \cup L_L^1$  are scheduled in organization  $N_1$  by creating shelves according to what is already scheduled in organization  $N_1$ . We denote by  $L_{XL}^2$  and  $L_L^2$  the remaining jobs of  $L_{XL}^1$  and  $L_L^1$  after phase 2. Let us first define two procedures used for phase 2 and phase 3.

The procedure *Pack\_Shelf*( $X, b, f$ ) creates a shelf *sh* using the Best Fit (according to the height) policy (BFH), and packs it into bin  $b$ . The  $f$  parameter represents the available height of  $b$  (meaning that  $b$  corresponds to  $f$  free processors during a certain amount of time), implying of course that  $Q(sh) \leq f$ . Thus *Pack\_Shelf*( $X, b, f$ ) adds at each step the highest possible job of  $X$  that fits. We assume that the length of the bin is larger than  $p_j$ , for all  $j \in X$ .

The procedure *GreedyPack*( $X, seq$ ) creates for each empty bin  $b \in seq$  one shelf of jobs of  $X$  using *Pack\_Shelf*( $X, b, m$ ). This procedure returns the last bin in which a shelf has been created. Let us now come back to the description of phase 2.

Depending on the set of jobs already scheduled in  $O_{N_1}$ , the *Create\_Padding*() procedure creates  $n_{bin_L}$  empty bins of length  $3w/4$  and  $n_{bin_{XL}}$  empty bins of length  $w$ , which are added in organization  $O_{N_1}$ . Let us define for each case how many bins of each type are created by *Create\_Padding*():

- If  $P(Lay_{N_1}) \in ]3w/2, 7w/4]$  then set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(1, 0)$
- If  $P(Lay_{N_1}) \in ]w, 3w/2]$  then set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(0, 1)$
- If  $P(Lay_{N_1}) \in ]3w/4, w]$  then
  - if  $Q(L_L^1) \geq 5/4$  then set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(2, 0)$
  - else set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(0, 1)$
- If  $P(Lay_{N_1}) \in ]w/2, 3w/4]$  then set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(1, 1)$
- If  $P(Lay_{N_1}) \in [0, w/2]$  then set  $(n_{bin_L}, n_{bin_{XL}})$  to  $(0, 2)$

Let  $pad_L$  be a sequence of  $n_{bin_L}$  bins of length  $3w/4$  and  $pad_{XL}$  be a sequence of  $n_{bin_{XL}}$  bins of length  $w$ . *Create\_Padding*() returns  $(pad_L, pad_{XL})$ . All in all, phase 2 can be described by the following procedure calls:

- Let  $(pad_L, pad_{XL}) = \text{Create\_Padding}()$
- *GreedyPack*( $L_{XL}^1, pad_{XL}$ )
- *GreedyPack*( $L_L^1, pad_L$ )
- *GreedyPack*( $L_L^1, pad_{XL}$ )

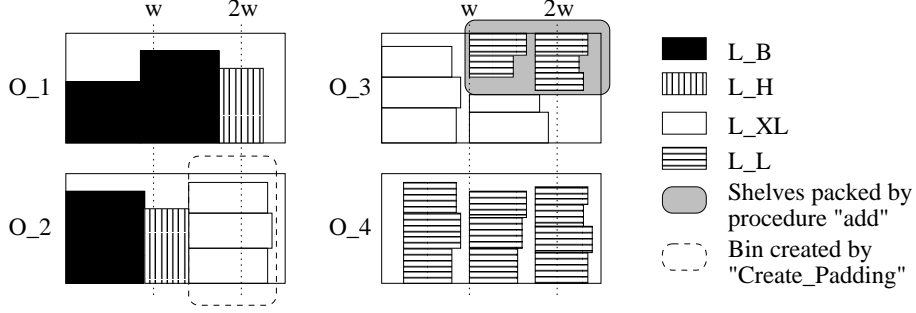


Fig. 1. An example of pre-allocation

### 3.3 Phase 3

In phase 3 we first schedule the jobs of  $L_{XL}^2$  using the  $N_2 = N - N_1$  remaining organizations. Then, we schedule the jobs of  $L_L^2$  using also this  $N_2$  organizations. Finally, the possibly remaining jobs of  $L_L^2$  are added to the last bin used for the extra long jobs. Therefore, let us define the  $add(X, b)$  procedure. The  $add(X, b)$  procedure packs one or two “small” shelves of jobs of  $X$  in the bin  $b$  (starting from the top of the bin for the sake of clarity). Notice that, as  $b$  will be the last bin used for extra long jobs, the available height (for the jobs of  $X$ ) in  $b$  will be generally lower than  $m$ . Here is the description of  $add(X, b)$ :

- If the left side of  $b$  is at time  $w$  then let  $l = 2$  else let  $l = 1$
- Repeat  $l$  times the call  $Pack\_Shelf(X, b, m - Q(b))$  and pack the created shelves in  $b$ .

An example of a call to the  $add$  procedure is given in Figure 1 for the case where  $l = 2$ .

We now define two sequences of bins  $seq_{XL}$  and  $seq_L$ , such that every bin of  $seq_{XL}$  (resp.  $seq_L$ ) will (possibly) contains one shelf of jobs of  $L_{XL}^2$  (resp.  $L_L^2$ ). Notice that a free organization can be seen as two bins of length  $w$  (and height  $m$ ), three bins of length  $3w/4$ , or one bin of length  $w$  and two bins of length  $3w/4$ . Thus,  $seq_{XL}$  is composed of  $2(N - N_1)$  bins  $(b_1, \dots, b_{2(N - N_1)})$  of length  $w$ , considering that we created two bins in each of the organizations  $\{O_{N_1+1}, \dots, O_N\}$ , starting from  $O_{N_1+1}$ . This implies that for all  $i \geq 1$ , bins  $b_{2i-1}$  and  $b_{2i}$  are in  $O_{N_1+i}$ . The sequence  $seq_L$  is composed of  $3(N - N_1)$  bins  $(b'_1, \dots, b'_{3(N - N_1)})$  of length  $3w/4$ , considering that we created three bins in each of the organizations  $\{O_{N_1+1}, \dots, O_N\}$ , from  $O_N$  to  $O_{N_1+1}$ . This implies that for all  $i \geq 1$ , bins  $b'_{3i-2}$ ,  $b'_{3i-1}$  and  $b'_{3i}$  are in  $O_{N-i+1}$ . Notice that these two sequences are not ordered in the same way.

All in all, phase 3 can be described by the following procedure calls:

- Let  $last = GreedyPack(L_{XL}^2, seq_{XL})$
- $GreedyPack(L_L^2, seq_L)$

- $add(L_L^2, last)$

Let start the analysis of phase 3 with a remark about  $Pack\_Shelf(X, b, f)$ .

**Lemma 2.** *Let  $Sh$  denote the shelf created by  $Pack\_Shelf(X, b, f)$ . If we know that the  $k$  highest jobs of  $X$  fit in  $f$ , then  $Q(Sh) > \frac{k}{k+1}f$ .*

*Proof.* Let  $x$  be the cardinal of  $X$ . Let us assume that  $q_i \geq q_{i+1}$  for  $1 \leq i < x$ . Let  $i_0 \geq k+1$  be the first index such that job  $i_0$  is not in  $Sh$ . Let  $a = \sum_{i=1}^{i_0-1} q_i$ . We have  $Q(Sh) \geq a \geq (i_0 - 1)q_{i_0} > (i_0 - 1)(f - a)$  leading to  $a > \frac{i_0-1}{i_0}f \geq \frac{k}{k+1}f$ .  $\square$

**Lemma 3 (phase 3).** *If there remains an unscheduled job after phase 3, then  $S(L_{XL}^2 \cup L_L^2) > (N_2 + 1/8)mw$ .*

*Proof.* Let us first suppose that  $L_{XL}^2 \neq \emptyset$ . Let  $a_{XL} = 2(N - N_1)$  be the number of bins in  $seq_{XL}$ . After having filled the first  $a_{XL} - 1$  bins (using a width of at least  $2/3$  according to Lemma 2), the width of remaining jobs of  $L_{XL}^2$  is strictly larger than  $m$ . Thus we get  $Q(L_{XL}^2) > 2m/3(a_{XL} - 1) + m = 4m/3N_2 + m/3$  and  $S(L_{XL}^2) > (N_2 + 1/4)mw$ .

We now suppose that  $L_{XL}^2 = \emptyset$ . In every organization that contains two bins of jobs of  $L_{XL}^2$ , the total scheduled area is strictly larger than  $2 \times 2m/3 \times 3w/4 = wm$ . In every organization that contains three bins of jobs of  $L_L^2$ , the total scheduled area is strictly larger than  $3 \times 2m/3 \times w/2 = wm$ . We have to consider two cases according to the position of the last bin  $last$  (the left side of  $last$  may be located at time 0 or  $w$ ). Let  $i_0$  be the index of the organization that contains  $last$ .

In the first case where the left side of  $last$  is at time 0, two bins (of length  $3w/4$  and height  $m$ ) were created after the bin  $last$  in organization  $O_{i_0}$ . Then, if the remaining jobs of  $L_L^2$  do not fit in  $last$ , the total area of the jobs scheduled in organization  $O_{i_0}$  is strictly larger than  $(2^{2m/3} + m)w/2 > 7wm/6$ . Then we just sum the area packed over all the organizations, and get the desired result.

In the second case where the left side of  $last$  is at time  $w$  (as depicted in Figure 1), the only room in organization  $O_{i_0}$  to schedule jobs of  $L_L^2$  is in  $last$ . In organization  $O_{i_0}$ , the area of (extra long) jobs contained in the first bin is strictly larger than  $wm/2$ . The  $add$  procedure will create two shelves (one next to the other) of jobs of  $L_L^2$  in  $last$ .

Let  $last'$  and  $L_L'$  be the set of jobs in  $last$  and  $L_L^2$  respectively, just before the call of the  $add$  procedure. If  $Q(last') > m/2$ , and as the remaining jobs of  $L_L^2$  don't fit in  $last$ , we have that  $Q(L_L') > m - Q(last')$ . This implies  $S(last' \cup L_L') > 3w/4Q(last') + w/2(m - Q(last')) > 5wm/8$ . If  $Q(last') \leq m/2$  (see Figure 1) then  $add$  creates a first shelf of jobs of  $L_L^2$  of height at least  $(m - Q(last'))/2$ , and then tries to pack the remaining jobs in the second shelf. Thus in this case,  $S(last' \cup L_L') > 3w/4Q(last') + w/2 \left( \frac{m - Q(last')}{2} + m - Q(last') \right) > 3mw/4$ .  $\square$

### 3.4 Main algorithm

In this section we recall the overall algorithm that builds the preallocation, and we provide the main proof of the preallocation. Notice that we drop the  $L_L^i$  and



$L_{XL}^i$  notations for writing the algorithm as we consider that the sets of jobs (used as parameters in the procedures) are modified after the calls.

**Algorithm 1** *Build\_Prealloc(I')*

Phase 1 [1] Let  $i = 0$   
 [2] Let  $i = i + 1$ , and let  $Lay_i = \text{Create\_Layer}(L_H, 5w/2)$   
 Pack  $Lay_i$  in organization  $S_i$  from time 0  
 [3] Repeat step 3 until  $L_H$  is empty  
 Phase 2 [4] Let  $(pad_L, pad_{XL}) = \text{Create\_Padding}()$   
 [5] Let  $last = \text{GreedyPack}(L_{XL}, pad_{XL})$   
 [6] Call  $\text{GreedyPack}(L_L, pad_L)$   
 [7] Call  $\text{GreedyPack}(L_L, pad_{XL})$   
 Phase 3 [8] Let  $seq_{XL}$  and  $seq_L$  be defined as described in Section 3.3  
 [9] Let  $last_2 = \text{GreedyPack}(L_{XL}, seq_{XL})$   
 [10] If  $last_2$  is not null, set  $last$  to  $last_2$   
 [11] Call  $\text{GreedyPack}(L_L, seq_L)$   
 [12] Call  $\text{add}(L_L, last)$

**Theorem 1.** *Build\_Prealloc(I')* creates a preallocation  $\pi_0$  of makespan lower than  $5w/2$ .

*Proof.* Remind that  $L_{XL}^1$  and  $L_L^1$  denote the remaining jobs of  $L_{XL}$  and  $L_L$  after Phase 1. The makespan of the preallocation is by construction lower than  $5w/2$ . We know that according to Lemma 1 phase 1 terminates and the area scheduled in the first  $N_1 - 1$  organizations is greater than  $(N_1 - 1)wm$ . We proceed by contradiction by supposing that  $L_{XL}^1 \cup L_L^1$  is not empty after Phase 2 and Phase 3, and showing that  $S(I') > Nm w$ . We proceed by case analysis according to what is scheduled in  $O_{N_1}$ .

If  $P(Lay_{N_1}) > \frac{7}{4}w$ , then  $S(Lay_{N_1}) > \frac{7}{8}mw$  and *CreatePadding* doesn't create any bin. If  $L_{XL}^1$  and  $L_L^1$  are not completely scheduled by phase 3, then according to Lemma 3 we get  $S(L_{XL}^1 \cup L_L^1) > (N_2 + \frac{1}{8})mw$ . Thus in this case we have  $S(Lay_{N_1} \cup L_{XL}^1 \cup L_L^1) > (N_2 + 1)mw$ , implying  $S(I') > Nm w$ .

If  $\frac{7}{4}w \geq P(Lay_{N_1}) > \frac{3}{2}w$ , then  $S(Lay_{N_1}) > \frac{3}{4}mw$  and *CreatePadding* creates one bin of length  $\frac{3}{4}w$ . Recall that the jobs of  $L_L^1$  are first scheduled in  $pad_L$ . If  $Q(pad_L)$  is larger than  $\frac{m}{2}$ , then  $S(Lay_{N_1} \cup pad_L) > \frac{3}{4}mw + \frac{1}{4}mw = mw$ . Thus, the total area packed in the first  $N_1$  is strictly larger than  $N_1wm$ . Then, according to Lemma 3,  $L_{XL}^2 \cup L_L^2$  must fit in the  $N_2$  remaining organizations. If  $Q(pad_L) \leq \frac{m}{2}$ , then the  $N_2$  remaining organizations are available for  $L_{XL}^1$ . Thus, if  $L_{XL}^1 \neq \emptyset$  at the end, then  $S(L_{XL}^1) > (N_2 + \frac{1}{4})mw$ , and  $S(Lay_{N_1} \cup L_{XL}^1) > (N_2 + 1)mw$ .

If  $\frac{3}{2}w \geq P(Lay_{N_1}) > w$ , then  $S(Lay_{N_1}) > \frac{1}{2}mw$  and *CreatePadding* creates one bin of length  $w$ . If  $Q(pad_{XL})$  is larger than  $\frac{2m}{3}$  then  $S(Lay_{N_1} \cup pad_{XL}^1) > mw$  and we conclude with Lemma 3. Otherwise, the  $N_2$  remaining organizations are available for  $L_L^1$ . Moreover, remind that in this case the only bin in  $pad_{XL}$  will be used for jobs of  $L_L$  during the call of *add*. Then, if  $L_L^1$  does not fit, we have  $Q(L_L^1 \cup L_{XL}^1) > (2N_2 + 1)m$  and  $S(Lay_{N_1} \cup L_L^1 \cup L_{XL}^1) > \frac{mw}{2} + N_2wm + \frac{wm}{2} = (N_2 + 1)mw$ .

If  $w \geq P(\text{Lay}_{N_1}) > \frac{3}{4}w$ , then  $S(\text{Lay}_{N_1}) > \frac{3}{8}mw$  and two cases are possible according to the value of  $Q(L_L^1)$ . If  $Q(L_L^1) \geq \frac{5m}{4}$ , *CreatePadding* creates two bins of length  $\frac{3}{4}w$ . Then,  $S(\text{Lay}_{N_1} \cup \text{pad}_L) > (\frac{3}{8} + \frac{5}{8})mw$  and we conclude with Lemma 3. Otherwise, if  $Q(L_L^1) < \frac{5m}{4}$ , *CreatePadding* creates one bin  $\text{pad}_{XL}$  of length  $w$ . If  $Q(\text{pad}_{XL})$  (after the call line 5) is larger than  $\frac{2m}{3}$  then  $S(\text{Lay}_{N_1} \cup \text{pad}_{XL}^1) > \frac{7}{8}mw$  and we conclude with Lemma 3. Otherwise, jobs of  $L_{XL}^1$  are all scheduled in  $\text{pad}_{XL}$ . As  $N_2 \geq 1$ , at least three bins are available for  $L_L^1$ , which is sufficient given that  $Q(L_L^1) < \frac{5m}{4}$ .

If  $\frac{3}{4}w \geq P(\text{Lay}_{N_1}) > \frac{1}{2}w$ , then  $S(\text{Lay}_{N_1}) > \frac{1}{4}mw$  and *CreatePadding* creates one bin of length  $w$  and one bin of length  $\frac{3}{4}w$ . If extra long jobs are not scheduled at the end of the algorithm, then  $S(\text{Lay}_{N_1} \cup \text{pad}_{XL}) > \frac{3}{4}mw$ . Since  $L_{XL}^2$  do not fit into  $N_2$  free organizations, we have also  $S(L_{XL}^2) > (N_2 + \frac{1}{4})mw$ . Thus we conclude that the extra long jobs are successfully scheduled. Let us suppose now that the long jobs are not completely scheduled. If  $Q(\text{pad}_{XL}) \geq \frac{5m}{9}$  then  $S(\text{Lay}_{N_1} \cup \text{pad}_{XL} \cup \text{pad}_L) > (\frac{1}{4} + \frac{5}{12} + \frac{1}{3})mw = mw$ . Otherwise, let  $L'_L$  denote the set of remaining jobs of  $L_L^1$  just before the call to *add*. The area scheduled in the  $N_2$  last organizations is larger than the one scheduled in the optimal. If  $L'_L$  does not fit in  $\text{pad}_{XL}$  during the call to *add*, then  $Q(L_{XL}^1 + L'_L) > m$  and  $S(\text{Lay}_{N_1} \cup \text{pad}_L \cup \text{pad}_{XL} \cup L'_L) > (\frac{1}{4} + \frac{1}{3} + \frac{1}{2})mw > mw$ .

If  $\frac{1}{2}w > P(\text{Lay}_{N_1})$ , *CreatePadding* creates two bins of length  $w$ . If  $N_1 > 1$ , then  $S(\bigcup_{i=1}^{N_1} \text{Lay}_i) > S(\bigcup_{i=1}^{N_1-2} \text{Lay}_i) + \frac{5}{4}mw > (N_1 - 1)mw + \frac{1}{4}mw$  because the first job of  $\text{Lay}_{N_1}$  does not fit in the previous organization. Thus, if  $Q(L_{XL}^1) > m$  then we have  $S(\bigcup_{i=1}^{N_1} \text{Lay}_i \cup \text{pad}_{XL}) > N_1mw$  and we conclude with Lemma 3. Otherwise, we have an empty bin in the sequence  $\text{pad}_{XL}$  and  $N_2$  free organizations available for  $L_L^1$ . Let  $L'_L$  be  $L_L^1$  before the call to *add*. If *add* does not schedule  $L'_L$  in *last* then  $Q(L_{XL}^1) + Q(L'_L) > m$  and  $S(\bigcup_{i=1}^{N_1} \text{Lay}_i \cup L_{XL}^1 \cup L'_L) > (N_1 - \frac{3}{4} + \frac{1}{3} + \frac{1}{2})mw > N_1mw$ . If  $N_1 \leq 1$  then we have two bins of length  $w$  in each of the  $N$  organizations, which is of course sufficient to pack  $L_{XL}^1 \cup L_L^1$ .  $\square$

## 4 From the preallocation to the final schedule

From now on, we suppose that the preallocation  $\pi_0$  is built. For each organization  $O_i$ ,  $\pi_0$  indicates first a (possibly empty) sequence of high jobs  $j_1^i, \dots, j_{x_i}^i$  that have to be scheduled sequentially from time 0. Then,  $\pi_0$  contains an ordered sequence of shelves  $Sh_1^i, \dots, Sh_{x_i}^i$ . Moreover, the makespan of  $\pi_0$  is by construction less than  $5w/2$ .

**Definition 1.** Let  $u_i(t)$  be the utilization of organization  $O_i$  at time  $t$ , i.e.  $u_i(t)$  is the sum of all the  $q_j$  for any job  $j$  which scheduled on organization  $i$  at time  $t$ . A schedule is  $1/2$  compact if and only if for every organization  $O_i$  there exists a time  $t_i$  such that for all  $t \leq t_i$ ,  $u_i(t) \geq m/2$  and  $u_i$  restricted to  $t > t_i$  is not increasing.

Let us now describe the algorithm  $LS_{\pi_0}$  which turns  $\pi_0$  into a  $1/2$  compact schedule  $\pi_1$  of  $I'$ . We first define the procedure  $Add\_Asap(X, O_i)$  which scans

organization  $O_i$  from time 0, and for every time  $t$  starts any possible job(s) in  $X$  that fit(s) at time  $t$ . The  $LS_{\pi_0}$  works as follows: for every organization  $O_i$ , pack first sequentially the high jobs  $j_x^i$  for  $1 \leq x \leq x_i$  and then call  $Add\_Asap(Sh_x, O_i)$  for  $1 \leq x \leq x'_i$ .

**Lemma 4.** *The makespan of  $\pi_1$  is lower than the one of  $\pi_0$ , and  $\pi_1$  is  $1/2$  compact.*

*Proof.* Let  $\sigma_i$  be a schedule in a (single) organization  $O_i$  (of makespan  $C_i$ ), let  $X$  be a set of jobs and let  $\sigma'_i$  be the schedule (of makespan  $C'_i$ ) produced by  $Add\_Asap(X, O_i)$ . If  $\sigma_i$  is  $1/2$  compact and if for all  $j \in X, q_j \leq m/2$ , then  $\sigma'_i$  is  $1/2$  compact. The proof is straightforward by induction on the cardinality of  $X$ . Moreover, if  $\sum_X q_j \leq m$ , then  $C'_i \leq C_i + \max_X p_j$  because in the worst case all the jobs of  $X$  only start at time  $C_i$ . Using these two properties, we prove the lemma for every organization  $O_i$  by induction on the number of call(s) to  $Add\_Asap(Sh_x, O_i)$ .  $\square$

*Remark 1.* Notice that in Lemma 4 we do not take care of the particular structure which occurs when *add* creates two shelves of jobs of  $L_L$  as depicted Figure 1. However, it is easy to see that the proof can be adapted.

Now that  $\pi_1$  is built, we add the small remaining jobs ( $I \setminus I'$ ) using a list algorithm that scans all the organizations from time 0 and schedules as soon as possible any non scheduled job. Let  $\pi$  denote the obtained schedule.

**Theorem 2.** *The makespan of  $\pi$  is lower than  $5w/2$ .*

*Proof.* The proof is by induction on the cardinal of  $I \setminus I'$ . At the beginning,  $\pi_1$  is  $1/2$  compact, as proved in Lemma 4. Each time a job  $j$  is scheduled by the list algorithm, the obtained packing remains  $1/2$  compact because  $q_j \leq \frac{m}{2}$ . Thus it is clear that  $\pi$  is  $1/2$  compact.

Let us assume that the makespan of  $\pi$  is due to a job  $j \in I \setminus I'$  that starts at time  $s$ . As  $\pi$  is  $1/2$  compact, this implies that when scheduling job  $j$  we had  $t_i \geq s$  for any organization  $i$ . Thus, we have  $S(I) > \sum_{i=1}^N \frac{t_i}{2} \geq N \frac{s}{2}$ , implying that  $s < 2w$ , and thus that the makespan of  $\pi$  is lower than  $5w/2$ .  $\square$

## 5 Complexity

Phase 1 can be implemented in  $O(Nn + n \log(n))$ . Indeed, we first sort the high jobs in non increasing order of their processing times. Then, each layer can be created in  $O(n)$ . Phase 2 and phase 3 can also be implemented in  $O(Nn + n \log(n))$  by sorting the long (and extra long) jobs in non increasing order of their required processors. Thus  $\pi_0$  is constructed in  $O(Nn + n \log(n))$ .

The  $LS_{\pi_0}$  algorithm can be implemented in  $O(n \log(n))$ . Instead of scanning time by time and organization by organization, this algorithm can be implemented by maintaining a list that contains the set of “currently” scheduled jobs. The list contains 3-tuples  $(j, t, i)$  indicating that job  $j$  (scheduled on organization

$i$ ) finishes at time  $t$ . Thus, instead of scanning every time from 0 it is sufficient to maintain sorted this list according to the  $t$  values (in non decreasing order), and to only consider at every step the first element of the list. Then, it takes  $O(\log(n))$  to find a job  $j_0$  in the appropriate shelf that fits at time  $t$ , because a shelf can be created as a sorted array. It also takes  $O(\log(n))$  to insert the new event corresponding to the end of  $j_0$  in the list.

The last step, which turns  $\pi_1$  into the final schedule can also be implemented in  $O(n \log(n))$  using a similar global list of events. Notice that for any organization  $O_i$ , there exists a  $t_i$  such that before  $t_i$  the utilization is an arbitrary function strictly larger than  $m/2$ , and after  $t_i$  a non increasing after. Scheduling a small job before  $t_i$  would require additional data structure to handle the complex shape. Thus we do not schedule any small job before  $t_i$  as it is not necessary for achieving the  $5/2$  ratio. Therefore, we only add those events that happen after  $t_i$  when initializing the global list for this step. To summarize, for this step we only need to sort the small jobs in non increasing order of their required number of processors, and then apply the same global list algorithm.

The binary search on  $w$  to find the smallest  $w$  which is not rejected can be done in  $O(\log(np_{max}))$  as all the processing times can be assumed to be integers. Thus the overall complexity of the  $5w/2$  approximation is in  $O(\log(np_{max})n(N + \log(n)))$ .

## 6 Toward better approximation ratios

In this paper we provided a low cost  $5/2$ -approximation algorithm using a new approach. We discuss in this section how the proposed approach can be used for reaching better approximation bounds. The approach can be summarized in the two following main steps. The first one consists in constructing a  $1/2$  compact schedule  $\pi_1$  of the big jobs  $I'$  by creating a pre-allocation  $\pi_0$  and “compressing” it. Then, the remaining small jobs ( $I \setminus I'$ ) are added to  $\pi_1$  in a second step using the classical list scheduling algorithm  $LS$ .

We would like to recall the arguments that make our second step easy to analyze, and see what could be some other promising partitions. In our partition, the second step guaranties a makespan lower than  $5w/2$  because:

- adding a job  $j$  with  $q_j \leq m/2$  to a  $1/2$  compact schedule with  $LS$  produces another  $1/2$  compact schedule,
- if the makespan is due to a small job  $j_0$  that starts at time  $s_0$ , then  $s_0 \leq 2w$  (since the schedule is  $1/2$  compact), leading to a makespan lower than  $s_0 + p_{j_0} \leq 5w/2$ .

Let us now propose other partitions that could be considered. We could define  $I' = \{j | q_j > \alpha m_i \text{ or } p_j > \beta w\}$  with appropriate values  $0 < \alpha < 1$  and  $0 < \beta < 1$ . Then, the previous steps become:

1. construct a pre-allocation  $\pi_0$  of  $I'$  (for instance based on shelves and layers) and make sure that, when compressed using  $LS_{\pi_0}$ , the obtained schedule

- $\pi_1$  is  $1 - \alpha$  compact (meaning that for every organization, the utilization is greater than  $1 - \alpha$ , and then it is non-increasing),
- add the small remaining jobs ( $I \setminus I'$ ) using *LS*.

Thus, the makespan of jobs added in the second step would be bounded by  $b = (\frac{1}{1-\alpha} + \beta)w$ , implying that the makespan of the pre-allocation should also be bounded by  $b$ .

For example, we can target a  $7/3$  ratio by only studying how to pre-allocate  $I' = \{j | q_j > \frac{m_i}{2} \text{ or } p_j > \frac{w}{3}\}$ , or a ratio 2 by studying how to pre-allocate  $I' = \{j | q_j > \frac{m_i}{3} \text{ or } p_j > \frac{w}{2}\}$ . Obviously, if the preallocation is built using again shelves and layers, the difficulty will probably arise when merging the different types of jobs (high, extra long or long ones for example), and will may be only need to handle more particular cases.

Let us remark that this technique will not be easy to apply with (contiguous) rectangles, since the property of  $1/2$  compactness becomes hard to guarantee.

## References

- S. Zhuk, "Approximate algorithms to pack rectangles into several strips," *Discrete Mathematics and Applications*, vol. 16, no. 1, pp. 73–85, 2006.
- E. Coffman Jr, M. Garey, D. Johnson, and R. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM J. Comput.*, vol. 9, p. 808, 1980.
- I. Schiermeyer, "Reverse-fit: A 2-optimal algorithm for packing rectangles," *Lecture Notes in Computer Science*, pp. 290–290, 1994.
- A. Steinberg, "A strip-packing algorithm with absolute performance bound 2," *SIAM Journal on Computing*, vol. 26, p. 401, 1997.
- C. Kenyon and E. Rémila, "A near-optimal solution to a two-dimensional cutting stock problem," *Mathematics of Operations Research*, pp. 645–656, 2000.
- K. Jansen and R. Solis-Oba, "New approximability results for 2-dimensional packing problems," *Lecture Notes in Computer Science*, vol. 4708, p. 103, 2007.
- D. Ye, X. Han, and G. Zhang, "On-Line Multiple-Strip Packing," in *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*, p. 165, Springer, 2009.
- M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, and D. Trystram, "Approximation algorithm for multiple strip packing," *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA)*, 2009.
- M. Garey and R. Graham, "Bounds for multiprocessor scheduling with resource constraints," *SIAM J. Comput.*, vol. 4, no. 2, pp. 187–200, 1975.
- U. Schwiegelshohn, A. Tchernykh, and R. Yahyapour, "Online scheduling in grids," in *Proceedings of IPDPS*, pp. 1–10, 2008.
- P.-F. Dutot, F. Pascual, K. Rzdca, and D. Trystram, "Approximation algorithms for the multi-organization scheduling problem," *Submitted to: IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2010.
- P.-F. Dutot, G. Mounié, and D. Trystram, *Handbook of Scheduling*, ch. Scheduling Parallel Tasks: Approximation Algorithms. CRC Press, 2004.
- D. S. Hochbaum and D. B. Shmoys, "A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach," *SIAM J. Comput.*, vol. 17, no. 3, pp. 539–551, 1988.