

# Tight approximation for scheduling parallel job on identical clusters

Marin Bougeret\*, Pierre-Francois Dutot†, Klaus Jansen‡,   
Christina Robenek‡ Denis Trystram†

January 5, 2012

LIRMM Research report *RR* – 12001

## Abstract

In the grid computing paradigm, several clusters share their computing resources in order to distribute the workload. Each of the  $N$  cluster is a set of  $m$  identical processors (connected by a local interconnection network), and  $n$  parallel jobs are submitted to a centralized queue. A job  $J_j$  requires  $q_j$  processors during  $p_j$  units of time. We consider the Multiple Cluster Scheduling Problem (*MCSP*), where the objective is to schedule all the jobs in the clusters, minimizing the maximum completion time (makespan). This problem is closely related to the multiple strip packing problem, where the objective is to pack  $n$  rectangles into  $m$  strips of width 1, minimizing the maximum height over all the strips.

*MCSP* is 2-inapproximable (unless  $P = NP$ ), and several approximation algorithm have been proposed. However, ratio 2 has only been reached by algorithms that use extremely costly and complex subroutines as "black boxes" (for example area maximization subroutines on a constant ( $\approx 10^4$ ) number of bins, of *PTAS* for the classical  $P||C_{max}$  problem).

Thus, our objective is to find a reasonable restriction of *MCSP* where the inapproximability lower bound could be tightened in almost linear time. In this spirit we study a restriction of *MCSP* where jobs do not require strictly more than half of the processors, and we provide for this problem a 2-approximation running in  $O(\log_2(nh_{max})n(N + \log(n)))$ , where  $h_{max}$  is the maximum processing time of a job. This result is somehow optimal, as this restriction of *MCSP* (and even simpler ones, where jobs require less than  $\frac{m}{c}$ ,  $c \in \mathbb{N}$ ,  $c \geq 2$ ) remain 2-inapproximable unless  $\mathcal{P} = \mathcal{NP}$ .

---

\*LIRMM, Montpellier

‡Department of Computer Science, University Kiel, 24118 Kiel, Germany

†LIG, Grenoble, France

# 1 Introduction

## 1.1 Problem statement

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, so that the jobs of the next batch can be processed as soon as possible. Many such computational grid systems are available all over the world, and the efficient management of the resources is a crucial problem.

Let us start by stating the Multiple Cluster Scheduling Problem (*MCSP*) more formally (see Figure 1).

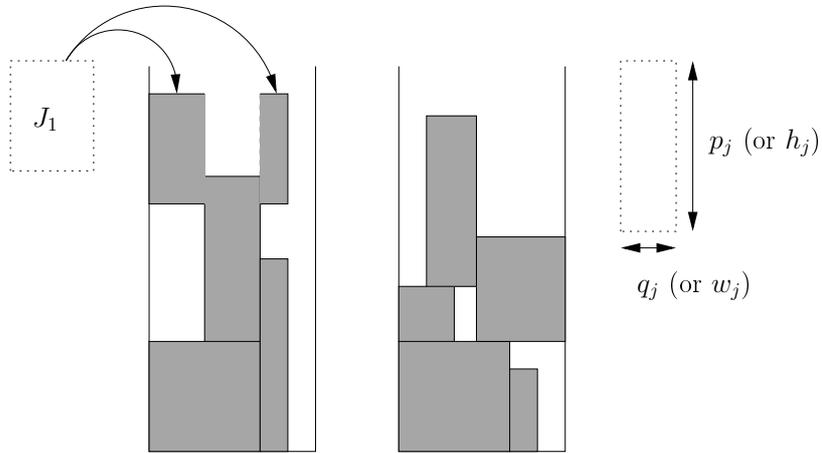


Figure 1: Example (for  $n = 9$  jobs and  $N = 2$  clusters) of a solution that is feasible for the *MCSP* and not feasible for the *MSPP*. Notice that  $J_1$  is packed in a "non continuous" way (using non consecutive indexes of processors).

**Definition 1** (*MCSP*). We are given  $n$  parallel rigid jobs  $J_j$ ,  $1 \leq j \leq n$ , and  $N$  clusters. A job  $J_j$  requires  $q_j$  processors during  $p_j$  units of time, and each cluster owns  $m$  identical processors. The objective is to schedule all the jobs in the clusters, minimizing the maximum completion time (makespan). Constraints are:

1. the  $q_j$  processors allocated to job  $J_j$  must belong to the same cluster
2. at any time, the total number of used processors in any cluster must be lower or equal to  $m$

We now recall the Multiple Strip Packing problem (*MSPP*), which is closely related to *MCSP*.

**Definition 2 (MSPP).** We are given  $n$  rectangles  $r_j$ ,  $1 \leq j \leq n$ , and  $N$  strips. Rectangle  $r_j$  have height  $h_j$  and width  $w_j$ , and all the strips have width 1. The objective is to pack all the rectangles in the strips such that the maximum reached height is minimized. Constraints are

1. a rectangle must be entirely packed in a strip (saying it differently cannot be split between two strips)
2. at any level of any strips, the total width of packed rectangle must be lower or equal to 1
3. a rectangle must be allocated "contiguously"

Thus, the only difference between *MCSP* and *MSPP* is constraint 3), which in terms of job scheduling amounts to force jobs to use consecutive indexes of processors (see Figure 1). Of course, results for *MCSP* generally not apply to *MSPP*, because of the additional contiguous constraint. The converse is also not clear, as ratio of approximation algorithms for *MSPP* may not be preserved when considering *MCSP* (optimal value of an *MCSP* instance may be strictly better than the corresponding one for *MSPP*). However, as we can notice in Figure 2, many results for *MSPP* directly apply to *MCSP*, as the proposed algorithms build contiguous schedules that are compared to non-contiguous optimal solutions.

We chose to adopt the "packing" vocabulary in this paper, so that solutions can be described using classical vocabulary of packing problems (like shelves for example). Thus, the problem treated in this paper (*MCSP*) is seen as the *MSPP*, without constraint 3).

## 1.2 Related Work

As shown in [Zhu06] using a gap reduction from the 2 partition problem, *MCSP* (and *MSPP*) are 2-inapproximable in polynomial time unless  $\mathcal{P} = \mathcal{NP}$ , even for  $N = 2$ . The main positive results for *MCSP* are summarized in Figure 2.

We must distinguish the 3-approximation of [STY08] and the  $\frac{5}{2}$ -approximation of [BDJ<sup>+</sup>10] that have a low computational complexity (these algorithms are usable on real size instances) from the 2-approximation in [BDJ<sup>+</sup>09] and the  $2 + \epsilon$ -approximation in [YHZ09].

At a first sight, the best results seem to be the 2-approximation in [BDJ<sup>+</sup>09] and the  $2 + \epsilon$ -approximation in [YHZ09]. However, the 2-approximation requires using high running time algorithm when the number of clusters is lower than a huge constant  $N_0$ . Thus, any exponential dependency in  $N_0$  is hidden, and the value of this constant ( $N_0 \approx 10^4$ ) makes this algorithm impossible to use. Moreover, the  $2 + \epsilon$ -approximation requires solving the famous  $P//C_{max}$  problem (which is makespan minimization when scheduling sequential jobs on identical machines) with a ratio  $1 + \frac{\epsilon}{2}$ . Thus, to give an rough idea, applying this technique with  $\epsilon = \frac{1}{3}$  would lead to a  $\Omega(n^{36})$  algorithm, using the famous PTAS of [HS87].<sup>1</sup>

---

<sup>1</sup>even if some recent advances in the PTAS design for  $P//C_{max}$  allowed to decrease the

Problem	Ratio	Remarks	Source
$MCSP, MSPP$	$2\rho$	Need solving $P//C_{max}$ with a ratio $\rho$	[YHZ09]
$MCSP$	$5/2$	Fast algorithm	[BDJ <sup>+</sup> 10]
$MCSP, MSPP$	2	Costly algorithm	[BDJ <sup>+</sup> 09]
$MCSP, MSPP$	AFPTAS	Additive constant in $\mathcal{O}(\frac{1}{\epsilon^2})$ , and in $\mathcal{O}(1)$ for large values of $N$	[BDJ <sup>+</sup> 09]
$MCSP$	3	Fast (and decentralized) algorithm that handle clusters having different size	[STY08]
$MCSP$	2	<b>Requires <math>\max_j w_j \leq \frac{1}{2}</math></b> Fast algorithm	this paper

Figure 2: Main results

### 1.3 Motivations and contributions

As explained below, the two previous results of [BDJ<sup>+</sup>09, YHZ09] both require extremely high running time, and do not provide insight on  $MCSP$  because of these black boxes subroutines. Thus, we followed in [BDJ<sup>+</sup>10] another approach by looking for fast and "direct" algorithms for  $MCSP$ . Our previous  $\frac{5}{2}$ -approximation in [BDJ<sup>+</sup>10] is based on the discarding technique presented in Section 2.2. What we call discarding technique is a classical framework in scheduling problems. The idea is to define properly a set of "negligeable" items (items are rectangles here), and to prove that it is possible to add these items only at the end of the algorithm without degrading the approximation ratio. Thus, the effort can be focused on the set  $I'$  of remaining "large" items, that are generally more structured.

The  $\frac{5}{2}$ -approximation was obtained through a basic application of the technique (*i.e.* with a set  $I'$  containing only really huge rectangles, for example rectangles whose width is larger than  $\frac{1}{2}$ ). As we believe that the discarding technique of Section 2.2 is well suited for the  $MCSP$  problem, we are intended to apply it again using a more "challenging" set  $I'$ . A natural direction would be to improve the  $\frac{5}{2}$  ratio for  $MCSP$  by targeting a *fixed* ratio  $\rho < \frac{5}{2}$ . Typically, one could target  $\rho = \frac{7}{3}$  by defining the small jobs as those whose length is lower than  $\frac{1}{3}$  (instead of  $\frac{1}{2}$ ). However, as the relative performance improvement is getting smaller, and the difficulty of these "ratio tailored" proofs is likely to increase rapidly, we consider here a different approach.

Our objective is to find a reasonable restriction of  $MCSP$  where the inapproximability lower bound could be tightened in almost linear time. In this spirit we study a restriction of  $MCSP$  where all rectangles have width lower than  $\frac{1}{2}$  (*i.e.* jobs submitted to clusters do not require strictly more than half of the processors), and we provide for this problem a very fast 2-approximation

---

asymptotic dependencies in  $\frac{1}{\epsilon}$  (like  $2^{\mathcal{O}(\frac{1}{\epsilon^2} \log^3(\frac{1}{\epsilon}))}$ ) in [Jan09], the running time of these new algorithms remain very high due to constants

running in  $O(\log_2(nh_{max})n(N + \log(n)))$ , where  $h_{max}$  is the maximum height of the rectangles. It turns out that this result is somehow optimal, as this restriction of *MCSP* (and even simpler ones, where width of rectangles is lower than  $\frac{1}{c}$ ,  $c \in \mathbb{N}, c \geq 2$ ) remain 2-inapproximable unless  $\mathcal{P} = \mathcal{NP}$ .

## 2 General principles

In this section, we generalize the framework used in the  $\frac{5}{2}$ -approximation of [BDJ<sup>+</sup>10]. This framework will be applied in Section 3 to get the 2-approximation.

### 2.1 Preliminaries

Recall that our objective is to (non contiguously) pack a set  $I$  of  $n$  rectangles  $r_j$  into  $N$  strips of width 1. Rectangle  $r_j$  has a height  $h_j$  and a width  $w_j$ . We denote by  $s(r_j) = w_j h_j$  the surface of  $r_j$ . These notations are extended to  $W(X)$ ,  $H(X)$  and  $S(X)$  (where  $X$  is a set of rectangles), which denote the sum of the widths (resp. heights, surfaces) of rectangles in  $X$ .

A *layer* is a set of rectangles packed one on top of the other in the same strip (as depicted Figure 3). The height of a layer *Lay* is  $H(Lay)$ , the sum of the height of all the rectangles in *Lay*. A *shelf* is a set of rectangles that are packed in the same strip, such as the bottom level of all the rectangles is the same. Even if it is not relevant for the non-contiguous case, we consider for the sake of simplicity that in a shelf, the right side of any rectangle (except the right most one) is adjacent to the left side of the next rectangle in the shelf. Given a shelf *sh* (*sh* denotes the set of rectangles in the shelf), the value  $W(sh)$  is called the *width* of *sh*. Packing a shelf at level  $l$  means that all the rectangles of the shelf have their bottom at level  $l$ . A *bin* is a rectangular area that can be seen as reserved space in a particular strip for packing rectangles. As a bin always has width 1, we define a bin by giving its height  $h_b$ , its bottom level  $l_b$  and the index  $i_b$  of the strip it belongs to. Packing a shelf *sh* in a bin  $b$  means that *sh* is packed in strip  $S_{i_b}$  at level  $l_b$ . Moreover we always guarantee that the height of any rectangle of *sh* is lower than  $h_b$ .

The *utilization*  $u_i^\pi(l)$  of a packing  $\pi$  in strip  $S_i$  at level  $l$  (sometimes simply denoted by  $u(l)$  or  $u_i(l)$ ) is the sum of the width of all the rectangles packed in  $S_i$  that cut the horizontal line-level  $l$  (see Figure 3). Of course we have  $0 \leq u_i^\pi(l) \leq 1$  for any  $l$  and  $i$ .

Let us now describe three useful procedures. The *CreateLayer*( $X, h$ ) procedure creates a layer *Lay* (using rectangles of  $X$ ) of height at most  $h$ , using a Best Fit (according to the height) policy (BFH). Thus, *CreateLayer*( $X, h$ ) adds at each step the highest rectangle that fits. Of course, the layer produced by the procedure is such that  $H(Lay) \leq h$ . Moreover, notice that we will always pack the layers in the strips with **narrowest rectangles on the top**. The *CreateShelf*( $X, w$ ) creates a shelf *sh* (using rectangles of  $X$ ) of width at most  $w$ , using the Best Fit (according to the width) policy (BFW). Thus, *CreateShelf*( $X, w$ ) adds at each step the widest rectangle that fits. Of course,

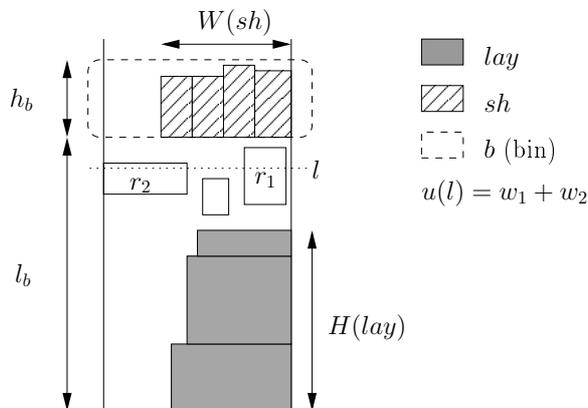


Figure 3: Example of a layer, a shelf, a bin and of the utilization function.  $sh$  is packed in  $b$ .

the shelf produced by the procedure is such that  $W(sh) \leq w$ . Throughout the paper, we consider that the sets of jobs used as parameters in the algorithms are modified after the calls.

Let us now state a standard lemma about the efficiency of the “best fit” policies.

**Lemma 3.** *Let  $Sh$  denote the shelf created by  $CreateShelf(X, w)$ . If the  $k$  widest rectangles of  $X$  are added to  $sh$ , then  $W(Sh) > \frac{k}{k+1}w$ .*

*Proof.* Let  $x$  be the cardinality of  $X$ . Let us assume that  $w_i \geq w_{i+1}$  for  $1 \leq i < x$ . Let  $i_0 \geq k + 1$  be the first index such that  $r_{i_0}$  is not in  $Sh$ . Let  $a = \sum_{i=1}^{i_0-1} w_i$ . We have  $W(Sh) \geq a \geq (i_0 - 1)w_{i_0} > (i_0 - 1)(w - a)$  leading to  $a > \frac{i_0-1}{i_0}w \geq \frac{k}{k+1}w$ .  $\square$

## 2.2 Discarding technique applied to MCSP

### 2.2.1 How to pack all rectangles in three steps

Discarding techniques are common for solving packing and scheduling problem. As mentioned before, the idea is to define properly a set of “small” items (rectangles here), and to prove that adding these small items only at the end of the algorithm will not degrade the approximation ratio. Thus, the effort can be focused on the remaining “large” items. In this section we present an adaptation of this general technique to the context of non-contiguous multiple strip packing. As usual, the set of big rectangles  $I'(\alpha, \beta) \subset I$  depends on parameters ( $\alpha$  and  $\beta$  here) that will be chosen carefully. The larger the set  $I'(\alpha, \beta)$  is, the better the approximation ratio will be (as the remaining small rectangles become really negligible).

In order to partition rectangles according to their height, we need to use the well-known dual approximation technique [HS88], and we denote by  $v$  the guess of the optimal value. Given an instance  $I$ , let  $L_{WD} = \{w_j > \alpha\}$  be the set of wide rectangles,  $L_H = \{r_j > \beta v\}$  be the set of high rectangles, and  $I' = L_{WD} \cup L_H$  be the set of big rectangles, with  $0 < \alpha < 1$  and  $0 < \beta < 1$ . Let  $r(\alpha, \beta) = (\frac{1}{1-\alpha} + \beta)$  be the approximation ratio we target (the origin of this formula will be explained in Section 2.2.2). We also need the following definition.

**Definition 4.** A packing is  $x$ -compact (see Figure 4) if and only if for every strip  $S_i$  there exists a level  $l_i$  such that for all  $l \leq l_i$ ,  $u_i(l) > x$  and  $u_i$  restricted to  $l > l_i$  is non-increasing.

Let us now describe the three main steps of our approach. Notice that what we call a preallocation is a "normal" packing (*i.e.* that define the bottom level of each rectangle, which is sufficient) that is based on simple structures like shelves and layers. We will prove that to get a  $r(\alpha, \beta) = (\frac{1}{1-\alpha} + \beta)$  ratio, it is sufficient to:

- a) construct a preallocation  $\pi_0$  of  $I'$  that fits in  $r(\alpha, \beta)v$ , and such that rectangles of  $L_{WD} \subset I'$  are already packed in a  $(1 - \alpha)$ -compact way
- b) turn  $\pi_0$  into a  $(1 - \alpha)$ -compact packing  $\pi_1$  by repacking rectangles of  $I' \setminus L_{WD}$  using the list algorithm  $LS_{\pi_0}$  of Lemma 5
- c) add the small remaining rectangles ( $I \setminus I'$ ) using  $LS$  (see Lemma 6)

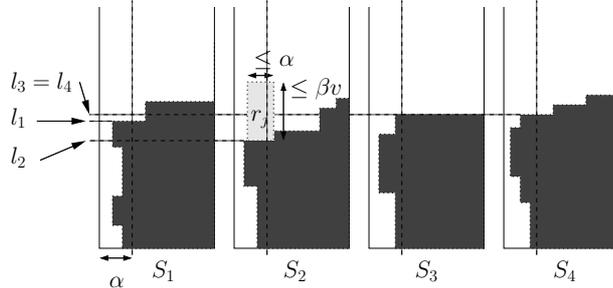


Figure 4: Example showing a  $(1 - \alpha)$  compact packing, and why step c) is simple. Indeed, adding as soon as possible a small rectangle  $r_j$  (having  $h_j \leq \beta v$  and  $w_j \leq \alpha$ ) to a  $(1 - \alpha)$  compact packing cannot exceed  $v(\frac{1}{1-\alpha} + \beta)$ . The  $l_i$  values are defined according to Definition 4.

Step a) is the most difficult one. Thus, Section 3 is entirely devoted to the construction of  $\pi_0$  (for  $(\alpha, \beta)$  equal to  $(\frac{1}{3}, \frac{1}{2})$ ). Of course, building the preallocation becomes harder when  $\alpha$  and  $\beta$  are small, as the number of rectangle of  $I'$  increases and  $r(\alpha, \beta)$  decreases. Roughly speaking, the simple shapes of rectangles of  $I'$  allows us to construct  $\pi_0$  with a simple structure. We will denote by  $\pi_0^i$  the set of rectangles packed by  $\pi_0$  in  $S_i$ .

### 2.2.2 Proving steps b) and c)

We now prove that applying steps b) and c) leads to a  $r(\alpha, \beta)$  ratio. In this section, we suppose that we are given a guess  $v$ , and a packing  $\pi_0$  (called the preallocation) of  $I' = L_{WD} \cup L_H$  that fits in  $r(\alpha, \beta)v$ , and such that rectangles of  $L_{WD} \subset I'$  are already packed in a  $(1 - \alpha)$ -compact way. We consider step b): how to turn  $\pi_0$  into a  $(1 - \alpha)$ -compact packing.

**Lemma 5** (Step b)). *Let  $\pi_0$  be the preallocation of  $I'$  constructed in Step a). Let  $\widehat{\pi}_1 = \pi_0 \cap L_{WD}$  denote  $\pi_0$  when keeping only rectangles of  $L_{WD}$ . Recall that  $\widehat{\pi}_1$  is already a  $(1 - \alpha)$ -compact packing of rectangles of  $L_{WD}$ .*

*Then, we can complete  $\widehat{\pi}_1$  into a  $(1 - \alpha)$ -compact packing  $\pi_1$  of  $I'$ , such that the height of  $\pi_1$  is lower or equal to the height of  $\pi_0$ .*

*Proof.* Let us define the  $LS_{\pi_0}$  algorithm that adds rectangles of  $I' \setminus L_{WD}$ . Let us consider a single strip  $S_i$ . Let  $\pi_0^i$  denote  $\pi_0$  restricted to  $S_i$ , and  $\widehat{\pi}_1^i$  denote  $\widehat{\pi}_1$  restricted to  $S_i$ . Let  $X = \{r_1, \dots, r_p\}$  be the set of preallocated rectangles of  $I' \setminus L_{WD}$  that we have to add to  $S_i$ . We assume that  $lvl(j) \leq lvl(j + 1)$ , where  $lvl(j)$  is the bottom level of  $r_j$  in  $\pi_0$ .

For our considered strip  $S_i$ , the  $LS_{\pi_0}$  algorithm executes  $AddAsap(r_j, \widehat{\pi}_1^i)$ , for  $1 \leq j \leq p$ , where  $AddAsap(r, \widehat{\pi}_1^i)$  adds rectangle  $r$  to  $\widehat{\pi}_1^i$  (in  $S_i$ ) at the smallest possible level. Notice first that adding with  $AddAsap$  a rectangle  $r_j$  with  $w_j \leq \alpha$  to a  $(1 - \alpha)$ -compact packing creates another  $(1 - \alpha)$ -compact packing. Thus it is clear that  $\pi_1$  is  $(1 - \alpha)$ -compact.

For any  $1 \leq j \leq p$ , let  $(\widehat{\pi}_1^i, j)$  denote the packing in  $S_i$  just before adding  $r_j$  with  $AddAsap$ , and let  $(\pi_0^i, j)$  denote the packing  $\pi_0^i \cap (L_{WD} \cup \{r_1, \dots, r_{j-1}\})$ . Let us prove by induction on  $j \in \{1, \dots, p\}$  that  $u^{(\widehat{\pi}_1^i, j)}(l) \leq u^{(\pi_0^i, j)}(l)$ , for any  $l \geq lvl(j)$ . The definition of  $\widehat{\pi}_1^i$  gives the property for  $j = 1$  (we even have an equality). Let us suppose that the property is true for  $j$ , and prove it for  $j + 1$ . Let  $l \geq lvl(j + 1)$ . The induction property for rank  $j$  implies that  $r_j$  is added by  $AddAsap$  at a level lower or equal to  $lvl(j)$ . Thus, if  $r_j$  intersects  $l$  in  $(\widehat{\pi}_1^i, j + 1)$ , then it also occurs in  $(\pi_0^i, j + 1)$ . Thus in this case we have

$$\begin{aligned} u^{(\widehat{\pi}_1^i, j+1)}(l) &= u^{(\widehat{\pi}_1^i, j)}(l) + w_j \\ &\leq u^{(\pi_0^i, j)}(l) + w_j \\ &= u^{(\pi_0^i, j+1)}(l) \end{aligned}$$

If  $r_j$  does not intersect  $l$  in  $(\widehat{\pi}_1^i, j)$ , then clearly  $u^{(\widehat{\pi}_1^i, j+1)}(l) = u^{(\widehat{\pi}_1^i, j)}(l) \leq u^{(\pi_0^i, j)}(l) \leq u^{(\pi_0^i, j+1)}(l)$

Thus we proved that for any  $1 \leq j \leq p$  we have  $u^{(\widehat{\pi}_1^i, j)}(l) \leq u^{(\pi_0^i, j)}(l)$  for any  $l \geq lvl(j)$ , implying that every  $r_j$  is added by  $AddAsap$  at a level lower or equal to  $lvl(j)$ . Thus, the height of  $\pi_1$  is lower or equal to the height of  $\pi_0$   $\square$

We now prove in Lemma 6 that after adding rectangles in step c), the height of the packing do not exceed  $r(\alpha, \beta)v = (\frac{1}{1-\alpha} + \beta)v$ . This explains why the height of the pre-allocation should also be bounded by  $r(\alpha, \beta)v$ .

**Lemma 6** (Step c)). *Let  $\pi_1$  be a  $(1 - \alpha)$ -compact packing of  $I'$ . Adding to  $\pi_1$  rectangles of  $I \setminus I'$  with a List Scheduling algorithm (LS) leads to a packing  $\pi$  having height lower than  $\max(\text{height}(\pi_1), v(\frac{1}{1-\alpha} + \beta))$ .*

*Proof.* The LS algorithm scans all the strips from level 0, and at any level adds any rectangle of  $I \setminus I'$  that fits. Notice that the final packing  $\pi$  is  $(1 - \alpha)$ -compact, since we add rectangles  $r_j$  with  $w_j \leq \alpha$  to an  $(1 - \alpha)$ -compact packing.

Let us assume that the height of  $\pi$  is due to a rectangle  $r_j \in I \setminus I'$  that starts at level  $s$ . This implies that when packing  $r_j$  we had  $l_i \geq s$  for any strip  $i$  (with  $l_i$  defined as in Definition 4). According to this definition we have  $u_i(l) > 1 - \alpha$  for any  $l \leq l_i$ . Thus, we have  $S(I) > \sum_{i=1}^N l_i(1 - \alpha) \geq N(1 - \alpha)s$ , implying that  $s < v\frac{1}{1-\alpha}$ , and thus that of height of  $\pi$  is lower or equal to  $s + \max_{j \in I \setminus I'} h_j \leq v(\frac{1}{1-\alpha} + \beta)$ .  $\square$

Thus, we now apply this framework with  $\alpha = \frac{1}{3}$  and  $\beta = \frac{1}{2}$  to get a 2-approximation.

### 3 A 2-approximation for a special case of MCSP

#### 3.1 Hardness

As explained in the related work, the  $2 + \epsilon$ -approximation in [YHZ09] and the 2-approximation we recently proposed in [BDJ<sup>+</sup>09] are rather complexity results than practical algorithms. We aim at constructing a low cost algorithm that could be used in a practical context. Thus, we are looking for a (reasonable) restriction of the MCSP that would help to tight the bounds, and we consider that all the rectangles have width lower or equal to  $1/2$ .

**Lemma 7.** *The MCSP where every rectangle has width lower (or equal) to  $\frac{1}{2}$  has no polynomial algorithm with a ratio strictly better than 2, unless  $P = NP$ .*

*Proof.* As in [Zhu06] for the general version, we construct a gap reduction from the 2-partition problem. Let  $\{x_1, \dots, x_n\} \subset \mathbb{N}^n$  and  $a$  such that  $\sum_{i=1}^n x_i = 2a$ . Without loss of generality, let us assume that for any  $i$ ,  $x_i < a$ . In order to only have items with size at least two, we define  $x'_i = 2x_i$  for any  $i$ , and  $a' = 2a$ . We construct the following instance  $I_{MSP}$  of "restricted" MCSP. We chose  $N = 2$  strips, each strip having size  $2a' - 1$ . The set of rectangle is  $\{r_1, \dots, r_n, r_{n+1}, r_{n+2}\}$ , with  $w_i = x'_i$  for  $1 \leq i \leq n$ ,  $w_{n+1} = w_{n+2} = a' - 1$ , and  $h_i = 1$  for  $1 \leq i \leq n + 2$ . We have  $w_i \leq \frac{2a'-1}{2}$  for any  $i$ , as all the  $x_i$  are strictly lower than  $a$ . Notice that any solution to  $I_{MSP}$  that packs  $r_{n+1}$  and  $r_{n+2}$  is the same strip have a height of at least 2, as the available width of size 1 in that strip cannot be used by any rectangle.

Obviously, if there is a 2-partition, then  $Opt(I_{MSP}) = 1$ . Otherwise, as  $r_{n+1}$  and  $r_{n+2}$  cannot be packed together, we have  $Opt(I_{MSP}) = 2$   $\square$

The previous proof can easily be adapted for any fixed value  $c \in \mathbb{N}, c \geq 2$ . Therefore, the fast 2-approximation presented in this section is the best result we can hope, even for simpler versions of the MCSP.

### 3.2 Preliminaries

We follow the ideas presented in Section 2, and thus we re-use the notion of **layer**, **shelf**, **bin**, and the procedures named **CreateLayer** and **CreateShelf**.

We use the dual approximation technique [HS88], and we denote by  $v$  the guess of the optimal value. Conforming to the dual approximation technique, we will prove that either we pack  $I$  with a resulting height lower than  $2v$ , or  $v < Opt$ . Then, we will perform a binary search on  $v$  to turn the dual approximation algorithm into a classical approximation algorithm. 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  $v \geq Opt$ , and it is implicit that if one of the claimed properties is wrong during the execution, the considered  $v$  should be rejected.

Recall that all rectangles have  $w_j \leq \frac{1}{2}$ . Let us define the following sets:

- let  $L_{WD} = \{r_j | w_j > 1/3\}$  be the set of wide rectangles
- let  $L_{XH} = \{r_j | h_j > 2v/3\}$  be the set of extra high rectangles
- let  $L_H = \{r_j | 2v/3 \geq h_j > v/2\}$  be the set of high rectangles
- let  $L_B = (L_{XH} \cup L_H) \cap L_{WD}$  be the set of huge rectangles, and  $b = Card(L_B)$ .
- let  $I' = L_{WD} \cup L_{XH} \cup L_H$

Notice than we only consider the values  $v$  such that

- $W(L_{XH} \cup L_H) \leq N$
- $H(L_{WD}) \leq 2Nv$

As we expected, the set  $I'$  corresponds in our framework to the set of big rectangles for  $\alpha = \frac{1}{3}$  and  $\beta = \frac{1}{2}$ . The construction of the preallocation  $\pi_0$  of  $I'$  is presented from Section 3.3 to 3.5. The final steps to turn  $\pi_0$  into a  $\frac{2}{3}$ -compact packing  $\pi_1$  and to turn  $\pi_1$  into the final packing  $\pi$  are quickly described in Section 3.6, as they follow the steps presented in Section 2.2.

We now provide a two phases algorithm that builds the preallocation  $\pi_0$  of the rectangles of  $I'$ . Let  $\pi_0^i$  denote the set of rectangles packed in  $S_i$ . Phase 1 (see Section 3.3) preallocates rectangles of  $L_{WD}$ , and phase 2 (see Section 3.5) preallocates rectangles of  $L_H \cup L_{XH}$ .

### 3.3 Phase 1

#### 3.3.1 Description of phase 1

Phase 1 packs the rectangles of  $L_{WD}$  by calling for each strip (until  $L_{WD}$  is empty) two times  $CreateLayer(L_{WD}, 2v)$ . Let us denote by  $Lay_{2i-1}$  and  $Lay_{2i}$  the layers created in strip  $S_i$ . Let us say that  $Lay_{2i-1}$  is packed left justified, and  $Lay_{2i}$  is packed right justified. Moreover, each layer is repacked in non increasing order of the widths, such that the narrowest rectangles are packed on the top.

Let  $N_1$  denote the number of strips used in phase 1, and let  $i_1$  denote the index of the last created layer ( $Lay_{i_1}$  is of course in  $S_{N_1}$ ). Let  $L_H^1$  and  $L_{XH}^1$  denote the set of remaining rectangles after phase 1 of  $L_H$  and  $L_{XH}$ , respectively. Thus, for the moment we have  $\pi_0^i = Lay_{2i} \cup Lay_{2i-1}$  for all  $i \leq N_1$ .

#### 3.3.2 Analysis of phase 1

**Lemma 8.** *If  $\exists i_0 < i_1$  such that  $H(Lay_{i_0}) \leq \frac{3v}{2}$  then it is straightforward to preallocate  $I'$ .*

*Proof.* Let  $i_0 < i_1$  such that  $H(Lay_{i_0}) \leq \frac{3v}{2}$ . This implies that we ran out of rectangles of  $L_{Wd} \setminus (L_H \cup L_{XH})$  while creating layer  $i_0$ . Thus, because of the *BFH* order there are at least two rectangles of  $L_B$  in every layer  $Lay_i$ , for  $1 \leq i < i_1$ , implying that the width of high and extra high rectangles packed in each of these layers is strictly larger than  $2/3$ . Thus,  $W(\pi_0^i \cap (L_H \cup L_{XH})) > 4/3 > 1$  for  $1 \leq i < N_1$ . Thus, the total width of remaining high and extra high rectangles is lower than  $N - (N_1 - 1)$ .

Let us prove that we can pack all the remaining rectangles of  $I'$  (which are included in  $(L_H \cup L_{XH})$ ) in the remaining strips. For each  $i \in [[N_1 + 1, N]]$  we create two shelves in  $S_i$  (one at level 0 and one at level  $v$ ). If there are still some unpacked rectangles, then all the shelves are "full", that is the width of each shelf is larger than  $2/3$  (as all the width of any rectangle of  $L_H \cup L_{XH}$  is lower than  $1/3$ ). Thus, we have  $W(\pi_0^i \cap (L_H \cup L_{XH})) > 4/3 > 1$  (for  $N_1 + 1 \leq i \leq N$ ). This implies that the total width of remaining rectangles of  $L_H \cup L_{XH}$  (including those in strip  $S_{N_1}$ ) is now lower than 1. Thus, we can pack all of them in one shelf in  $S_{N_1}$ .  $\square$

From now we assume that  $H(Lay_i) > \frac{3w}{2}$  for all  $i < i_1$ . This implies that  $S(\pi_0^i) > v$  for  $i < N_1$ . Moreover, we have  $2Nv \geq H(L_{Wd}) \geq \sum_{i=1}^{N_1-1} H(\pi_0^i \cap L_{Wd}) > (N_1 - 1)2(3v/2)$ , implying  $N_1 < \frac{2}{3}N + 1$ .

**Lemma 9.** *If there is a rectangle of  $L_B$  in  $lay_{i_1-1}$ , then it is straightforward to preallocate  $I'$ .*

*Proof.* We first consider the case where there are two layers in strip  $N_1$ . Let us count the cumulative width of high and extra high rectangles that already packed. In the first  $N_1 - 1$  strips, we packed at least two rectangles of  $B$  in each layer, implying  $\sum_{1 \leq i \leq N_1-1} W(\pi_0^i \cap (L_{XH} \cup L_H)) \geq \frac{4}{3}(N_1 - 1)$ . In strip  $N_1$ , we

have  $W(\pi_0^{N_1} \cap (L_{XH} \cup L_H)) > 1/3$  by hypothesis. Let  $N_2 = N - N_1$ . As in Lemma 8, we create two shelves (using a widest first policy) of rectangles of  $L_{XH} \cup L_H$  in each strip  $S_{N_1+i}$ , for  $1 \leq i \leq N_2$ . If we don't run out of rectangles, the remaining width of high and extra high rectangles after packing the first  $2N_2 - 1$  shelves is strictly larger than 1. Given that each shelf has width at least  $\frac{3}{4}$  (see Lemma 3), we have  $N \geq W(L_{XH} \cup L_H) > \frac{4}{3}(N_1 - 1) + \frac{1}{3} + \frac{3}{4}(2N_2 - 1) + 1 = -\frac{N_1}{6} + \frac{3N}{2} - \frac{3}{4}$ , leading to  $N_1 \geq 3N - \frac{9}{2}$ . As  $N_1 < \frac{2}{3}N + 1$ , we conclude that  $3N - \frac{9}{2} < \frac{2N}{3} + 1$ , which is a contradiction for  $N \geq 3$ .

If  $N = 2$ , we have  $W(L_{XH} \cup L_H) \leq 2$  and  $H(L_{WD}) \leq 4v$ . Given that  $H(\pi_0^1 \cap L_{WD}) > 3v$ , we get  $H(\text{lay}_3) \leq v$  and  $H(\text{lay}_4) = 0$  which is a contradiction because we supposed that there were two layers in strip  $N_1$ .

The case where there is only one layer in strip  $N_1$  can be treated using the same arguments for  $N \geq 3$ . If  $N = 2$ , then we have (as before)  $H(\text{lay}_3) \leq v$ . Moreover,  $W(\pi_0^1 \cap (L_{XH} \cup L_H)) > \frac{4}{3} > 1$  because there are at least two rectangles of  $L_B$  in  $\text{lay}_1$  and one rectangle of  $L_B$  in  $\text{lay}_2$ . Thus, there is enough space in strip  $S_2$  to pack one shelf of  $L_{XH} \cup L_H$ , which is sufficient.  $\square$

Naturally we consider from now on that the area packed in the first  $N_1 - 1$  is strictly more than  $(N_1 - 1)v$ , and that there is no huge rectangle in the last two layers created by phase 1. It remains now to pack  $L_H^1 \cup L_{XH}^1$ . Notice that  $(L_H^1 \cup L_{XH}^1) \cap L_{WD} = \emptyset$  (we say that  $(L_H^1 \cup L_{XH}^1)$  contains purely high and extra high rectangles).

## 3.4 Packing techniques for (purely) high and extra high rectangles

### 3.4.1 Preliminaries

Let  $N_2 = N - N_1$  denote the number of free strips after phase 1. Roughly speaking, phase 2 packs shelves of high or extra high rectangles in each of the  $N_2$  last strips and merges some high or extra high rectangles with the ones packed in strip  $N_1$  (using the *Merge* procedure).

In this section we present a technique to fill  $\alpha$  empty strips with high or extra high rectangles. In the Section 3.5, we use this technique for  $\alpha = N_2$  (using strips  $S_{N_1+1} \dots S_N$ ) and an additional merging algorithm (that fills efficiently strip  $S_{N_1}$ ) to pack  $L_H^1 \cup L_{XH}^1$ .

Let us now introduce the procedure **GreedyPack(X, seq)**. Given an ordered sequence of bins *seq*, *GreedyPack* creates for each *empty* bin  $b \in \text{seq}$  a shelf of rectangles of  $X$  using *CreateShelf(X, 1)* and packs it into  $b$  (an example of a shelf packed in a bin is depicted Figure 3, Page 6). This procedure returns the last bin in which a shelf has been created, or *null* if no shelf is created. Notice that we will always use sequence of bins that have always width 1, and the same height  $h_b$  such that  $\max_{r_x \in X} h_x \leq h_b$ .

We now define the two sequences of bins  $\text{seq}_{XH}$  and  $\text{seq}_H$  that will be used by *GreedyPack*. Every bin of  $\text{seq}_{XH}$  (resp.  $\text{seq}_H$ ) will (possibly) contain one shelf of rectangles of  $L_{XH}$  (resp.  $L_H$ ). Notice that in a free strip it is possible

to pack two bins of height  $v$  (width of bins is always 1), three bins of height  $2v/3$ , or one bin of size  $v$  and one bin of size  $2v/3$ . Thus,  $seq_{XH}$  is composed of  $2\alpha$  bins  $(b_1, \dots, b_{2\alpha})$  of height  $v$ , considering that we created two bins of height one in each of the strips  $S_1, \dots, S_\alpha$ . More precisely, for all  $i$  we locate  $b_{2i-1}$  and  $b_{2i}$  in  $S_i$ , with  $b_{2i-x}$  at level  $v(1-x)$  for  $x \in \{0, 1\}$ . The sequence  $seq_H$  is composed of  $3\alpha$  bins  $(b'_1, \dots, b'_{3\alpha})$  of height  $2v/3$ , considering that we created three bins in each of the strips  $S_\alpha, \dots, S_1$ . It means that for all  $i \geq 1$ , bins  $b'_{3i-2}$ ,  $b'_{3i-1}$  and  $b'_{3i}$  are located in  $S_{\alpha-i+1}$ , with  $b'_{3i-x}$  at level  $\frac{2xv}{3}$  for  $x \in \{0, 2\}$ . This sequences of bins will be used in Lemma 11, and later in phase 2.

Finally, let us define the  $Add(X, S_{i_{last}})$  procedure that packs the set of rectangles  $X \subset L_H \setminus L_{Wd}$  in  $S_{i_{last}}$ . As one can see in Lemma 11,  $S_{i_{last}}$  is the last strip where *GreedyPack* created a shelf. Thus, we assume for the moment that  $S_{i_{last}}$  may only contain two different shapes of packing, and define the *Add* procedure accordingly.

In the first case  $S_{i_{last}}$  contains a first “full” shelf (full means that the surface of the shelf is at least  $v/2$ ) of rectangles of  $L_{XH}$  at level 0, and a shelf  $sh$  of rectangles of  $L_{XH}$  packed at level  $v$ , right justified. In this case, *Add* creates a shelf  $sh_1$  using  $CreateShelf(X, 1 - W(sh))$  and preallocate  $sh_1$  at level  $v$ , left justified.

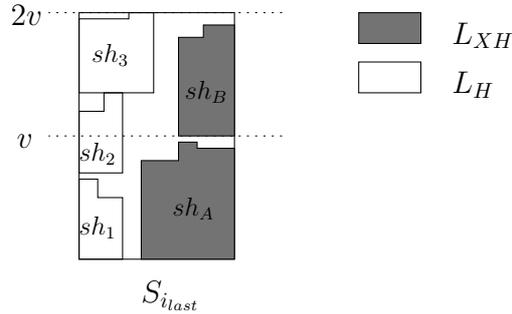


Figure 5: Example the  $add_2$  procedure.

In the second case (see Figure 5),  $S_{i_{last}}$  contains only a shelf of rectangles of  $L_{XH}$  packed at level 0, right justified. In this case, *Add* first moves some rectangles from  $sh$  to a new shelf  $\widehat{sh}$  until  $W(\widehat{sh}) \leq 2/3$ . Then, *Add* packs (right justified) the widest of these two shelves (denoted by  $sh_A$ ) at level 0, and the other one (denoted by  $sh_B$ ) at level  $v$ . Finally, *Add* creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, 1 - W(sh_A))$  and one shelf  $sh_3$  using  $CreateShelf(X, 1 - W(sh_B))$ . Then,  $sh_i$  is packed at level  $\frac{2v(i-1)}{3}$ , left justified. Notice that stacking shelves  $sh_1, sh_2, sh_3$  does not exceed  $2v$ .

We end preliminaries with the following Lemma about the efficiency of *Add*.

**Lemma 10.** *Let  $X \subset L_H \setminus L_{WD}$  and  $S_i$  be a strip packed as expected for  $Add(X, S_i)$ . Let  $\pi_0^i$  denote the rectangles packed in  $S_i$  before the call  $Add(X, S_i)$ . If  $X \neq \emptyset$  after calling the procedure, then  $S(\pi_0^i \cup X) > v$ .*

*Proof.* Remember that two cases are possible according to what is already packed in  $S_i$  before the call. Let us first suppose that there is one full shelf (of area strictly larger than  $v/2$ ) of extra high rectangles (at level 0) and another shelf  $sh$  of extra high rectangles at level  $v$ . Then,  $X \neq \emptyset$  after the call implies that  $W(X) > 1 - W(sh)$ , and we have  $S(\pi_0^i \cup X) > \frac{v}{2} + W(sh)\frac{2v}{3} + W(X)\frac{v}{2} > v$ .

Let us now suppose that  $S_i$  contains only one shelf  $sh$  of  $L_{XH}$  at level 0. Let  $sh_A, sh_B, sh_1, sh_2, sh_3$  be defined as described in the *Add* procedure. As  $W(sh_A) \leq \frac{2}{3}$  ( $W(sh_b) \leq \frac{2}{3}$  is also true), and  $X \cap L_{WD} = \emptyset$ ,  $sh_1$  and  $sh_2$  contain at least one rectangle, implying that  $W(sh_1)$  and  $W(sh_2)$  are strictly larger than  $\frac{1-W(sh_A)}{2}$  according to Lemma 3. Moreover,  $X \neq \emptyset$  after the call implies that after creating  $sh_1$  and  $sh_2$  the total width of remaining rectangles of  $X$  was strictly larger than  $1 - W(sh_B)$ . Putting this together, we get  $S(\pi_0^i \cup X) > (1 - W(sh_A))\frac{v}{2} + (1 - W(sh_B))\frac{v}{2} + (W(sh_A) + W(sh_B))\frac{2v}{3} > v$ .  $\square$

### 3.4.2 Filling $\alpha$ empty strips with high and extra high rectangles

The next lemma shows how to fill  $\alpha$  free strips.

**Lemma 11.** *Let  $\widehat{L_{XH}} \subset L_{XH} \setminus L_{WD}$  and  $\widehat{L_H} \subset L_H \setminus L_{WD}$  be two sets of rectangles that we have to pack. Suppose that we execute the following calls:*

1.  $last = GreedyPack(\widehat{L_{XH}}, seq_{XH})$
2.  $GreedyPack(\widehat{L_H}, seq_H)$
3.  $Add(\widehat{L_H}, S_{i_{last}})$  where  $S_{i_{last}}$  denotes the strip containing bin *last*.

*Then, we get the following properties:*

- If  $\widehat{L_{XH}} \neq \emptyset$  after 1, then  $S(\widehat{L_{XH}}) > (\alpha + \frac{1}{6})v$
- Otherwise, if  $\widehat{L_H} \neq \emptyset$  after 3, then  $S(\widehat{L_{XH}} \cup \widehat{L_H}) > \alpha v$ .

**Remark 12.** *Notice that before the call to  $Add_2$ ,  $S_{i_{last}}$  is the only strip that maybe already contains high and extra high rectangles. Moreover, this can happen only if the last shelf of extra high rectangles is at level 0, (because otherwise the places corresponding to  $b'_{3i_{last}-2}$  and  $b'_{3i_{last}-2}$  are not completely free, and thus  $GreedyPack$  do not pack any high rectangles in these bins).*

**Remark 13.** *Let  $X$  such that  $X \cap L_{Wd} = \emptyset$  and let  $sh$  denote a shelf created by  $CreateShelf(X, 1)$ , supposing that we didn't run out of rectangle while creating the shelf. Then, according to Lemma 3, as at least three rectangles fit we have  $W(sh) > 3/4$ . Moreover, if  $X \subset L_{XH}$  then  $S(sh) > v/2$ , and if  $X \subset L_H$  then  $S(sh) > 3v/8$ .*

*Proof of lemma 11.* Let us first suppose that  $\widehat{L}_{XH} \neq \emptyset$  after step 1. It implies that after creating the first  $2\alpha - 1$  shelves of width at least  $3/4$  (according to Remark 13), the total remaining width of rectangles of  $\widehat{L}_{XH}$  was strictly larger than 1. Thus,  $S(\widehat{L}_{XH}) > \frac{3}{4}(2\alpha - 1)\frac{2v}{3} + \frac{2v}{3} = (\alpha + \frac{1}{6})v$ .

Let us now suppose that  $\widehat{L}_{XH} = \emptyset$  and  $\widehat{L}_H \neq \emptyset$  after step 3. Let  $sh$  denote the shelf of rectangles of  $\widehat{L}_{XH}$  contained in bin  $last$ . Remind that  $i_{last}$  is the index of the strip containing  $last$ . For all  $i \in [1, i_{last} - 1]$ ,  $S(\pi_0^i \cap (\widehat{L}_{XH} \cup \widehat{L}_H)) > 2\frac{v}{2} = v$ . For all  $i \in [i_{last} + 1, \alpha]$ ,  $S(\pi_0^i \cap (\widehat{L}_{XH} \cup \widehat{L}_H)) > 3\frac{3v}{8} > v$ . According to Lemma 10,  $\widehat{L}_H \neq \emptyset$  implies  $S(\pi_0^{i_{last}} \cap \widehat{L}_H) > v$ . Thus, we get that  $S(\widehat{L}_{XH} \cup \widehat{L}_H) > \alpha v$ .  $\square$

### 3.5 Phase 2

In phase 1 we preallocated  $L_{WD}$  in strips  $S_1, \dots, S_{N_1}$ . Recall that each layer created in phase 1 is sorted with the narrowest rectangles on the top. It remains now to preallocate  $L_{XH}^1 \cup L_H^1$  in  $S_{N_1}, \dots, S_N$ .

**Lemma 14.** *It is possible to preallocate  $L_{XH}^1 \cup L_H^1$  in  $S_{N_1}, \dots, S_N$  with a resulting height lower than  $2v$ .*

An example of the final packing is depicted Figure 6.

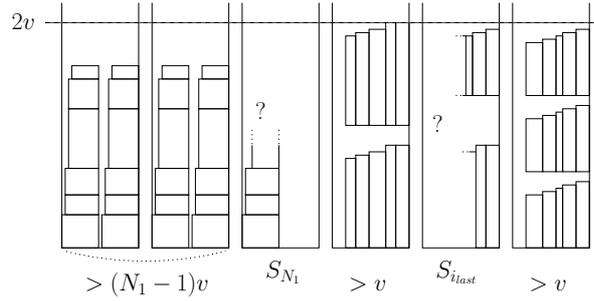


Figure 6: Example of a preallocation. Phase 1 optimally filled the  $(N_1 - 1)$  first strips using rectangles of  $L_{WD}$ . Phase 2 optimally filled strips  $N_1 + 1$  to  $i_{last} - 1$  using two shelves of extra high rectangles in each strip, and optimally filled strips  $i_{last} + 1$  to  $N$  using three shelves of high rectangles in each strip. Strips  $N_1$  and  $i_{last}$  will be carefully filled according to a case by case analysis.

The general idea to pack  $L_{XH}^1 \cup L_H^1$  is to call *GreedyPack* on strips  $S_{N_1+1}, \dots, S_N$ , and to finish the packing by carefully studying what was packed in  $S_{N_1}$  by phase 1. Thus, Lemma 14 is proved by case distinction according to the shape of the preallocation in  $S_{N_1}$ . Notice that the sequence of bins used by *GreedyPack* is the one described in Section 3.4.1, replacing strips  $S_1, \dots, S_\alpha$  by strips  $S_{N_1+1}, \dots, S_N$ .

For the sake of clarity, we define for several cases an appropriate **Merge** procedure that is responsible for adding rectangles in  $S_{N_1}$ . Finally, let us in-

roduce the notation  $av(l)$  to denote the available width at level  $l$  (in a given strip).

### 3.5.1 Cases where two layers are preallocated in $S_{N_1}$

Recall that  $i_1$  is the index of the last created layer in phase 1. As two layers are preallocated in  $S_{N_1}$ , we have here  $i_1 = 2N_1$ . Let  $p_1 = H(Lay_{i_1})$ ,  $p_2 = H(Lay_{i_1-1})$  and  $k = Card(Lay_{i_1})$ . Without loss of generality, let us denote by  $\{r_1, \dots, r_k\}$  the rectangles of  $Lay_{i_1}$ , with  $r_j, 1 \leq j \leq k$  sorted in non increasing order of their heights. Remember that we assume that the last two layers of phase 1 do not contain any rectangle of  $L_B$ . We proceed by case analysis according to the value of  $p_1$ .

**If**  $p_1 > \frac{4v}{3}$ .

The algorithm makes the following calls:

1.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
2.  $GreedyPack(L_H^1, seq_H)$
3.  $Add(L_H^1, S_{i_{last}})$

Let us prove that  $p_1 + p_2 > 3v$ . We have  $p_2 > 2 - h_k$  as  $r_k$  was not included in  $Lay_{i_1-1}$ , and  $p_1 \geq \max(kh_k, \frac{4v}{3})$ . Thus  $p_1 + p_2 \geq \max(2 + (k-1)h_k, 2 - h_k + \frac{4v}{3})$ . This last term is minimized for  $h_k = \frac{4v}{3k}$ , leading to  $p_1 + p_2 \geq 2 + \frac{4v}{3}(1 - \frac{1}{k})$  which is greater than  $3v$  for  $k \geq 4$ .

Moreover,  $k > 2$  as  $p > \frac{4v}{3}$  and no rectangles of  $L_B$  are packed in strip  $S_{N_1}$  according to Lemma 9. Thus, it remains to consider the case where  $k = 3$ . Given that there is at least 4 rectangles in  $Lay_{i_1-1}$  (as no rectangles of  $L_B$  are packed in strip  $N_1$ ) we have  $p_2 > 4h_1$ , and as  $h_1 > \frac{4v}{3k}$ , we get  $p_1 + p_2 > \frac{16v}{3k} + \frac{4v}{3} > \frac{28v}{9} > 3v$ .

Thus in this case it is not necessary to preallocate anything else in  $S_{N_1}$  as we have  $p_1 + p_2 > 3v$  implying  $S(\pi_0^{N_1}) > v$ . Then, we conclude that no rectangle of  $L_{XH}^1 \cup L_H^1$  remains after step 3 using Lemma 11.

**If**  $\frac{4v}{3} \geq p > v$ .

Let us first define the  $Merge(X)$  procedure for this case. In this case (see Figure 7 Page 17),  $merge(X)$  creates one shelf  $sh$  using  $CreateShelf(X, x)$ , where  $x = av(\frac{4v}{3})$ , and packs it at level  $\frac{4v}{3}$ , right justified.

The algorithm makes the following calls:

1.  $Merge(L_H^1)$
2.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
3.  $GreedyPack(L_H^1, seq_H)$
4.  $Add(L_H^1, S_{i_{last}})$

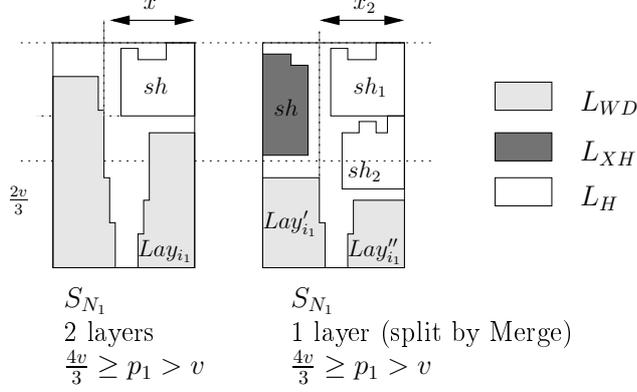


Figure 7: Example of *Merge* for two different cases.

Let us consider a first case where  $L_H^1$  is not empty after step 1. According to Lemma 3,  $W(sh) > \frac{x}{2}$ . After step 1, we have  $S(\pi_0^{N_1}) > \frac{1}{3}(p_1) + S(Lay_{i_1-1}) + S(sh)$ . Moreover,  $S(Lay_{i_1-1}) + S(sh) \geq (1-x)\frac{4v}{3} + (p_2 - \frac{4v}{3})\frac{1}{3} + \frac{xv}{4}$ , (see Figure 7 Page 17), which is decreasing in  $x$ . Thus, the lower bound is reached for  $x = \frac{2}{3}$ , and in this case  $S(\pi_0^{N_1}) > \frac{p_1+p_2}{3} + W(sh)\frac{v}{2} > \frac{5v}{6} + \frac{1}{3}\frac{v}{2} = v$ . Then, according to Lemma 11, no rectangle remains after step 4.

We now suppose that  $L_H^1$  is empty after step 1. Before 1,  $S(\pi_0^{N_1}) > \frac{p_1+p_2}{3} > (\frac{3}{2} + 1)\frac{v}{3} = \frac{5v}{6}$ . If  $L_{XH}^1 \neq \emptyset$  after step 2, then according to Lemma 11  $S(L_{XH}^1) > (N_2 + \frac{1}{6})v$  implying  $S(\pi_0^{N_1}) + S(L_{XH}) > (N_2 + 1)v$ . Thus in this case no rectangle remains after step 2.

**If  $v \geq p > \frac{2v}{3}$ .**

We first analyze the case where  $N_1 > 1$ . Let us consider the same algorithm as in the previous case (with in particular the same definition of  $x$ ). Let us first suppose that  $L_H^1 \neq \emptyset$  after step 1. We will bound the surface of rectangle preallocated in  $\pi_0^{N_1}$  after step 2 as before, except that we also take into account the  $N_1 - 1$  first strips. Thus, after step 2 we have  $S(\bigcup_{i=1}^{N_1} \pi_0^i) > (N_1 - 2)v + (H(Lay_{i_1-3}) + H(Lay_{i_1-2}) + p_1)\frac{1}{3} + S(Lay_{i_1-1}) + S(sh)$ . As before,  $S(Lay_{i_1-1}) + S(sh)$  is decreasing in  $x$ , and we replace  $x$  by  $\frac{2}{3}$ . Moreover, notice that  $H(Lay_{i_1-3}) + H(Lay_{i_1-2}) + p_1 > 4v$  as there is at least two rectangles in  $lay_{i_1}$ , and none of these rectangles has been included in  $Lay_{i_1-3}$  or  $Lay_{i_1-2}$ . Finally, we get  $S(\bigcup_{i=1}^{N_1} \pi_0^i) > (N_1 - 2)v + \frac{4v}{3} + \frac{p_2}{3} + \frac{1}{3}\frac{v}{2}$ . Using  $p_2 \geq \frac{3v}{2}$ , we get  $S(\bigcup_{i=1}^{N_1} \pi_0^i) > N_1v$ . The case where  $L_H^1$  is empty after step 1 can be adapted in the same way.

Let us now consider the case with  $N_1 = 1$ . In this case it is sufficient to create two shelves of rectangles of  $L_H^1 \cup L_{XH}^1$  in each  $S_i$  for  $2 \leq i \leq N$ . If we

don't run out of rectangles, the total width of high and extra high rectangles packed in each of these strips is strictly larger than  $\frac{3}{2}$ . Thus, the total width  $\lambda$  of remaining rectangles of  $L_H^1 \cup L_{XH}^1$  is at most  $N - \frac{3}{2}(N - 1)$  which is negative for  $N \geq 3$ . If  $N = 2$ ,  $\lambda \leq \frac{1}{2}$ , and we can finish packing  $L_H^1 \cup L_{XH}^1$  using one bin of width  $\frac{1}{2}$  and height  $v$  whose bottom is located at level  $v$ .

**If  $\frac{2v}{3} \geq p > 0$ .**

Let us first define the *Merge*( $X$ ) procedure for this case. If  $X \subset L_{XH}$ , *merge*( $X$ ) creates one shelf  $sh$  using *CreateShelf*( $X, x$ ), where  $x = av(v)$ , and packs it at level  $v$ , right justified.

If  $X \subset L_H$ , two sub-cases are possible according to what is packed in strip  $S_{N_1}$ . If no shelf of  $L_{XH}$  is preallocated in  $S_{N_1}$ , *merge*( $X$ ) creates two shelves  $sh_1$  and  $sh_2$  using *CreateShelf*( $X, x_1$ ) and *CreateShelf*( $X, x_2$ ) respectively, where  $x_i = av(\frac{(6-2i)v}{3})$ . Notice that if  $sh_2$  is not empty then  $W(sh_1) + W(sh_2) > x_1$  as all the rectangles of  $sh_2$  were not included in  $sh_1$ . Then,  $sh_i$  is packed at level  $\frac{(6-2iv)}{3}$ , right justified.

If there is a shelf  $sh$  of rectangles of  $L_{XH}$  preallocated in  $S_{N_1}$  (right justified, at level  $v$ ), *merge*( $X$ ) creates one shelf  $sh_1$  using *CreateShelf*( $X, x$ ) where  $x = av(v)$  ( $x$  takes into account the wide rectangles added in phase 1 and the extra high ones in  $sh$ ), and packs it at level  $v$ .

The algorithm makes the following calls:

1.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
2.  $Merge(L_{XH}^1)$
3.  $Merge(L_H^1)$
4.  $GreedyPack(L_H^1, seq_H)$
5.  $Add(L_H^1, S_{i_{last}})$

Let us prove that  $L_{XH}^1$  is empty after step 2 by contradiction. If  $L_{XH}^1$  is not empty after step 2, then we claim that  $S(I') > Nv$ . Let  $L_{XH}^{-1}$  and  $\bar{\pi}_0^{N_1}$  be respectively  $L_{XH}^1$  and  $\pi_0^{N_1}$  just before step 2. Just before step 2, the area packed in each of the  $N_2$  last strips is strictly larger than  $v$  as *GreedyPack*( $L_{XH}^1, seq_{XH}^1$ ) packs two shelves of area strictly larger than  $\frac{v}{2}$  in each strip. If  $L_{XH}^1$  is not empty after step 2 then  $W(L_{XH}^{-1}) > x$  (with  $x$  as defined just before for the *Merge* procedure) and  $S(\bar{\pi}_0^{N_1}) + S(L_{XH}^{-1}) > p_1 \frac{1}{3} + S(Lay_{i_1-1}) + S(L_{XH}^{-1})$ . As before,  $S(Lay_{i_1-1}) + S(L_{XH}^{-1}) > v(1-x) + (p_2 - v) \frac{1}{3} + \frac{2v}{3}x$  is decreasing in  $x$  and the minimum is reached for  $x = \frac{2}{3}$ . Replacing  $x$  by this value and using the fact that  $p_1 + p_2 > 2v$ , we get  $S(\bar{\pi}_0^{N_1}) + S(L_{XH}^{-1}) > \frac{2v}{3} + \frac{4v}{9} > v$ , implying  $S(I') > Nv$ .

Thus  $L_{XH}^1$  is empty after step 2. There is now two cases according to what is packed in  $S_{N_1}$ . Let us start with the first case where no extra high rectangle packed in  $S_{N_1}$  (*i.e.*  $L_{XH}$  is empty after step 1). Remind that in this case two

shelves ( $sh_1$  and  $sh_2$ ) of rectangles of  $L_H^1$  are created, and  $W(sh_1) + W(sh_2) > x_1$ . We will prove that  $L_H$  is empty after step 5 by contradiction. We first show that after step 3 we have  $S(\pi_0^{N_1}) > v$ , and then we will conclude using Lemma 11.

After phase 3 we get  $S(\pi_0^{N_1}) > p_1 \frac{1}{3} + S(Lay_{i_1-1}) + S(sh_1) + S(sh_2)$ . As before,  $S(Lay_{i_1-1}) + S(sh_1) + S(sh_2) > \frac{4v}{3}(1-x_1) + (p_2 - \frac{4v}{3})\frac{1}{3} + x_1 \frac{v}{2}$  is decreasing in  $x_1$  and the minimum is reached for  $x_1 = \frac{2}{3}$ . Replacing  $x_1$  we get  $S(\pi_0^{N_1}) > (p_1 + p_2 - \frac{4v}{3})\frac{1}{3} + \frac{4v}{9} + \frac{v}{3} > v$ . Then, we prove that step 4 must pack the remaining rectangles using Lemma 11. Thus in this case all the rectangles are packed.

In the second case some rectangles of  $L_{XH}^1$  are packed in  $S_{N_1}$  during step 2. Thus, the area packed (with rectangles of  $L_{XH}^1$ ) in each of the last  $N_2$  strips using *GreedyPack* is strictly larger than  $v$ . We prove by contradiction that  $L_H^1$  must be packed at step 3. We use the same argument as before: the case which minimizes our lower bound on  $S(pi_0^{N_1}) + S(L_H^1)$  occurs when all the rectangles of  $Lay_{i_1-1}$  have width  $\frac{1}{3}$ . Thus, if  $L_H^1$  is not packed at step 3 we get  $S(pi_0^{N_1}) + S(L_H^1) > v$ , leading as usual to  $S(I') > Nv$ .

### 3.5.2 Cases where one layer is preallocated in $S_{N_1}$

We consider now cases where  $i_1 = 2N_1 - 1$ . Let  $p_1 = H(Lay_{i_1})$ ,  $p_2 = H(Lay_{i_1-1})$ ,  $p_3 = H(Lay_{i_1-2})$  and  $k = \text{Card}(Lay_{i_1})$ . Without loss of generality, let us denote by  $\{r_1, \dots, r_k\}$  the set  $Lay_{i_1}$  (so  $p_1 = \sum_{1 \leq j \leq k} h_j$ ), with  $r_j$  sorted in non increasing order of their height. We proceed by case analysis according to the value of  $p_1$ . Remember that we assume that no rectangle of  $L_B$  in the last two layers of phase 1.

We start treating all these cases with one layer by stating some useful properties in Lemma 15.

**Lemma 15.** *For the cases with one layer in  $S_{N_1}$ , we have*

1. if  $N_1 = 1$  then it is straightforward to preallocate  $L_H^1 \cup L_{XH}^1$
2. for all the cases where  $p_1 > 1$ , then  $N_1 < N$
3. if  $p_1 > \lambda$  with  $\lambda > \frac{v}{2}$ , then  $p_1 + p_2 + p_3 > 3v + \lambda + \max(\frac{v}{5}, v - 2\frac{\lambda}{k})$ .

Notice that property 3 states that we can improve the naive bound  $p_1 + p_2 + p_3 > 3v + \lambda$ .

*Proof.* We start with property 1. If  $N_1 = 1$  we can pack two shelves of rectangles of  $L_H^1 \cup L_{XH}^1$  in each strip  $S_i, i \geq 2$ , implying that the width packed rectangles of  $L_H^1 \cup L_{XH}^1$  is at least  $\frac{3}{2}$  in each strip. Therefore, the remaining width of rectangles of  $L_H^1 \cup L_{XH}^1$  will be at most  $\frac{1}{2}$ , and all these rectangles must fit in one shelf in  $S_1$ .

Property 2 is true as  $2N \geq H(L_{WD}) > 3(N_1 - 1) + p_1$ , leading to  $N_1 < \frac{2N-1}{3} + 1$ .

We now prove property 3. First notice that  $\lambda > \frac{v}{2}$  implies  $k \geq 2$ . Moreover,  $p_2$  and  $p_3$  are strictly greater than  $2v - h_k$  as  $r_k$  did not fit in  $Lay_{i_1-2}$  and

$Lay_{i_1-1}$ . On one side, we have  $p_1 + p_2 + p_3 > b_1$  with  $b_1 = \lambda + 2(2v - h_k)$ , which is large for small values of  $h_k$ . On the other side (for large values of  $h_k$ ), we have  $p_1 + p_2 + p_3 > b_2$  with  $b_2 = \lambda + (2v - h_k) + 4h_k$  as there is at least four rectangles in  $Lay_{i_1-1}$ , and  $p_1 + p_2 + p_3 > b_3$  with  $b_3 = kh_k + 2(2v - h_k)$  as  $p_1 \geq kh_k$ . Thus,  $p_1 + p_2 + p_3$  is larger than  $\max(b_1, b_2)$  and  $\max(b_1, b_3)$ . Then, we just prove that  $\max(b_1, b_2) \geq \frac{v}{5}$  and  $\max(b_1, b_3) \geq v - 2\frac{\lambda}{k}$ .  $\square$

We now study the different cases, assuming that  $N_1 > 1$ .

**If  $p_1 > \frac{5v}{3}$ .**

Let us first define the  $Merge(X)$  procedure for this case. If  $X \subset L_{XH}$ ,  $Merge$  creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, x_1)$  and  $CreateShelf(X, x_2)$  respectively, where  $x_i = av((2-i)v)$ . Notice that if  $sh_2$  is not empty then  $W(sh_1) + W(sh_2) > x_1$ , as rectangles of  $sh_2$  were not included in  $sh_1$ . Then  $sh_i$  is packed at level  $(2-i)v$ , right justified. If  $X \subset L_H$ ,  $Merge$  creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, x_1)$  and  $CreateShelf(X, x_2)$  respectively, where  $x_i = av((i-1)v)$  (we take into account wide rectangles preallocated in phase 1 and maybe extra high rectangles). Then  $sh_i$  is packed at level  $(i-1)v$ , right justified.

The algorithm makes the following calls:

1.  $Merge(L_{XH}^1)$
2.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
3.  $GreedyPack(L_H^1, seq_H)$
4.  $Add(L_H^1, S_{i_{last}})$
5.  $Merge(L_H^1)$

Let first suppose that two shelves of extra high rectangles are created in step 1. Recall that in this case  $W(sh_1) + W(sh_2) > x_1$ . Thus, after phase 1 we get  $S(\pi_0^{N_1}) > (1-x_1)v + (p_1-v)\frac{1}{3} + \frac{2v}{3}x_1$ , which is a decreasing function of  $x_1$ . With  $x_1 = \frac{2}{3}$  we get  $S(\pi_0^{N_1}) > \frac{5v}{9} + \frac{4v}{9} = v$ . Therefore we conclude that step 2, 3 and 4 must pack the remaining rectangles using Lemma 11.

We now suppose that step 1 created only one shelf, implying that  $L_{XH}^1$  is empty after step 1. Let  $\bar{L}_H^{-1}$  denote the remaining rectangles of  $L_H^1$  just before step 5. If  $\bar{L}_H^{-1}$  is not empty after step 5 the end, we get  $S(\pi_0^i) > v$  for any  $i \in [N_1+1, N]$ . Moreover,  $S(\pi_0^{N_1} \cup \bar{L}_H^{-1}) > (1-x_2)v + (p_1-v)\frac{1}{3} + S(sh_1) + x_2\frac{v}{2}$  which is a decreasing function of  $x_2$ . With  $x_2 = \frac{2}{3}$  we get  $S(\pi_0^{N_1} \cup \bar{L}_H^{-1}) > \frac{5v}{9} + \frac{v}{8} + \frac{v}{3} > v$ , leading to  $S(I) > v$ .

**If  $\frac{5v}{3} \geq p_1 > \frac{4v}{3}$ .**

In this case, we first repack  $Lay_{i_1}$  by calling two times  $CreateLayer(Lay_{i_1}, v)$ . Let  $Lay'_{i_1}$  and  $Lay''_{i_1}$  be respectively the first and the second new layers. We repack these layers at level 0, with  $Lay'_{i_1}$  left justified and  $Lay''_{i_1}$  right justified,

with the narrowest rectangles on the top. As there no huge rectangles remaining, we have  $H(Lay'_{i_1}) > \frac{2v}{3}$ , implying  $H(Lay''_{i_1}) \leq v$ .

In this case, the  $Merge(X)$  procedure creates a shelf  $sh$  using  $CreateShelf(X, x)$  where  $x = av(v)$ , and packs it at level  $v$ , right justified. Notice that when  $X \subset L_H$ ,  $x$  can be lower than 1 as there may be some extra high rectangles at level  $v$ .

The algorithm makes the following calls:

1.  $Merge(L_{XH}^1)$
2.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
3.  $GreedyPack(L_H^1, seq_H)$
4.  $Add(L_H^1, S_{last})$
5.  $Merge(L_H^1)$

According to Lemma 15, we get  $p_1 + p_2 + p_3 > 3v + \frac{4v}{3} + \frac{v}{6}$ . If  $L_{XH}^1$  is not empty after step 1, then  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1}) > (p_1 + p_2 + p_3) \frac{1}{v} + S(sh) > v + \frac{4v}{9} + \frac{v}{18} + \frac{v}{2} > 2v$ . Then step 2, 3 and 4 must pack all the remaining rectangles according to Lemma 11.

If  $L_{XH}^1$  is empty after step 1, then step 3 packs at least a surface  $v$  in each of the last  $N_2$  strips. Then, if  $L_H^1$  is not empty after step 4 we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1} \cup L_H^1 \cup L_{XH}^1) > v + \frac{4v}{9} + \frac{v}{18} + \frac{v}{2} > 2v$ .

**If  $\frac{4v}{3} \geq p_1 > v$ .**

In this case we also repack the wide rectangles, but not in every sub-cases. Let us describe the  $Merge(X)$  procedure in this case (see Figure 7 Page 17 for an example of  $Merge(X)$ ). If  $X \subset L_{XH}$  (and  $X \neq \emptyset$ ),  $Merge(X)$  repacks  $Lay_{i_1}$  by calling two times  $CreateLayer(Lay_{i_1}, v)$ . Let  $Lay'_{i_1}$  and  $Lay''_{i_1}$  be respectively the first and the second new layers.  $Merge$  repacks these layers at level 0, with  $Lay'_{i_1}$  left justified and  $Lay''_{i_1}$  right justified, with the narrowest rectangles on the top. As there no huge rectangles remaining, we have  $H(Lay'_{i_1}) > \frac{2v}{3}$ , implying  $H(Lay''_{i_1}) \leq \frac{2v}{3}$ . Then,  $Merge(X)$  creates a shelf  $sh$  using  $CreateShelf(X, 1)$  and packs it at level  $v$ , left justified.

If  $X \subset L_H$ , two cases are possible as  $Lay_{i_1}$  had maybe been already repacked by a previous call to  $Merge$ . If  $Lay_{i_1}$  is already repacked,  $Merge(X)$  creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, x_1)$  and  $CreateShelf(X, x_2)$  respectively, where  $x_1 = av(\frac{4v}{3})$  and  $x_2 = \text{Min}(x_1, av(\frac{2v}{3}))$ . Then  $sh_i$  is packed right justified at level  $\frac{6-2i}{3}$ . If  $Lay_{i_1}$  is not already repacked (then  $S_{N_1}$  does not contain any extra high rectangle),  $Merge(X)$  creates three shelves  $shi$  (for  $1 \leq i \leq 3$ ) using  $CreateShelf(X, x_i)$ , where  $x_i = av(\frac{(6-2i)v}{3})$ . Then,  $shi$  is packed at level  $\frac{(6-2i)v}{3}$ , right justified. Notice that if  $sh_3$  is not empty then  $W(sh_2) + W(sh_3) > x_2$  as rectangles of  $sh_3$  were not included in  $sh_2$ .

The algorithm makes the following calls:

1.  $last = GreedyPack(L_{XH}^1, seq_{XH})$

2.  $Merge(L_{XH}^1)$
3.  $Merge(L_H^1)$
4.  $GreedyPack(L_H^1, seq_H)$
5.  $Add(L_H^1, S_{i_{last}})$

According to Lemma 15, we have  $p_1 + p_2 + p_3 > 3v + v + (v - \frac{2}{k}) > 4v + \frac{v}{3}$  as  $k > 3$  in this case (remember that we assume that there is no huge rectangles in the last two layers of phase 1).

Let us suppose first that  $L_{XH}^1$  is not empty after step 2. Then after step 1 each of the last  $N_2$  strips is filled with an area strictly larger than  $v$ . Let  $L_{XH}^{\bar{1}}$  denote  $L_{XH}^1$  after step 1 and  $\bar{\pi}_0^{N_1}$  denote the set packed in  $S_{N_1}$  before step 2. We have  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_{XH}^{\bar{1}}) > \frac{4v}{3} + \frac{v}{9} + \frac{2v}{3} > 2v$ , leading to  $S(I) > Nv$ .

If  $L_{XH}^1$  is empty after step 2, two cases are possible according to what is packed in  $S_{N_1}$ . If no extra high rectangle is packed in  $S_{N_1}$ , then  $Lay_{i_1}$  is not repacked by  $Merge$ . After step 3, if  $L_H^1$  is not empty we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1}) > S(sh_1) + (W(sh_2) + W(sh_3))\frac{v}{2} + (p_1 + p_2 + p_3)\frac{1}{3} > \frac{3v}{8} + x_2\frac{v}{2} + (1 - x_2)\frac{2v}{3} + (p_1 - \frac{2v}{3} + p_2 + p_3)\frac{1}{3}$  which is decreasing in  $x_2$ . For  $x_2 = \frac{2}{3}$  we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1}) > 2v$ , and according to Lemma 11 step 4 and 5 must pack all the remaining rectangles.

If a shelf  $sh$  of extra high rectangles is packed in  $S_{N_1}$ , then  $Lay_{i_1}$  is repacked by  $Merge$ . Before step 3 the area packed in the last  $N_2$  strips is already strictly larger than  $v$ . Let us analyze what happen if  $L_H^1$  is not packed after step 3. Let  $\bar{\pi}_0^{N_1}$  denote the set of packed rectangles before step 1. We proceed by case analysis according to  $W(sh)$ . If  $W(sh) \leq \frac{1}{3}$  then  $x_2$  is the available width at level  $\frac{2v}{3}$ , and we get  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup sh \cup L_H^1) > (1 - x_2)\frac{2v}{3} + (p_1 - \frac{2v}{3} + p_2 + p_3)\frac{1}{3} + x_2\frac{v}{2} + \frac{1 - W(sh)}{2}\frac{v}{2} + W(sh)\frac{2v}{3}$  which is a decreasing function of  $x_2$ . For  $x_2 = \frac{2}{3}$  we get  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup sh \cup L_H^1) > \frac{13v}{9} + \frac{v}{3} + \frac{1 - W(sh)}{2}\frac{v}{2} + W(sh)\frac{2v}{3}$ , which is an increasing function of  $W(sh)$ . Replacing  $W(sh)$  by 0 we get  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup sh \cup L_H^1) > 2v$ . If  $W(sh) > \frac{1}{3}$  then we only need to use that  $W(L_H^1) > 1 - W(sh)$ . Thus we have  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup sh \cup L_H^1) > \frac{13v}{9} + W(sh)\frac{2v}{3} + (1 - W(sh))\frac{v}{2}$ , leading for  $W(sh) = \frac{1}{3}$  to  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup sh \cup L_H^1) > 2v$ .

**If  $v \geq p_1 > \frac{2v}{3}$ .**

Let us first describe the  $Merge(X)$  in this case. If  $X \subset L_{XH}^1$ ,  $Merge$  creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, 1)$  and  $CreateShelf(X, x_2)$  where  $x_2 = av(0)$ . Then,  $sh_i$  is packed right justified at level  $(2 - i)v$ . If  $X \subset L_H^1$ ,  $Merge$  creates two shelves  $sh_1$  and  $sh_2$  using  $CreateShelf(X, x_1)$  and  $CreateShelf(X, x_2)$  where  $x_i = av(\frac{v(2i-1)}{3})$ . Then,  $sh_i$  is packed at level  $\frac{v(2i-1)}{3}$ , right justified.

The algorithm makes the following calls:

1.  $Merge(L_{XH}^1)$
2.  $last = GreedyPack(L_{XH}^1, seq_{XH})$

3.  $GreedyPack(L_H^1, seq_H)$
4.  $Add(L_H^1, S_{i_{last}})$
5.  $Merge(L_H^1)$

According to Lemma 15, we have in this case  $p_1 + p_2 + p_3 > 4v$ . If step 1 creates two shelves of extra high rectangles, then (after step 1)  $W(\pi_0^{N_1} \cap L_{XH}^1) > 1$  and we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1}) > \frac{4v}{3} + \frac{2v}{3} > 2v$ . Thus step 2, 3 and 4 must pack the remaining rectangles according to Lemma 11. Otherwise,  $L_{XH}^1$  is completely packed (in one shelf) in  $S_{N_1}$ . Let us assume that  $L_H^1$  is not packed after step 5. The surface packed in the last  $N_2$  strips is strictly larger than  $v$ . Let  $\bar{L}_H^1$  be  $L_H^1$  just before step 5. If  $\bar{L}_H^1$  is not packed by step 5 we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1} \cup \bar{L}_H^1 \cup L_{XH}^1) > (1-x_1)\frac{v}{3} + (p_1 - \frac{v}{3} + p_2 + p_3)\frac{1}{3} + \frac{x_1 v}{2} + \frac{v}{2}$ , which is decreasing in  $x_1$ . Thus, for  $x_1 = \frac{2}{3}$  we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1} \cup \bar{L}_H^1 \cup L_{XH}^1) > \frac{4v}{3} + \frac{v}{6} + \frac{v}{2} = 2v$ .

**If  $\frac{2v}{3} \geq p_1$ .**

If  $k \geq 2$ , then we get  $p_1 + p_2 + p_3 \geq 4v$ , as none of the rectangles of  $Lay_{i_1}$  fit in the previous layers. Thus, the analysis is the same as in the previous case. We now assume that  $k = 1$ , meaning that there is only one wide rectangle  $r_1$  of height  $h_1$  and width  $w_1$ .

Let us describe the  $Merge(X)$  procedure in this case. If  $X \subset L_{XH}$ ,  $Merge(X)$  creates two shelves  $sh_1$  and  $sh_2$ , using  $CreateShelf(X, x_1)$  and  $CreateShelf(X, 1)$  respectively, where  $x_1 = av(0)$ . Then  $sh_i$  is packed at level  $(i-1)v$ , right justified.

If  $X \subset L_H$ ,  $Merge(X)$  creates three shelves  $sh_i$  for  $1 \leq i \leq 3$  using  $CreateShelf(X, x_i)$ , where  $x_i = av(\frac{v(6-2i)}{3})$ . Then  $sh_i$  is packed at level  $\frac{v(6-2i)}{3}$ , with  $sh_1$  and  $sh_2$  left justified, and  $sh_3$  right justified.

The algorithm makes the following calls:

1.  $last = GreedyPack(L_{XH}^1, seq_{XH})$
2.  $Merge(L_{XH}^1)$
3.  $Merge(L_H^1)$
4.  $GreedyPack(L_H^1, seq_H)$
5.  $Add(L_H^1, S_{i_{last}})$

Let us first bound  $S(Lay_{i_1-2} \cup Lay_{i_1-1} \cup r_1)$ . As  $r_1$  did not fit in  $Lay_{i_1-2}$  and  $Lay_{i_1-1}$ ,  $p_2$  and  $p_3$  are greater than  $2v - h_1$ . Moreover, as no rectangle of  $L_B$  is in  $Lay_{i_1-1}$  we get  $p_2 > 4h_1$ . Thus  $S(Lay_{i_1-2} \cup Lay_{i_1-1} \cup r_1) > \max(\frac{2(2v-h_1)}{3} + w_1 h_1, \frac{2v-h_1}{3} + \frac{4h_1}{3} + w_1 h_1)$ . Notice that the left part of the max is decreasing in  $h_1$  as  $w_1 \leq \frac{1}{2}$ . Thus, replacing  $h_1$  by  $\frac{2v}{5}$  we get  $S(Lay_{i_1-2} \cup Lay_{i_1-1} \cup r_1) > \frac{16v}{15} + \frac{2w_1 v}{5}$ .

Let us suppose first that  $L_{XH}^1$  is not empty after step 2. Then after step 1 each of the last  $N_2$  strips is filled with an area strictly larger than  $v$ . Let  $L_{XH}^{-1}$  denote  $L_{XH}^1$  after step 1 and  $\bar{\pi}_0^{N_1}$  denote the set packed in  $S_{N_1}$  before step 2. We have  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_{XH}^{-1}) > \frac{16v}{15} + \frac{2w_1v}{5} + \frac{1-w_1}{2} \frac{2v}{3} + \frac{2v}{3}$  which is increasing in  $w_1$ . For  $w_1 = \frac{1}{3}$  we get  $S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_{XH}^{-1}) > 2v$ , leading to  $S(I') > Nv$ .

If  $L_{XH}^1$  is empty after step 2, two cases are possible according to what is packed in  $S_{N_1}$ . Let us first assume that no extra high rectangle is packed in  $S_{N_1}$ . After step 3, if  $L_H^1$  is not empty we get  $S(\pi_0^{N_1} \cup \pi_0^{N_1-1}) > \frac{16v}{15} + \frac{2w_1v}{5} + \frac{6v}{8} + \frac{1-w_1}{2} \frac{v}{2}$  which is greater than  $2v$  for  $w_1 = \frac{1}{3}$ . Then, step 4 and 5 must pack all the remaining rectangles according to Lemma 11.

We now assume that some extra high rectangles are packed in  $S_{N_1}$ . In this case, the area packed before step 3 in the last  $N_2$  strips is already strictly larger than  $v$ . Let  $\bar{\pi}_0^{N_1}$  denote the set of packed rectangles before step 1. We proceed by contradiction by supposing that  $L_H^1$  is not packed after step 3.

If only one shelf  $sh_1$  of extra high rectangles is packed in  $S_{N_1}$ , we get

$$\begin{aligned} S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_H^1 \cup sh_1) &> \frac{16v}{15} + \frac{2w_1v}{5} + \frac{3v}{8} + (1 - W(sh_1)) \frac{v}{2} \\ &\quad + W(sh_1) \frac{2v}{3} \\ &> \frac{16v}{15} + \frac{2w_1v}{5} + \frac{7v}{8} \\ &> 2v \end{aligned}$$

If two shelves  $sh_1$  and  $sh_2$  of extra high rectangles are packed in  $S_{N_1}$ , two cases are possible according to  $W(sh_2)$ . If  $W(sh_2) > \frac{2}{3}$ , we get

$$\begin{aligned} S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_H^1 \cup sh_2 \cup sh_1) &> \frac{16v}{15} + \frac{2w_1v}{5} + (1 - W(sh_2)) \frac{v}{2} \\ &\quad + W(sh_2) \frac{2v}{3} + \frac{1-w_1}{2} \frac{2v}{3} \\ &> \frac{16v}{15} + \frac{2w_1v}{5} + \frac{v}{6} + \frac{4v}{9} + (1 - w_1) \frac{v}{3} \\ &> 2v \end{aligned}$$

If  $W(sh_2) \leq \frac{2}{3}$ , we get

$$\begin{aligned} S(\bar{\pi}_0^{N_1} \cup \pi_0^{N_1-1} \cup L_H^1 \cup sh_2 \cup sh_1) &> \frac{16v}{15} + \frac{2w_1v}{5} + \frac{3}{2}(1 - W(sh_2)) \frac{v}{2} \\ &\quad + W(sh_2) \frac{2v}{3} + \frac{1-w_1}{2} \frac{2v}{3} \\ &> \frac{16v}{15} + \frac{2w_1v}{5} + \frac{v}{4} + \frac{4v}{9} + (1 - w_1) \frac{v}{3} \\ &> 2v \end{aligned}$$

Thus, it is possible to pack the remaining rectangle for all possible cases that may occur after phase 1, and thus Lemma 14 is proved.

### 3.6 Complexity

According to the main steps defined in Section 2.2, the algorithm that preallocates  $I'$  is sufficient to get a 2-approximation. Indeed, we simply add rectangles of  $I \setminus I'$  using list algorithms defined in Section 2.2.

Let us now sketch the analysis of the running time of the preallocation algorithm. Roughly speaking, Phase 1 runs in  $\mathcal{O}(n \log(n) + Nn)$  as for each strip creating a layer can be done in  $\mathcal{O}(n)$  (by initially sorting wide rectangles according to their heights). Phase 2 also runs in  $\mathcal{O}(n \log(n) + Nn)$  as for each strip creating a constant number of shelves (generally two or three) can be done in  $\mathcal{O}(n)$  (by initially sorting high and extra high rectangles according to their widths).

Let us now bound the running time of algorithms of Section 2.2.2 that turn  $\pi_0$  into the final packing. Due to the simple structure of preallocation, the  $LS_{\pi_0}$  algorithm can be implemented in  $\mathcal{O}(n \log(n))$ . Indeed, instead of scanning level by level and strip by strip, this algorithm can be implemented by maintaining a list that contains the set of “currently” packed rectangles. The list contains 3-tuples  $(j, l, i)$  indicating that the top of rectangle  $r_j$  (packed on strip  $S_i$ ) is at level  $l$ . Thus, instead of scanning every level from 0 it is sufficient to maintain sorted this list according to the  $l$  values (in non decreasing order), and to only consider at every step the first element of the list (*i.e.* the first job that completes). Then, it takes  $\mathcal{O}(\log(n))$  to find a rectangle  $r_{j_0}$  in the appropriate shelf that fits at level  $l$ , because a shelf can be created as a sorted array. It also takes  $\mathcal{O}(\log(n))$  to insert the new event corresponding to the end of  $r_{j_0}$  in the list.

The last step, which turns  $\pi_1$  into the final packing can also be implemented in  $\mathcal{O}(n \log(n))$  using a similar global list of events. Notice that for any strip  $S_i$ , there exists a  $l_i$  such that below  $l_i$  the utilization is an arbitrary function strictly larger than  $1/2$ , and a non increasing after. Packing a small rectangle before  $l_i$  would require additional data structure to handle the complex shape. Thus we do not pack any small rectangle before  $l_i$  as it is not necessary for achieving the 2 ratio. Therefore, we only add those events that happen after  $l_i$  when initializing the global list for this step. To summarize, for this step we only need to sort the small rectangles in non increasing order of their required number of processors, and then apply the same global list algorithm.

Thus, taking into account the repetitions due to the binary search on  $v$ , this algorithm can be implemented in  $\mathcal{O}(\log_2(nh_{max})n(N + \log(n)))$ .

## References

- [BDJ<sup>+</sup>09] M. Bougeret, P-F. Dutot, K. Jansen, C. Otte, and D. Trystram. Approximation algorithm for multiple strip packing. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA)*, 2009.

- [BDJ<sup>+</sup>10] M. Bougeret, P-F. Dutot, K. Jansen, C. Otte, and D. Trystram. Approximating the non-contiguous multiple organization packing problem. In *Proceedings of the 6th IFIP International Conference on Theoretical Computer Science (TCS)*, 2010.
- [HS87] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.
- [HS88] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.
- [Jan09] K. Jansen. An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables. *Automata, Languages and Programming*, pages 562–573, 2009.
- [STY08] U. Schwiegelshohn, A. Tcherykh, and R. Yahyapour. Online scheduling in grids. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–10, 2008.
- [YHZ09] D. Ye, X. Han, and G. Zhang. On-Line Multiple-Strip Packing. In *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications (COCOA)*, page 165. Springer, 2009.
- [Zhu06] SN Zhuk. Approximate algorithms to pack rectangles into several strips. *Discrete Mathematics and Applications*, 16(1):73–85, 2006.