Improved approximation algorithms for scheduling parallel jobs on identical clusters

Marin Bougeret

LIRMM, Université Montpellier 2, 34095 Montpellier, France

Pierre-Francois Dutot, Denis Trystram

LIG, Laboratoire d'Informatique de Grenoble, 38334 Saint Ismier Cedex, France

Klaus Jansen, Christina Robenek

Department of Computer Science, Universitt Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, Germany

Abstract

The Multiple Cluster Scheduling Problem corresponds to minimize the maximum completion time (makespan) of a set of n parallel rigid (and non-preemptive) jobs submitted to N identical clusters. It cannot be approximated with a ratio better than 2 (unless $\mathcal{P} = \mathcal{NP}$). We present in this paper the methodology that encompasses several existing results [1, 2]. We detail first how to apply it for obtaining a $\frac{5}{2}$-approximation. Then, we use it to provide a new $\frac{7}{3}$-approximation running in $\mathcal{O}(\log\,(nh_{max})N(n + \log(n)))$, where h_{max} is the processing time of the longest job. Finally, we apply it to a restriction of the problem to jobs of limited size, leading to a 2-approximation which is the best possible ratio since the restriction remains 2-inapproximable.

Keywords: Scheduling, Parallel job, Strip packing, Approximation algorithm

1. Introduction

1.1. Problem statement

In the grid computing paradigm, several clusters share their computing resources in order to better distribute the workload. Each cluster is composed of a set of identical processors connected by a fast local interconnection network [3]. Jobs are submitted over time in successive packets called batches. The objective is to minimize the time when all the jobs of a batch are completed, thus, minimizing the date when the next batch of jobs can be processed. Many such computational grid systems are available all over the world, and the efficient management of the resources is known to be one of the most important challenge today. Let us start by stating the corresponding Multiple Cluster Scheduling Problem (MCSP) more formally.

Definition 1 (MCSP)**.** *We are given n parallel rigid non-preemptive 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). The 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*

This problem is closely related to the following Multiple Strip Packing problem (MSPP).

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 under the following constraints.*

1. *a rectangle must be entirely packed into a strip (it cannot be split between two strips)*
2. *at any level of any strips, the total width of packed rectangles 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 term of job scheduling amounts to force the jobs to use consecutive indexes of processors (see Figure 1). Of course, the results for MCSP generally do not apply to MSPP because of the additional contiguous constraints. The converse is also not clear, since the approximation ratios for MSPP may not be preserved when considering MCSP. 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. In this paper, the studied problem (MCSP) is seen as MSPP without constraint 3), and from now on we use the vocabulary and notations of packing.

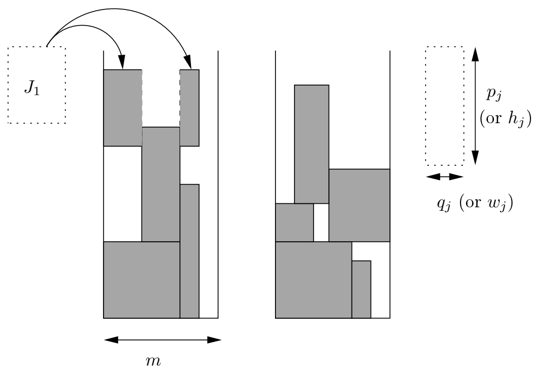


Figure 1: Example (for $n = 9$ jobs and $N = 2$ clusters) of a solution that is feasible for MCSP and not feasible for MSPP. Notice there hat J_1 is packed in a "non-continuous" way (using non consecutive indexes of processors).

1.2. Related Work

As shown in [4] using a gap reduction from the 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. For the sake of readability, we call "fast algorithm" algorithms with a running time in $\mathcal{O}(n^p)$, with $p \leq 3$ (the exact complexity of these algorithms is not relevant here).

We must distinguish the 3-approximation in [7] and the $\frac{5}{2}$-approximation in [1] that have a low cost from the costly 2-approximation in [6] and $2 + \epsilon$-approximation in [5].

Problem	Ratio	Remarks	Source
MCSP, MSPP	$2+\epsilon$	Need solving $P\|\|C_{max}$ with a ratio $1+\frac{\epsilon}{2}$	[5]
MCSP	5/2	Fast algorithm	[1]
MSPP	2	Costly algorithm (at least $\Omega(n^{256})$)	[6]
MCSP, MSPP	AFPTAS	Additive constant in $\mathcal{O}(\frac{1}{\epsilon^2})$, and in $\mathcal{O}(1)$ for large values of N	[6]
MCSP	3	Fast (and decentralized) algorithm that handles clusters of different sizes	[7]
MCSP	7/3	Fast algorithm	This paper
MCSP	2	Requires $\boldsymbol{\max_j w_j \leq \frac{1}{2}}$ Fast algorithm	This paper, from [2]

Figure 2: Summary of existing results.

The 2-approximation does not apply to $MCSP$. Moreover, it is directly obtained from asymptotic approximation algorithms when the number of strips is larger than a constant N_0, but requires algorithms that are exponential in N_0 when the number of strips is lower than N_0. Thus, the value of this constant ($\approx 10^4$) makes this algorithm impossible to use for real size instances.

The $2+\epsilon$-approximation applies to $MCSP$, but requires to solve the famous $P||C_{max}$ problem (which is makespan minimization when scheduling sequential jobs on identical machines) with a ratio $1+\frac{\epsilon}{2}$ [8]. As all the well-known "fast" approximation algorithms for $P||C_{max}$ (like Longest Processing Time First or Multifit [9]) have a ratio ρ such that $2\rho > \frac{7}{3}$, using this black box technique to get a $\frac{7}{3}$ ratio for $MCSP$ requires using a PTAS for $P||C_{max}$ with $\epsilon' = \frac{1}{6}$. Even if some recent advances in the PTAS design for $P||C_{max}$ allowed to decrease the asymptotic dependencies in $\frac{1}{\epsilon}$ (like $2^{\mathcal{O}(\frac{1}{\epsilon^2}\log^3(\frac{1}{\epsilon}))}$ in [10]), the running time of these new algorithms remains very large due to the hidden constants.

Lastly, in [11] we extended MCSP for clusters of different sizes and provide a polynomial time algorithm with ratio $(2+\epsilon)$. Even if the complexity is polynomial, the running time algorithm is not practical and does not compete with the results presented here.

1.3. Motivations and contributions

In this paper, we recall the methodology that formalizes the common principles used in [1, 2], and we apply it to get a new $\frac{7}{3}$-approximation for MCSP running in $\mathcal{O}(\log(nh_{max})N(n+\log(n)))$, which is clearly faster than the previous mentioned algorithms.

The underlying principle consists in 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 (*i.e.* with a set I' containing only really huge rectangles) of the technique. As we believe that the discarding framework of Section 2.2 is well suited for MCSP, the objective is to apply it using a more "challenging" set I'. Thus, we present in Section 3 a new $\frac{7}{3}$-approximation, and we sketch in Section 4 the 2-approximation in [2] for a restriction of MCSP. Both algorithms run in $\mathcal{O}(\log(nh_{max})N(n+\log(n)))$, and thus are very fast.

Of course, it could possible to improve again the $\frac{7}{3}$ ratio using the same method (targeting for example a $\frac{9}{4}$ ratio). However, we think this could be a tough work as the relative performance improvement is getting smaller, and of course the difficulty of proofs (typically the number of particular cases to handle according to how many rectangles of each "type" remains) is likely to increase. Thus, we concluded our study by looking at a reasonable restriction of MCSP where the bounds could be tightened. In this spirit, the result in [2] sketched in Section 4 holds on a restriction of MCSP where the width of all rectangles are lower than $\frac{1}{2}$ (*i.e.* jobs submitted to clusters do not require strictly more than the half of the resources). Using the same framework, it leads to a 2-approximation. As this restriction remains 2-inapproximable unless $\mathcal{P} = \mathcal{NP}$, this result is optimal in the sense of approximation theory.

2. General principles

In this section, we give some definitions and we provide a general methodology which serves as a basis for the design of efficient algorithms, and in particular the new $\frac{7}{3}$-approximation of Section 3 and the restricted 2-approximation of Section 4.

2.1. Preliminaries

Recall that our objective is to (non-contiguously) pack 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.

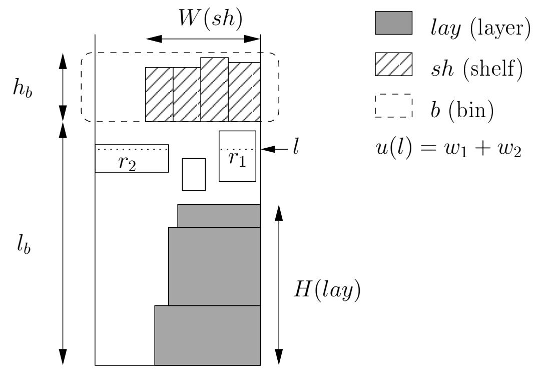


Figure 3: Example of a layer, a shelf, a bin and of the utilization function. sh is packed in b.

The *utilization* $u_i^\pi(l)$ of a packing π 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. By definition, 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 the 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. Again, by definition the shelf produced by the procedure is such that $W(sh) \leq w$. Throughout the paper, we consider that the procedures modify the sets of jobs given as parameters.

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

Lemma 1. *Let sh denote the shelf created by $CreateShelf(X, w)$. If the k widest rectangles of X are added to sh and there is at least one rectangle in X which cannot be added, 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 $a = \Sigma_{i=1}^{i_0-1} w_i$. We have $W(sh) \geq a \geq (i_0 - 1)w_{i_0} > (i_0 - 1)(w - a)$ leading to $a > \frac{i_0-1}{i_0} w \geq \frac{k}{k+1} w$. □

Finally, let us recall two useful results that we will use to claim that a set *select* of rectangles (each of height at most v) of total surface at most $\frac{7v}{6}$ can be packed in one strip with a height at most $\frac{7v}{3}$.

Theorem 1 ([12])**.** *Let $L = \{r_1, \ldots, r_n\}$ be a set of rectangles. Let $w_{max} = max_j w_j$ and $h_{max} = max_j h_j$. If $w_{max} \leq u, h_{max} \leq v$ and*

$$2S(L) \leq uv - \max(2w_{max} - u, 0)\max(2h_{max} - v, 0)$$

then it is possible to pack L (in time $\mathcal{O}(n\log^2(n)/\log(\log(n)))$) in a rectangular box of width u and height v.

Notice also that in our particular case of non-contiguous packing, we can simply use the Widest First algorithm (which runs in $\mathcal{O}(n\log(n))$) that scans the strips upward from level 0 and packs the widest possible remaining rectangle for every level. Indeed, let us recall the following simple lemma (proved in [13])

Lemma 2. *Let X be a set of rectangles, $\lambda \geq 1$ and v such that*

- $S(X) \leq \lambda v$
- *for all $j \in X$, $h_j \leq v$.*

Then, the Widest first algorithm packs X in a strip with a height lower than $2\lambda v$.

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 problems. 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 change 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 in the context of non-contiguous multiple strip packing. Given an instance I, the set of big rectangles $I'(\alpha,\beta) \subset I$ depends on parameters (α and β 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 use the well-known dual approximation technique [14], and we denote by v the guess of the optimal value. Given an instance I, let $L_{WD} = \{r_j | w_j > \alpha\}$ be the set of wide rectangles, $L_H = \{r_j | h_j > \beta v\}$ 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 target approximation ratio (the origin of this formula will be explained in Section 2.2.2).

Following the dual approximation technique, we will prove that either I is packed with a resulting height lower than $r(\alpha, \beta)v$, or $v < Opt$. Notice that for the sake of simplicity we did not add the "reject" instructions in the algorithms. Thus we consider in all the proofs 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.

We also need the following definition.

Definition 3. *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.*

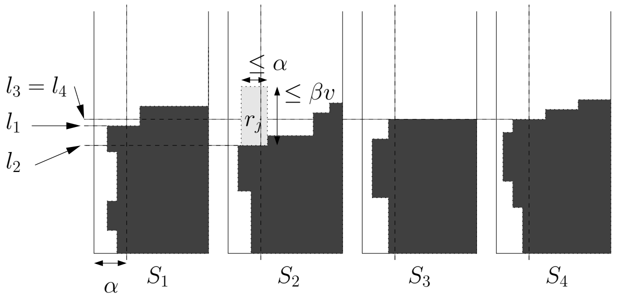


Figure 4: Example showing a $(1 - \alpha)$ compact packing, and why step c) is simple. Indeed, adding as soon as possible a small rectangle r_j (with $h_j \leq \beta v$ and $w_j \leq \alpha$) to a $(1-\alpha)$ compact packing cannot exceed $v(\frac{1}{1-\alpha} + \beta)$. The l_i values are defined according to Definition 3.

Let us now describe the three main steps of our approach. Notice that what we call a preallocation is a "normal" packing (*i.e.* consists in defining the bottom level of each rectangle, which is sufficient to fully describe the solution) 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 π^0 of I' such that

 – (prop. (1)) π^0 fits in $r(\alpha, \beta)v$

  - (prop. (2)) wide rectangles of π^0 ($\pi^0 \cap L_{WD}$) are already packed in a $(1-\alpha)$-compact way (in our case, all of these items are packed in a layer together starting at the bottom of the strip, as depicted Figure 3 for example).

b) turn π^0 into a $(1-\alpha)$-compact packing π^1 (by keeping the large rectangles as they were preallocated, and repacking greedily rectangles of $I' \setminus L_{WD}$)

c) add the small remaining rectangles ($I \setminus I'$) using LS (see Lemma 4)

Step a) is the most difficult one. Indeed, building the preallocation becomes harder when α and β are small, as the number of rectangles of I' increases and $r(\alpha, \beta)$ decreases. Thus, Sections 3 and 4 are entirely devoted to the construction of π^0 (for (α, β) equal to $(\frac{1}{2}, \frac{1}{3})$ and $(\frac{1}{3}, \frac{1}{2})$, respectively).

Let us now see how steps b) and c) lead to a $r(\alpha, \beta)$ ratio.

2.2.2. Steps b) and c)

In this section, we suppose that we are given a guess v, and a preallocation π^0 of $I' = L_{WD} \cup L_H$ that satisfies property (1) and (2). Let us consider step b): how to turn π^0 into a $(1-\alpha)$-compact packing.

Lemma 3 (Step b)). *Let π^0 be the preallocation of I' constructed in Step a) that satisfies hypothesis* (1) *and* (2)*. Let $\widehat{\pi}^1 = \pi^0 \cap L_{WD}$ denote π^0 when keeping only rectangles of L_{WD} ($\widehat{\pi}^1$ is already a $(1-\alpha)$-compact packing).*

Then, we can complete $\widehat{\pi}^1$ into a $(1-\alpha)$-compact packing π^1 of I', such that the height of π^1 is lower or equal to the height of π^0.

Proof. Let us define the LS_{π^0} algorithm that packs rectangles of $I' \setminus L_{WD}$. Let us consider a single strip S_i. Let π^0_i denote π^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 should be added to S_i. We assume that $lvl(j) \leq lvl(j+1)$, where $lvl(j)$ is the bottom level of r_j in π^0.

For the considered strip S_i, the LS_{π^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 $\widehat{\pi}^1$ (in S_i) at the smallest possible level. Notice first that adding a rectangle r_j with *AddAsap* (with $w_j \leq \alpha$) to a $(1-\alpha)$-compact packing creates another $(1-\alpha)$-compact packing. Thus it is clear that π^1 is $(1-\alpha)$-compact.

For any $1 \leq j \leq p$, let $(\widehat{\pi}_i^1, j)$ denote the packing in S_i just before adding r_j with *AddAsap*, and let (π_i^0, j) denote the packing $\pi_i^0 \cap (L_{WD} \cup \{r_1, \ldots, r_{j-1}\})$. Let us prove by induction on $j \in \{1, \ldots, p\}$ that $u^{(\widehat{\pi}_i^1, j)}(l) \leq u^{(\pi_i^0, j)}(l)$, for any $l \geq lvl(j)$. The definition of $\widehat{\pi}^1$ 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 $(\widehat{\pi}_i^1, j+1)$, then it also occurs in $(\pi_i^0, j+1)$. Thus in this case we have

$$\begin{aligned} u^{(\widehat{\pi}_i^1, j+1)}(l) &= u^{(\widehat{\pi}_i^1, j)}(l) + w_j \\ &\leq u^{(\pi_i^0, j)}(l) + w_j \\ &= u^{(\pi_i^0, j+1)}(l) \end{aligned}$$

If r_j does not intersect l in $(\widehat{\pi}_i^1, j)$, then clearly $u^{(\widehat{\pi}_i^1, j+1)}(l) = u^{(\widehat{\pi}_i^1, j)}(l) \leq u^{(\pi_i^0, j)}(l) \leq u^{(\pi_i^0, j+1)}(l)$.

Thus we proved that for any $1 \leq j \leq p$ we have $u^{(\widehat{\pi}_i^1, j)}(l) \leq u^{(\pi_i^0, j)}(l)$ 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 π^1 is lower or equal to the height of π^0.

□

We now prove in Lemma 4 that after adding rectangles in step c), the height of the packing does not exceed $r(\alpha, \beta)v = (\frac{1}{1-\alpha} + \beta)v$. This explains why the height of the preallocation should also be bounded by $r(\alpha, \beta)v$.

Lemma 4 (Step c)). *Let π^1 be a $(1-\alpha)$-compact packing of I'. Adding to π^1 rectangles of $I \setminus I'$ with a List Scheduling algorithm (LS) leads to a packing π whose height is lower than* $\max(height(\pi^1), v(\frac{1}{1-\alpha} + \beta))$.

Proof. The *LS* algorithm scans all the strips from level 0, and at any level adds any rectangle of $I \backslash I'$ that fits. Notice that the final packing π 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 π is due to a rectangle $r_j \in I \setminus I'$, whose bottom is 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 3). 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 $s < v\frac{1}{1-\alpha}$. Therefore, the height of π is bounded by $s + max_{j \in I \backslash I'} h_j \leq v(\frac{1}{1-\alpha} + \beta)$. $\square$

We give in the next section an example on how to apply this framework to obtain a $5/2$-approximation (using $(\alpha, \beta) = (\frac{1}{2}, \frac{1}{2})$). In section 3 we will apply this framework with $(\alpha, \beta) = (\frac{1}{2}, \frac{1}{3})$ to get a new $\frac{7}{3}$-approximation. Finally, we give in section 4 the sketch of a 2-approximation (for a special case of the problem where $w_j \leq \frac{1}{2}$ for all j), with $(\alpha, \beta) = (\frac{1}{3}, \frac{1}{2})$.

Remark 1. *Notice that for the sake of clarity we will only focus on property (1): build a preallocation that fits in $r(\alpha, \beta)v$. It is sufficient to notice that the simple structure used for L_{WD} will directly implies property (2). Indeed, L_{WD} will be preallocated using only*

- *one layer (for the $7/3$-approximation where $L_{WD} = \{r_j | w_j > 1/2\}$) or two layers "in parallel" (for the 2-approximation where $L_{WD} = \{r_j | 1/2 \geq w_j > 1/3\}$, see Figure 8 Page 27)*
- *layers starting at level* 0
- *layers having narrowest rectangles on the top*

Thus, packing rectangles of L_{WD} as they were preallocated is $1/3$ (or $1/2$) compact.

2.3. Example for a $5/2$-approximation

Let us show how to build a preallocation π^0 for $(\alpha, \beta) = (\frac{1}{2}, \frac{1}{2})$ that fits in $r(\alpha, \beta)v = \frac{5}{2}v$ (again, we skip property (2) as it will be obvious that rectangles of

L_{WD} are preallocated in a $^1\!/\!_2$-compact way). Recall that according to Lemma 3 and 4 this is sufficient to get a $^5\!/\!_2$-approximation. We only provide here a sketch of the construction of π^0, and refer the reader to [1] for more details.

Let us consider the following partition

- let $L_{WD} = \{r_j | w_j > {}^1\!/\!_2\}$ be the set of wide rectangles
- let $L_{XH} = \{r_j | h_j > {}^{3v}\!/\!_4\}$ be the set of extra high rectangles
- let $L_H = \{r_j | {}^{3v}\!/\!_4 \geq h_j > {}^v\!/\!_2\}$ be the set of high rectangles
- let $L_B = L_{WD} \cap (L_{XH} \cup L_H)$ be the set of huge rectangles
- let $I' = L_{WD} \cup L_{XH} \cup L_H$ be the set of big rectangles that we have to preallocate

We start creating π^0 by packing L_{WD}. We create one layer lay_i of rectangles of L_{WD} per strip, until L_{WD} gets empty (let us say using strip 1 to i_1). Each layer is constituted by adding first one huge rectangle, and then rectangles of $L_{WD} \setminus L_B$ until $H(lay_i) \geq 2v$. Thus, if we do not run out of rectangles (this particular case is treated in [1]) all the layers except the last one have a surface greater than $2v \times \frac{1}{2} = v$ and fit in $\frac{5}{2}v$ (as the height of rectangles of $L_{WD} \setminus L_B$ is lower than $\frac{v}{2}$) .

Let us now pack the remaining rectangles of $L_{XH} \cup L_H$ by creating two kind of shelves. Notice first that

- an "extra high" shelf sh created by $CreateShelf(L_{XH}, 1)$ has a width of at least $\frac{2}{3}$ (as L_{WD} is empty we know that at least two rectangles fit in the shelf, and we apply Lemma 1), and thus a surface greater than $\frac{3}{4}v \times \frac{2}{3} = \frac{v}{2}$
- a "high" shelf sh_H created by $CreateShelf(L_H, 1)$ has a width of at least $\frac{2}{3}$ (for the same reason) and thus a surface greater than $\frac{1}{2}v \times \frac{2}{3} = \frac{v}{3}$

Thus, we fill empty strips (from N to i_1) using either two extra high shelves or three high ones. In both cases the total surface packed in each strip is greater than v, and the packing fits in $\frac{5}{2}v$. Notice that details about how packing the last shelves in strip i_1 and how mixing the two kind of shelves (when we run out of rectangles of L_{XH}) are skipped here, but can be treated quite simply.

The previous analysis provides the main arguments to pack an optimal area in a strip using layers or shelves, and thus build the preallocation to get the following theorem.

Theorem 2 ([1]). *There is a $\frac{5}{2}$-approximation for MCSP running in* $\mathcal{O}(\log(nh_{max})N(n+\log(n)))$

Let us now improve this result using a finer decomposition.

3. $\frac{7}{3}$-approximation algorithm

3.1. Definition of the considered partition

As presented in Section 2.2, we define several sets of rectangles. We have to define the set of high rectangles more precisely than $\{r_j | h_j > {}^{v}\!/_{3}\}$ as we will treat differently "extra high" rectangles having height larger than ${}^{2v}\!/_{3}$ and "medium" rectangles having height between ${}^{v}\!/_{3}$ and ${}^{v}\!/_{2}$.

- let $L_{WD} = \{r_j | w_j > {}^{1}\!/_{2}\}$ 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 $L_M = \{r_j | {}^{v}\!/_{2} \geq h_j > {}^{v}\!/_{3}\}$ be the set of medium rectangles
- let $L_B = L_{WD} \cap (L_{XH} \cup L_H \cup L_M)$ be the set of huge rectangles
- let $I' = L_{WD} \cup L_{XH} \cup L_H \cup L_M$
- let $\bar{L}_{XH} = L_{XH} \setminus L_{WD}$, and let $\bar{L}_H$ and $\bar{L}_M$ be defined in the same way

According to our framework, it is sufficient to provide a $\frac{1}{2}$-compact preallocation for rectangles of I'. We recall that according to the dual approximation technique, we will prove that either I is packed with a resulting height lower than ${}^{7v}\!/_{3}$, or $v < Opt$.

3.2. Counting the width of packed rectangles

We start by giving a bound on the total width of extra high, high and medium rectangles.

Lemma 5. *If $v \geq Opt$, then*

- $W(L_{XH}) + W(L_H) \leq N$
- $2W(L_{XH}) + W(L_H) + W(L_M) \leq 2N.$

Proof. Let us suppose that I can be packed in v. Let us consider an arbitrary packing of height at most v in a fixed strip S. We define abscissa x (with $x \in [0, 1]$) as the infinite vertical slice of the strip located at x, considering that the left part of the strip is at abscissa 0 and the right part at abscissa 1. Using the scheduling vocabulary, a fixed abscissa corresponds to a fixed processor x.

Let k_{XH} be the number of rectangles of L_{XH} packed in S that cut abscissa x. Again, in scheduling vocabulary k_{XH} would be the number of jobs of L_{XH} that are processed by processor x. We define k_H and k_M in the same way. Let h be the total height of rectangles packed in S_i at abscissa x. By our assumption we have $h \leq v$. Moreover, we have $h > k_{XH}\frac{2v}{3} + k_H\frac{v}{2} + k_M\frac{v}{3}$. This implies $4k_{XH} + 3k_H + 2k_M < 6$. From this we deduce $2k_{XH} + k_H + k_M \leq 2k_{XH} + \frac{3}{2}k_H + k_M < 3$ and $k_{XH} + k_H \leq \frac{4}{3}k_{XH} + k_H + \frac{2}{3}k_M < 2$. As the leftmost member of each inequality is an integer, we get $2k_{XH} + k_H + k_M \leq 2$, and $k_{XH} + k_H \leq 1$. Then, by summing over all the abscissas and strips we get the desired result. $\square$

We will sometimes use the bounds of Lemma 5 to prove that I' must be packed by counting the total width of extra high, high and medium rectangles packed by the algorithm. Given a packing π_i (of strip S_i), let us define functions $f^?$ such that $f^{XH}(\pi_i) = W(\pi_i \cap L_{XH})$, $f^H(\pi_i) = W(\pi_i \cap L_H)$ and $f^M(\pi_i) = W(\pi_i \cap L_M)$. We can now define the notion of dominating packing.

Definition 4. *A packing in one strip (or simply a set) π_i is dominating iff $2f^{XH}(\pi_i) + f^H(\pi_i) + f^M(\pi_i) > 2$. A packing $\pi = (\pi_1, \ldots, \pi_x)$ of x strips is dominating iff all the π_i are dominating.*

Remark 2. *This notion of domination should not be confused with area domination. A packing π_i is said area-dominating iff $S(\pi_i) > v$.*

We now state a Lemma that emphasizes why dominating packings are interesting.

Lemma 6. *Let π be a dominating packing of I' in x strips. Then $x < N$.*

Proof. We have $2N \geq 2W(L_{XH}) + W(L_H) + W(L_M) = 2\sum_{i=1}^{x} W(L_{XH} \cap \pi_i) + \sum_{i=1}^{x} W(L_H \cap \pi_i) + \sum_{i=1}^{x} W(L_M \cap \pi_i) = \sum_{i=1}^{x}(2f^{XH}(\pi_i) + f^H(\pi_i) + f^M(\pi_i)) > 2x$. $\square$

Finally, let us show how to create a dominating packing in an empty strip.

Lemma 7. *If we do not run out of rectangles, it is always possible to create a dominating packing (which fits in $\frac{7}{3}v$) in an empty strip using rectangles of $I' \setminus L_{WD}$.*

Proof. Let us consider a fixed strip. If $\bar{L}_{XH}$ is such that $W(\bar{L}_{XH}) > 1$, then we just create two shelves sh_1 and sh_2 (such that $W(sh_1) + W(sh_2) > 1$) using rectangles of $\bar{L}_{XH}$, that we pack at level 0 and v. This packing is dominating.

Otherwise (see Figure 5), we pack $\bar{L}_{XH}$ in one shelf sh_1 at level 0. Then, we complete shelf sh_1 by adding greedily (in any order) remaining rectangles of $I' \setminus L_{WD}$ at level 0, until a rectangle (say r_1) does not fit. We pack r_1 at level v. Notice that we have $W(sh_1) + w_1 > 1$. As $r_1 \notin \bar{L}_{XH}$ we know that the top of r_1 is at level at most $\frac{5}{3}v$.

Then, we create a shelf sh_2 by adding greedily (in any order) remaining rectangles of $I' \setminus L_{WD}$ (that belong to $\bar{L}_H \cup \bar{L}_M$) at level $\frac{5v}{3}$, until a rectangle (say r_2) does not fit. Notice that the top of shelf sh_2 does not exceed $\frac{7}{3}v$. Finally, r_2 is packed at level v (r_1 and r_2 both fit at level v as we consider rectangles of $I' \setminus L_{WD}$). Notice that we have $W(sh_2) + w_2 > 1$.

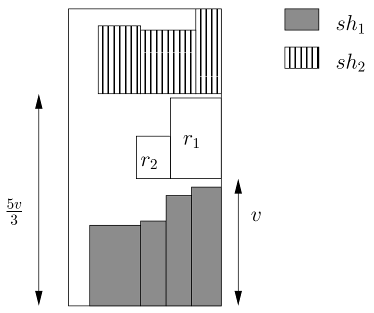


Figure 5: Example of a dominating packing built in Lemma 7.

The packing is dominating since

$$\begin{aligned}2f^{XH}(\pi_i) + f^{H}(\pi_i) + f^{M}(\pi_i) &\geq 2W(\bar{L}_{XH}) + \big(W(sh_1) - W(\bar{L}_{XH})\big) + w_1 \\ &\quad + W(sh_2) + w_2 \\ &> 2 + W(\bar{L}_{XH})\end{aligned}$$

□

3.3. *Description of the algorithm*

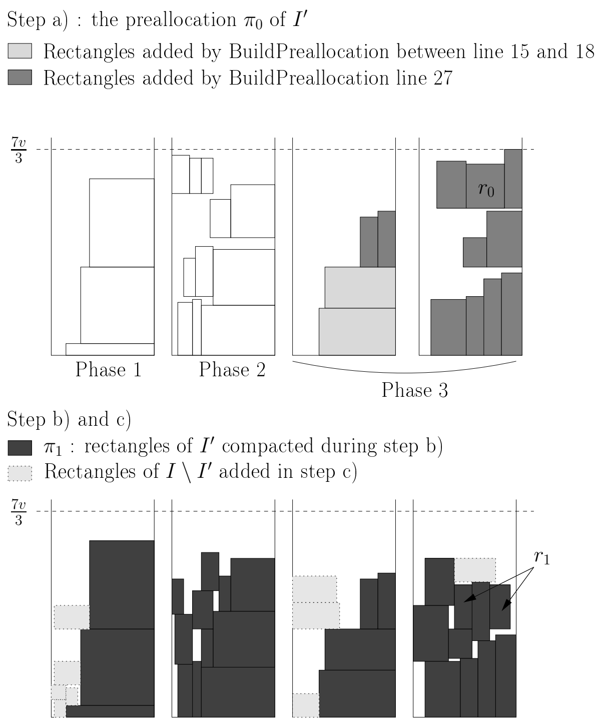
We now describe a three phases algorithm that builds the preallocation π^0 of the rectangles of I'. Remind that π_i^0 denotes the set of rectangles packed in S_i. The description of the *BuildPreallocation* is in Algorithm 1, Page 19, and uses several Lemmas which are detailed later. An overall example of packing (preallocation and re-packing) is depicted in Figure 6. Notice that there is one special case in phase 3 where we pack all the rectangles of I, and not only the ones in I'.

To prove that Algorithm 1 packs I' (or even sometimes I), we will either use area-domination (*i.e.* area argument) or domination (*i.e.* counting argument). Thus, the key ideas of this algorithm are the following.

- Creating a dominating packing with rectangles of $L_{WD} \cap L_{XH}$ is easy as packing 2 such rectangles is sufficient. Thus, Phase 1 starts with rectangles of $L_{WD} \cap L_{XH}$, and requires that $|L_{WD} \cap L_{XH}| \geq 2$.

Algorithm 1 BuildPreallocation

```
1:  $i \leftarrow 0$
2:  ------------------- phase 1 -------------------
3:  while $|L_{WD} \cap L_{XH}| \geq 2$ do
4:    $i \leftarrow i + 1$
5:    $lay_i = CreateLayer(L_{WD}, {}^{7v}/_{3})$
6:    Pack $lay_i$ in $S_i$ with the narrowest rectangles on the top
7:  end while
8:  ------------------- phase 2 -------------------
9:  while $(|L_{WD} \cap L_H| \geq 2)$ and $(W(\bar{L}_{XH}) > 1$ or $W(\bar{L}_H \cup \bar{L}_M) > \frac{3}{2})$ do
10:   $i \leftarrow i + 1$
11:   Create a packing $\pi_i$ in $S_i$ such that $\pi_i$ is dominating and area-dominating
      using Lemma 8
12: end while
13: ------------------- phase 3 -------------------
14: $few\_high \leftarrow (|L_{WD} \cap L_H| < 2)$
15: while $L_{WD}$ is not empty do
16:   $i \leftarrow i + 1$
17:   $lay_i = CreateLayer(L_{WD}, {}^{7v}/_{3})$
18:   Pack $lay_i$ in $S_i$ with the narrowest rectangles on the top
19: end while
20: if $few\_high = true$ then
21:   if $(S(\pi_x) \geq v$ for every $x \leq i - 1)$ then //we target area-domination
22:     //in this case we even pack $I$
23:     for $l = i + 1$ to $N$ do
24:       pack in $S_l$ an area of rectangles of $I$ greater than $v$ (if there are
          enough remaining rectangles) using Lemma 9
25:     end for
26:     pack in $S_i$ all the remaining rectangles of $I$ and the rectangles already
        contained in $S_i$ using Steinberg's (or Widest First) algorithm using
        Lemma 10.
27:   else //we target domination
28:     pack all the remaining rectangles of $I'$ using Lemma 11
29:   end if
30: else
31:   pack all the remaining rectangles of $I'$ using Lemma 12
32: end if
```



Step a) : the preallocation π_0 of I'
Rectangles added by BuildPreallocation between line 15 and 18
Rectangles added by BuildPreallocation line 27
$\frac{7v}{3}$
r_0
Phase 1
Phase 2
Phase 3
Step b) and c)
π_1 : rectangles of I' compacted during step b)
Rectangles of $I \setminus I'$ added in step c)
$\frac{7v}{3}$
r_1

Figure 6: Example of the overall algorithm with $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{3}$. Notice that during step b), r_1 is packed in a non contigous way.

- Creating a dominating packing with rectangles of $L_{WD} \cap L_H$ is hard. Indeed, if the width and height of all the remaining rectangles of $L_{WD} \cap L_H$ are $\frac{1}{2} + \epsilon$ and $\frac{2v}{3} - \epsilon'$, only three such rectangles could be packed in a strip. For appropriate ϵ and ϵ' values, such a packing is neither dominating nor area-dominating. Thus, phase 2 associates smaller rectangles to rectangles of $L_{WD} \cap L_H$ and creates "perfect" strips (*i.e.* dominating and area-dominating)

- Then, two cases are possible at the beginning of phase 3. In the first case ($few\_high = true$), phase 2 stopped because of a lack of rectangles of $L_{WD} \cap L_H$. Thus, in this case the problem of packing these "tricky" rectangles of $L_{WD} \cap L_H$ is "solved", as almost all these rectangles have been packed with smaller ones, avoiding the previous counter example. In the other case ($few\_high = false$), at most one strip will be necessary to pack rectangles of $\bar{L}_{XH} \cup \bar{L}_H \cup \bar{L}_M$ (as ($W(\bar{L}_{XH}) \leq 1$ and $W(\bar{L}_H \cup \bar{L}_M) \leq \frac{3}{2}$)). Thus, we only have to focus on rectangles of L_{WD} and prove that they will be packed in at most $N-1$ strips.

Let us now prove the feasibility of the different phases of Algorithm 1. Notice that it is obvious that phase 1 stops, and that it uses at most $\lfloor \frac{N}{2} \rfloor$ strips since $|L_{XH} \cap L_{WD}| \leq N$.

3.4. Feasibility of phase 2

Lemma 8 (Feasibility of Line 11)**.** *Let us suppose that* ($|L_{WD} \cap L_H| \geq 2$) *and* (($W(\bar{L}_{XH}) > 1$ *or* ($W(\bar{L}_H \cup \bar{L}_M) > \frac{3}{2}$)). *Then, it is possible to create a packing* π_i *such that* π_i *is dominating and area-dominating.*

Proof. Let r_1 and r_2 be in $L_{WD} \cap L_H$, with $w_1 \geq w_2$. We pack r_1 and r_2 right justified, with r_1 at level 0 and r_2 at level h_1. As both are in L_H, $h_1 + h_2 > v$. We could have packed r_2 at level $\frac{2}{3}v$ and still have its top side at or below level $\frac{4}{3}v$. However, this would not create a layer as required by Remark 1 (Page 13). Let us proceed by case analysis, and first suppose that $W(\bar{L}_{XH}) > 1$. In this case we create a shelf sh_1 using $CreateShelf(\bar{L}_{XH}, 1)$, and we pack it at level $\frac{4v}{3}$. Then, we try to pack a rectangle $r_j \in \bar{L}_{XH} \setminus sh_1$ at level 0. Notice that $W(sh_1) \geq 2w_j$ as $CreateShelf$ uses the Widest First order (hence the two widest rectangles of $\bar{L}_{XH}$ were packed in sh_1 and w_j is narrower than both). If r_j fits at level 0, then $S(\pi_i) > (h_1 + h_2)\frac{1}{2} + \frac{2v}{3}(W(sh_1) + w_j) > \frac{v}{2} + \frac{2v}{3} > v$. Moreover, $2f^{XH}(\pi_i) + f^H(\pi_i) + f^M(\pi_i) \geq 2f^{XH}(\pi_i) > 2$. If r_j does not fit, $w_1 > 1 - w_j$ and since $W(sh_1) \geq \max(2w_j, 1 - w_j)$ we get:

$$\begin{aligned}
S(\pi_i) &> h_1w_1 + h_2w_2 + \frac{2v}{3}W(sh_1) \\
&> h_1(1 - w_j) + (v - h_1)\frac{1}{2} + \frac{2v}{3}\max(2w_j, 1 - w_j) \\
&= h_1(\frac{1}{2} - w_j) + \frac{v}{2} + \frac{2v}{3}\max(2w_j, 1 - w_j) \\
&> \frac{3v}{4} + \max(\frac{5w_jv}{6}, \frac{2v}{3} - \frac{7w_jv}{6}) \\
&\overset{w_j=\frac{1}{3}}{\geq} \frac{3v}{4} + \frac{5v}{18} > v
\end{aligned}$$

Moreover, $2f^{XH}(\pi_i) + f^H(\pi_i) + f^M(\pi_i) \geq 2W(sh_1) + w_1 + w_2 > 2W(sh_1) + 1$, and as $W(sh_1) > \frac{1}{2}$ we get $2W(sh_1) + 1 > 2$.

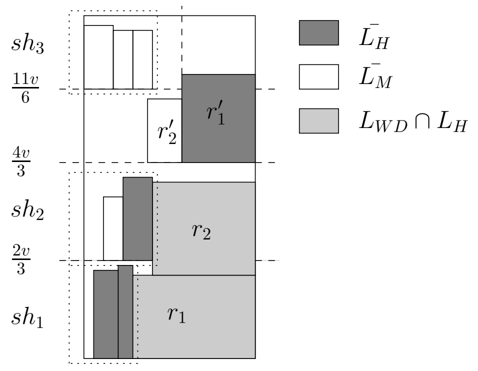


Figure 7: Example of packing built in phase 2 when $W(\bar{L}_H \cup \bar{L}_M) > \frac{3}{2}$.

Let us now suppose that $W(\bar{L}_H \cup \bar{L}_M) > \frac{3}{2}$ (Figure 7). As previously, r_1 and r_2 (taken from $L_{WD} \cap L_H$, with $w_1 \geq w_2$) are packed right justified, with r_1 at level 0 and r_2 at level h_1. We then create at level 0 (using rectangles of $\bar{L}_H \cup \bar{L}_M$) a shelf sh_1, using a **highest first** order. Since r_1 is already packed right justified, the available space for sh_1 is at most $1 - w_1$. Let r'_1 be the first rectangle that does not fit. We pack r'_1 right justified at level $\frac{4v}{3}$. Then we create at level $\frac{2v}{3}$ a second shelf sh_2 using again a **highest first** order. Let r'_2 be the first rectangle that does not fit. We pack r'_2 right justified at level $\frac{4v}{3}$. At this stage the packing is already dominating as $f^H(\pi_i) + f^M(\pi_i) > 2$.

We now prove that we can get an area dominating packing. Notice first that if all rectangles of $\bar{L}_H \cup \bar{L}_M$ are packed, since by hypothesis we have $W(\bar{L}_H \cup \bar{L}_M) > \frac{3}{2}$ and $\forall r_j \in \bar{L}_H \cup \bar{L}_M, h_j > \frac{v}{3}$, then $S(\pi_i) > \frac{v}{2} + \frac{3}{2}\frac{v}{3} = v$ and the proof is finished. Thus, let consider that all the rectangles of $\bar{L}_H \cup \bar{L}_M$ are not packed.

If $r'_2 \in L_H$, then $S(\pi_i) > 2\frac{v}{2} = v$.

If $r'_2 \in L_M$ and $r'_1 \in L_H$, then all the remaining rectangles are in $\bar{L}_M$. Thus, we create sh_3 using $CreateShelf(\bar{L}_M, 1 - w'_1)$ which by definition uses the widest first order, and we pack sh_3 left justified at level $\frac{11v}{6}$ (because sh_3 cannot be stacked on the top of r'_1). Again, either all rectangles of $\bar{L}_H \cup \bar{L}_M$

are packed with this extra shelf and the packing is area dominating, or we get $W(sh_3) > \frac{1-w_1'}{2} > \frac{1}{4}$ (using lemma 1 with $k = 1$). In this case

$$\begin{aligned} S(\pi_i) &> (w_1 + W(sh_1) + w_1')\frac{v}{2} + w_2\frac{v}{2} \\ &\quad +(W(sh_2) + w_2')\frac{v}{3} + W(sh_3)\frac{v}{3} \\ &> \frac{v}{2} + w_2\frac{v}{2} + (1 - w_2)\frac{v}{3} + \frac{v}{12} \end{aligned}$$

which for the $\frac{1}{2}$ lower bound on w_2 leads to v.

If r_2' and r_1' are in L_M, we create sh_3 using $CreateShelf(\bar{L}_M, 1)$, and we pack sh_3 right justified at level $\frac{11v}{6}$. Thus, once more, either all the rectangles are packed and the proof is complete, or we have $W(sh_3) > \frac{2}{3}$ (using lemma 1 with $k = 2$). Then, we get $S(\pi_i) > (w_1 + w_2)\frac{v}{2} + (1 - w_1 + 1 - w_2)\frac{v}{3} + \frac{2}{3}\frac{v}{3}$ which for the lower bound $\frac{1}{2}$ on w_1 and w_2 leads to $S(\pi_i) > v$.

□

3.5. Feasibility of phase 3: filling a strip with an area greater than v

Remember that in phase 3 (after Line 19) the problem gets easier as all the rectangles of L_{WD} have been packed.

Lemma 9 (Feasibility of Line 24)**.** *If we do not run out of rectangles, it is always possible to pack in polynomial time an area of rectangles of* $I \setminus L_{WD}$ *greater than* v *in an empty strip and with a height at most* $\frac{7v}{3}$.

Proof. Let us first mention that we skip complexity considerations as it is clear that the algorithm described below runs in polynomial time. Let *select* be an empty set. We add to *select* some rectangles (in non increasing order of their surface) until $S(select) \geq v$ or $I \setminus L_{WD}$ is empty. Let $select = \{r_1, \ldots, r_p\}$, with $S(r_j) \geq S(r_{j+1})$ for all j. If $S(select) \leq \frac{7}{6}v$, we know according to Steinberg's algorithm that *select* can be packed with a height at most $2 \times \frac{7}{6}v$.

Let us now suppose that $S(select) > \frac{7}{6}v$ (see Figure 8). This implies that the last rectangle r_p added to *select* has a surface strictly larger than $\frac{v}{6}$ (otherwise the algorithm would have stopped before), and thus $S(r_j) > \frac{v}{6}$ for all j. Moreover, we get $p \leq 6$, $h_j > \frac{v}{3}$ (as $r_j \notin L_{WD}$), and $w_j > \frac{1}{6}$ for all $j \in \{1, \ldots, p\}$.

Notice that for $p \leq 4$ the lemma is straightforward as two shelves are sufficient to pack *select*. Moreover, if four rectangles of *select* fit in one shelf, then the lemma is also proved as there is at most two remaining rectangles that fit in a second shelf. Thus, we consider now that $p \geq 5$, and we sort rectangles according to their width, implying now $w_j \geq w_{j+1}$.

If $w_1 + w_2 + w_3 \leq 1$ then $w_4 + w_5 + w_6$ is also lower than 1 and two shelves are sufficient. Thus, we now assume that $w_1 + w_2 + w_3 > 1$ and proceed with a case by case analysis.

Case 1 : $w_1 + w_2 + w_3 > 1$ *and* $w_3 + w_4 + w_5 \leq 1$.
. In this case it is possible to pack $\{r_1, \ldots, r_5\}$ in two shelves sh_1 and sh_2, with $sh_1 = \{r_1, r_2\}$ and $sh_2 = \{r_3, r_4, r_5\}$. Then, we pack sh_1 at level 0 and the rectangles of sh_2 top right justified, such that the highest rectangles are on the right side (see the left case in Figure 8). If $p = 5$, all the rectangles are packed and the proof is over. Let us now study case $p = 6$. We pack r_6 left justified at level $l_6 := \min\{l | w_6 \text{ processors are idle in the considered strip}\}$. If $l_6 = 0$ the packing is clearly feasible, otherwise $l_6 = \min(h_1, h_2)$. Let r_{j_0} be the shortest (with the smallest h_j) rectangle of sh_2, and let us prove by contradiction that r_6 fits. If r_6 intersects sh_2, it implies that r_6 intersects r_{j_0}. Thus, with $\gamma = S(select \setminus \{r_6\})$ we have:

$$\begin{aligned}
\gamma &\geq l_6(1-w_6) \\
&\quad +(\frac{7}{3}v - l_6 - h_6)(W(sh_2)) \\
&= l_6(1-w_6) \\
&\quad +(\frac{7}{3}v - l_6 - h_6)(1-w_6) + (\frac{7}{3}v - l_6 - h_6)(W(sh_2) - (1-w_6)) \\
&> l_6(1-w_6) \\
&\quad +(\frac{4}{3}v - l_6)(1-w_6) + \frac{v}{3}(W(sh_2) - (1-w_6)) \\
&> \frac{4v}{3}(1-w_6) + \frac{v(4w_6-1)}{3} \qquad \text{as } W(sh_2) > 3w_6 \\
&= v
\end{aligned}$$

which is a contradiction. Thus in this case r_6 must fit.

Case 2 : $w_1 + w_2 + w_3 > 1$ and $w_3 + w_4 + w_5 > 1$.

.

Case 2.1 : $p = 5$.

. In this case, we will prove that we can pack the five rectangles in two layers starting at level zero, one left justified and on right justified. First, as each of these rectangles are not in L_{WD}, their width is at most $\frac{1}{2}$ and the two layers won't overlap. Since there are five rectangles, a natural idea is to place the three smallest (according to height) rectangles in one layer and the two biggest in the other. Since all heights are lower or equal to v, the layer with two rectangles will fit in $\frac{7v}{3}$. Let us now prove by contradiction that the other layer also fits in $\frac{7v}{3}$. Let λ be the permutation which orders the five rectangles by non-increasing height, i.e. $h_{\lambda(j)} \geq h_{\lambda(j+1)}$. If the layer with the three smallest rectangle does not fit, we have $h_{\lambda(3)} + h_{\lambda(4)} + h_{\lambda(5)} > \frac{7v}{3}$, and hence $h_{\lambda(3)} > \frac{7v}{9}$ (and this also holds for the two taller rectangles $r_{\lambda(1)}$ and $r_{\lambda(2)}$). Since $h_{\lambda(3)} \leq v$, $h_{\lambda(4)} + h_{\lambda(5)} > \frac{4v}{3}$ and $h_{\lambda(4)} > \frac{2v}{3}$.

The total area of the four largest rectangles is $\sum_{j=1}^{5} w_{\lambda(j)} h_{\lambda(j)} - min_j w_j h_j > (w_{\lambda(1)} + w_{\lambda(2)} + w_{\lambda(3)})\frac{7v}{9} + w_{\lambda(4)}\frac{2v}{3}$. Let $w_{max} = max_{j \leq 4} w_{\lambda(j)}$. Notice that depending on λ, w_{max} is either w_1 or w_2, which are both strictly larger than $\frac{1}{3}$ since $w_3 + w_4 + w_5 > 1$, and thus $\min(w_1, w_2) \geq w_3 > \frac{1}{3}$. In both cases, $(w_{\lambda(1)} + w_{\lambda(2)} + w_{\lambda(3)})\frac{7v}{9} + w_{\lambda(4)}\frac{2v}{3} = (\sum_{j=1}^{4} w_{\lambda(j)})\frac{7v}{9} - w_{\lambda(4)}\frac{v}{9} = ((\sum_{j=1}^{4} w_{\lambda(j)}) - w_{max})\frac{7v}{9} + w_{max}\frac{2v}{3} + (w_{max} - w_{\lambda(4)})\frac{v}{9} > ((\sum_{j=1}^{4} w_{\lambda(j)}) - w_{max})\frac{7v}{9} + w_{max}\frac{2v}{3} > (w_3 + w_4 + w_5)\frac{7v}{9} + w_2\frac{2v}{3} > v$. Hence, p cannot be equal to 5 if the three smallest rectangles do not fit in $\frac{7v}{3}$.

We then repack these layers such that the narrowest rectangles are on the top, implying that the utilization function u of the strip is decreasing.

Case 2.2 : $p = 6$.

. It remains now to handle the case where $p = 6$.

Let us first prove that:

(1) $w_j > \frac{1}{3}$ for $1 \leq j \leq 3$

(2) $w_3 + w_4 > \frac{2}{3}$

(3) $h_{min} \leq \frac{3v}{5}$ with $h_{min} = Min_{j\in\{1,\ldots,5\}} h_j$

(4) $\sum_{j=1}^{4} h_j < 3v$

The first inequality is true as for $1 \leq j \leq 3$, $3w_j \geq w_3 + w_4 + w_5 > 1$. If the second one were false we would have $2w_4 \leq w_3 + w_4 \leq \frac{2}{3}$ and thus $w_3 + w_4 + w_5 \leq w_3 + 2w_4 \leq 1$. The third one is true since $v > S(\cup_{j=1}^{5} r_j) \geq \sum_{j=1}^{5} w_j h_{min} > \frac{5}{3} h_{min}$. The last one is true since

$$\begin{aligned} v &> S(\cup_{j=1}^{4} r_j) \\ &\geq (h_1 + h_2 + h_3) w_3 + h_4 w_4 \\ &= (h_1 + h_2 + h_3 - h_4) w_3 + h_4 w_3 + h_4 w_4 \\ &> (h_1 + h_2 + h_3 - h_4)\frac{1}{3} + \frac{2}{3} h_4 \\ &= \frac{1}{3}(h_1 + h_2 + h_3 + h_4) \end{aligned}$$

Notice that if $w_6 > \frac{1}{3}$ then $\sum_{j=1}^{6} h_j < 4v$ (as $v > S(\cup_{j=1}^{5} r_j) > \frac{1}{3}\sum_{j=1}^{5} h_j$) and as before the two layers are sufficient. Thus, we consider that $w_6 \leq \frac{1}{3}$. For $p = 6$ we have $h_1 + h_2 + h_3 \leq 2v$, since $v > S(\cup_{j=1}^{5} r_j) > \frac{1}{3}(h_1 + h_2 + h_3) + S(r_4) + S(r_5) > \frac{1}{3}(h_1 + h_2 + h_3) + 2\frac{v}{6}$ (recall that $S(r_j) > \frac{1}{6}$ for any j), implying $min_{1\leq j\leq 3} h_i \leq \frac{2v}{3}$ (as $h_1 + h_2 + h_3 \leq 2v$).

If $w_6 \leq \frac{1}{4}$, then we use the same algorithm as for $p = 5$, and then we add r_6 at level $\frac{4v}{3}$. Rectangle r_6 must fit, otherwise $u(\frac{4v}{3}) > \frac{3}{4}$ and thus $S(\cup_{j=1}^{5} r_j) > u(\frac{4v}{3})\frac{4v}{3} > v$. Thus, we consider now that $\frac{1}{3} \geq w_6 > \frac{1}{4}$.

If $w_4 > \frac{1}{3}$ (see Figure 8), we create (at level 0) a layer lay_1 using $CreateLayer(\{r_1, \ldots, r_4\}, \frac{7v}{3})$ and a layer lay_2 containing all the remaining rectangle except r_6. Notice that, as $min_{1\leq j\leq 3} h_j \leq \frac{2v}{3}$ and as $CreateLayer$ uses the highest rectangles first, we get $H(lay_1) > \frac{5v}{3}$. We repack these layers such that the narrowest rectangles are on the top, implying that the utilization function u of the strip is decreasing. Then, we add r_6 at level $\frac{4v}{3}$. If $\{r_1, \ldots, r_4\}$ all fit in lay_1 then it is clear that r_6 fits in $\frac{7v}{3}$ (as $lay_2 = \{r_5\}$). Otherwise, let us prove

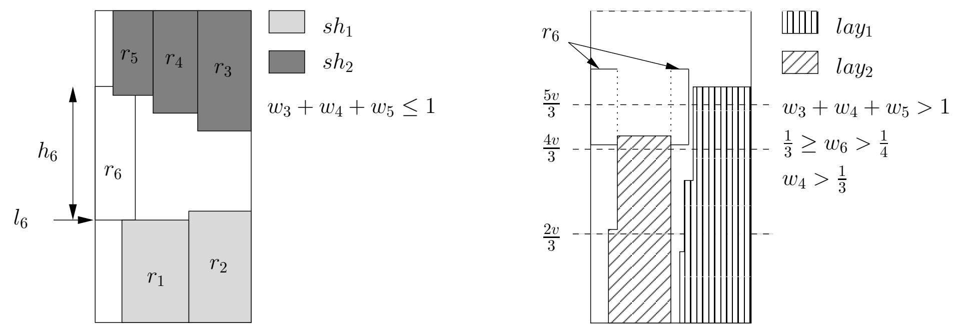


Figure 8: Example of two possible packings built in Lemma 9 when $w_1 + w_2 + w_3 > 1$.

that lay_2 fits:

$$\begin{aligned} H(lay_2) &\leq \sum_{j=1}^{5} h_j - H(lay_1) \\ &\leq 4v - H(lay_1) && \text{according to 4)} \\ &< 4v - \frac{5v}{3} \end{aligned}$$

If r_6 does not fit, we have $u(\frac{4v}{3}) > \frac{2}{3}$, and thus

$$\begin{aligned} S(\cup_{j=1}^{5} r_j) &> u(\frac{4v}{3})\frac{4v}{3} + (min_{1\leq j\leq 4} w_j)(H(lay_1) - \frac{4v}{3}) \\ &> \frac{8v}{9} + \frac{1}{3}\frac{v}{3} \\ &= v \end{aligned}$$

Finally, if $w_4 \leq \frac{1}{3}$ (implying $\sum_{j=4}^{6} w_j \leq 1$), we create one shelf at level 0 with $\{r_4, r_5, r_6\}$, and two layers lay_1 and lay_2 using $CreateLayer(\{r_1, r_2, r_3\}, \frac{4v}{3})$, that we pack at level v. Thus, given that $H(lay_1) > \frac{1}{2}\frac{4v}{3}$, we have $H(lay_2) \leq h_1 + h_2 + h_3 - H(lay_1) \leq 2v - H(lay_1) \leq \frac{4v}{3}$ and thus all the rectangles fit. This concludes the proof of the feasibility of Line 24. □

3.6. Feasibility of phase 3: analysing the possible ends of algorithm

It remains now to study how to finish packing all the rectangles in cases described Lines 26, 28 and 31. Remind that in case described Line 26 we pack all the rectangles of I, and not only I'.

Lemma 10 (First end of Algorithm 1)**.** *In the case Line 26, it is possible to pack all the remaining rectangles of I.*

Proof. Let i_0 be the value of i at Line 19 when all wide rectangles are packed. We have by definition here $S(\pi_x) \geq v$, for any $x \leq i_0 - 1$. Let X be the set of remaining rectangles and $X' = \pi_{i_0}$, at the beginning of Line 26. We will prove that $S(X \cup X') \leq v$, and thus Steinberg algorithm packs $X \cup X'$ with a height lower than $2v$. If $S(X \cup X') > v$, we never ran out of rectangles when packing S_l, $i_0 + 1 \leq l \leq N$ and according to Lemma 9 we have $S(\pi_l) > v$ for $i_0 + 1 \leq l \leq N$, leading to $S(I) > Nv$. □

Lemma 11 (Second end of Algorithm 1)**.** *In the case Line 28, it is possible to pack all the remaining rectangles of I'.*

Proof. Let i_0 be the value of i Line 19. We know that there exists $x \leq i_0 - 1$ such that $S(\pi_x) < v$. It means that we cannot use the same area argument as in Lemma 10. Thus, we will rather use counting arguments (see Section 3.2).

Let i_2 (resp. i_3) be the value of the index of the first strip used in phase 2 (resp. 3). Let us first prove that π_l is dominating (see Definition 4), for $1 \leq l \leq i_0 - 1$. All the strips packed in phase 1 are dominating since $CreateLayer$ packs at least two rectangles of $L_{WD} \cap L_{XH}$ in each strip. According to Lemma 8, all the strips packed in phase 2 are also dominating. Thus, we know that all the π_l are dominating for $1 \leq l \leq i_3 - 1$.

We now prove that the layers created at the beginning of phase 3 are dominating, i.e. π_l is dominating for $i_3 \leq l \leq i_0 - 1$.

Remark 3. *We never ran out of rectangles of L_B when creating lay_l,for any l, $i_3 \leq l \leq i_0 - 1$ (meaning that L_B was not empty when starting creating lay_{i_0}.)*

Proof. Let x be the smallest index such that $S(\pi_x) < v$. We know that S_x can only be a strip packed during phase 1 or phase 3. This implies that $H(lay_x) < 2v$, and thus the algorithm ran out of rectangles of $L_{WD} \backslash L_B$ when creating lay_x. Then, there are only rectangles of L_B in all the layers created after lay_x. □

Thus, when starting phase 3 we know that

- $|L_{WD} \cap L_{XH}| < 2$, as phase 1 finished
- $|L_{WD} \cap L_H| < 2$, as by assumption of this lemma we have $few\_high = true$
- we did not run out of rectangles of L_B when creating lay_l,for any l, $i_3 \leq l \leq i_0 - 1$ in Phase 3

Under these conditions, we will now prove that π_l is dominating for $i_3 \leq l \leq i_0 - 1$. Let (a_{XH}, a_H, a_M) be the number of rectangles of L_{XH}, L_H and L_M added to a layer lay_l, for $i_3 \leq l \leq i_0 - 1$. We have $(a_{XH}, a_H, a_M) \in \{(1,1,1),(0,1,3),(0,0,4)\}$, implying that any lay_l for any l, $i_3 \leq l \leq i_0 - 1$ is dominating.

Thus, π_l is dominating for $1 \leq l \leq i_0 - 1$. Then, according to Lemma 7, we can create (if we don't run out of rectangles) dominating packing in empty strips S_l, $i_0 + 1 \leq l \leq N$. Finally, let X be the remaining rectangles after filling these strips, and let X' be the rectangles of π_{i_0} at Line 19. It remains to pack $X \cup X'$ in S_{i_0}. We prove by case analysis (according to rectangles of L_B packed in π_{i_0}) that if the remaining rectangles do not fit in S_{i_0}, then I' is partitioned in N sets of rectangles $(\pi_1, \dots, \pi_{i_0-1}, X \cup X', \pi_{i_0+1}, \dots, \pi_N)$ that are dominating, which is impossible according to Lemma 6. We only have to consider cases where π_{i_0} is not dominating. Let (b_{XH}, b_H, b_M) be the number of rectangles of $L_{WD} \cap L_{XH}$, $L_{WD} \cap L_H$ and $L_{WD} \cap L_M$ contained in π_{i_0}. As before we know that b_{XH} and b_H are strictly lower than 2. Moreover, as π^0 must be non dominating, we must have $2b_{XH} + b_H + b_M \leq 4$. Thus, the only possible cases are $(b_{XH}, b_H, b_M) \in \{(1,1,0),(1,0,1),(1,0,0),(0,1,j),(0,0,j')\}$, for $0 \leq j \leq 2, 0 \leq j' \leq 3$. For each possible value of (b_{XH}, b_H, b_M), we have to consider different cases according to what kind of rectangles remain in X. For the sake of brevity, we only analyze here two different cases. Proofs for the other cases can be directly adapted.

Let start with $(b_{XH}, b_H, b_M) = (1,1,0)$. Let $X' = \{r_1, r_2\}$ with $r_1 \in L_{WD} \cap L_{XH}$ and $r_2 \in L_{WD} \cap L_H$. We start by repacking r_1 and r_2 from level 0, right justified, such that the narrowest rectangle is on the top. In this case we do not pack rectangles of X on the top of the one of X', since $H(X')$ could be equal to $v+\frac{2v}{3}$, and we could have $X \cap L_{XH} \neq \emptyset$. However, if all the rectangles of X do not fit at level $lvl(r_1)$ (which is the level where r_1 is packed), then we get $W(X) > 1-w_1$. Thus, we have $\sum_{l\in\{XH,H,M\}} f^l(X\cup X') > 2w_1+w_2+(1-w_1) > 2$. Let us now consider a case where we pack rectangles on top of the one of X'. For example with $(b_{XH}, b_H, b_M) = (0,1,1)$. Let $X' = \{r_1, r_2\}$ with $r_1 \in L_{WD} \cap L_H$ and $r_2 \in L_{WD} \cap L_M$. We also start by repacking r_1 and r_2 from level 0, right justified, such that the narrowest rectangle is on the top. We have $h_1+h_2 < \frac{4v}{3}$. If all the rectangles of X do not fit at level $\frac{4v}{3}$, then we get $W(X) > 1$, and $\sum_{l\in\{XH,H,M\}} f^l(X\cup X') > w_1+w_2+1 > 2$.

The other cases can be treated following the same arguments, and we conclude that we can pack $X \cup X'$ in S_{i_0}. □

It remains now to analyze the last possible end of Algorithm 1.

Lemma 12 (Third end of Algorithm 1)**.** *In the case Line 31, it is possible to pack all the remaining rectangles of I'.*

Proof. Let i_3 be the value of the index of the first strip packed in Phase 3. As we packed two rectangles of $L_{XH}\cap L_{WD}$ or $L_H\cap L_{WD}$ in each of the i_3-1 first strips, and as few_high is false, we have $2(i_3-1)+2 \leq |L_{WD}\cap(L_{XH}\cup L_H)| \leq N$, implying $i_3 \leq \lfloor \frac{N}{2} \rfloor$.

As few_high is false at the beginning of phase 3) we get $L_{WD}\cap L_H \geq 2$, implying $W(\bar{L}_{XH}) \leq 1$ and $W(\bar{L}_H \cup \bar{L}_M) \leq \frac{3}{2}$. Thus, we need at most one empty strip to pack $\bar{L}_{XH}\cup\bar{L}_H\cup\bar{L}_M$. Let x be the number of strips packed with rectangles of L_{WD} in phase 3. We need $i_3+x-1 < N$.

If $x \geq 2$, we have

$$\begin{aligned} H(L_{WD}) &> v(i_3-1) + \sum_{j=i_3}^{i_3+x-3} H(lay_j) \\ &\quad +H(lay_{i_3+x-2}) + H(lay_{i_3+x-1}) \\ &> v(i_3-1)+v(x-2)+2v \end{aligned}$$

Thus, as $H(L_{WD}) \leq Nv$, and we get $i_3 + x - 1 < N$. If $x \leq 1$, then $x + i_3 \leq \lfloor \frac{N}{2} \rfloor + 1 < N + 1$.

□

Thus, we proved that the three possible ends of *BuildPreallocation* are feasible. According to the main steps defined in Section 2.2, the *BuildPreallocation* algorithm that preallocates I' is sufficient to get a $7/3$-approximation. Indeed, we simply add rectangles of $I \setminus I'$ using list algorithms defined in Section 2.2

3.7. Complexity

Let us now bound the computational complexity of the algorithm. Remark first that Phase 1 and Phase 2 of *BuildPreallocation* can be implemented in $\mathcal{O}(Nn + n\log(n))$. Indeed, before Phase 1 we sort the wide rectangles according to their height. Thus, all the calls to *CreateLayer* in phase 1 can be implemented in $\mathcal{O}(n)$. Then, before Phase 2 we resort the remaining rectangles according to their width. Thus, all the calls to *CreateShelf* in phase 2 can be implemented in $\mathcal{O}(n)$ (remark that there is only a constant number of rectangles that are packed in Phase 2 without using *CreateShelf*).

Phase 3 runs in $\mathcal{O}(n(N + \log(n)))$, using Widest First rather than Steinberg algorithm. Indeed, this bound is clearly true for algorithms described in Lemma 11 and 12. Applying algorithm described in Lemma 9 on strips $\{S_a, \ldots, S_{a+x}\}$ requires $\mathcal{O}((x+1)n + n\log(n))$. Indeed, we start by sorting the rectangles according to their area, and then each "select" (as named in Lemma 9) set can be created in $\mathcal{O}(n)$. Moreover, applying Widest First on each strip can be done in $\mathcal{O}(\sum_{i=a}^{a+x} n_i \log(n_i)) = \mathcal{O}(n\log(n))$, where n_i is the number of rectangles of *select* in S_i. Thus π^0 is constructed in $\mathcal{O}(n(N + \log(n)))$.

Due to the simple structure of preallocation, the LS_{π^0} algorithm can be implemented in $\mathcal{O}(n\log(n))$. 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. 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 π^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 after l_i 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 $7/3$ 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.

The binary search on v to find the smallest v which is not rejected can be done in $\mathcal{O}(\log(nh_{max}))$. Thus the overall complexity of the $7/3$-approximation is in $\mathcal{O}(\log(nh_{max})N(n+\log(n)))$.

4. 2-approximation for a special case

4.1. Motivation

As explained in the related work, the $2+\epsilon$-approximation in [5] and the 2-approximation we recently proposed in [6] are rather complexity results than practical algorithms. As for the previous $\frac{7}{3}$-approximation, 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 13 ([2]). *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 $P = NP$.*

Sketch. As in [4] for the general version, we construct a gap reduction from the partition problem to the (restricted) MCSP with $N = 2$ strips. We associate one rectangle to each object of the partition instance. The width of a rectangle is equal to the size of the associated object, and all the rectangles have height one. Thus, a yes instance leads to an optimal packing of length one. Conversely, a no instance implies that any packing of height at least two, as the width of rectangle cannot be two partitioned, requiring to pack some rectangles at level one. The condition $w_j \leq \frac{1}{2}$ can be verified by simply scaling rectangles appropriately. $\square$

Therefore, the fast 2-approximation presented in this section is somehow optimal in the sense of approximation theory.

4.2. Sketch of proof

Let us show how to build a preallocation π^0 for $(\alpha, \beta) = (\frac{1}{3}, \frac{1}{2})$ that fits in $r(\alpha, \beta)v = 2v$ (again, we skip property (2) as it will be obvious that rectangles of L_{WD} are preallocated in a $^2/_3$-compact way). Recall that according to Lemma 3 and 4 this is sufficient to get a 2-approximation. We only provide here a sketch of the construction of π^0, and refer the reader to [2, 15] for more details.

Let us consider the following partition (recall that $w_j \leq \frac{1}{2}$ for any j):

- 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
- let $I' = L_{WD} \cup L_{XH} \cup L_H$

Let us sketch how to construct a preallocation π^0 depicted Figure 9.

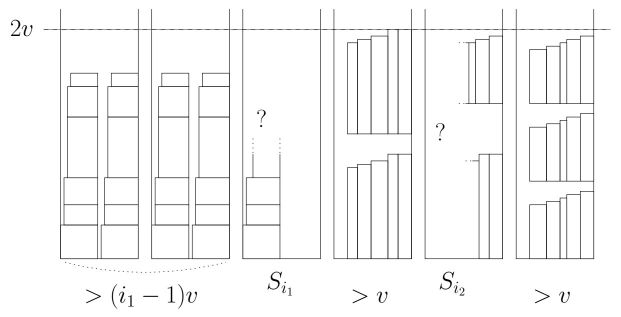


Figure 9: Example of a preallocation. The $(i_1 - 1)$ first strips are optimaly filled using rectangles of L_{WD}. Then, we optimally fill strips $i_1 + 1$ to $i_2 - 1$ using two shelves of extra high rectangles in each strip, and strips $i_2 + 1$ to N using three shelves of high rectangles in each strip. Strips i_1 and i_2 will be carefully filled according to a case by case analysis.

We first pack the rectangles of L_{WD} by packing two layers in parallel in each strip using $CreateLayer(L_{WD}, 2v)$ (let us say using strip 1 to i_1). Thus, if we do not run out of rectangles (this particular case is treated in [1]) all the layers except the last one have a surface greater than $\frac{3}{2}v \times \frac{1}{3} = \frac{v}{2}$ and fit in $2v$.

Then, we pack the remaining rectangles of $L_{XH} \cup L_H$ by creating two kind of shelves. Notice first that

- a "extra high" shelf sh created by $CreateShelf(L_{XH}, 1)$ has a width of at least $\frac{3}{4}$ (as L_{WD} is empty and $w_j \leq \frac{1}{2}$ we know that at least three rectangles fit in the shelf, and we apply Lemma 1), and thus a surface greater than $\frac{2}{3}v \times \frac{3}{4} = \frac{v}{2}$
- a "high" sh_H created by $CreateShelf(L_H, 1)$ has a width of at least $\frac{3}{4}$ (for the same reason) and thus a surface greater than $\frac{v}{2} \times \frac{3}{4} = \frac{3}{8}v$

Thus, we fill empty strips (from $i_1 + 1$ to N) using either two extra high shelves or three high ones. In both cases the total surface packed in each strip is greater than v, and the packing fits in $2v$. Notice that details about how packing the last shelves in strip i_1 and how mixing the two kind of shelves (when we run out of rectangles of L_{XH}) are skipped here.

The previous analysis provides the main arguments to pack an optimal area in a strip using layers or shelves, and thus build the preallocation to get a 2-approximation running in $O(\log(nh_{max})n(N + \log(n)))$.

References

[1] M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, D. Trystram, Approximating the non-contiguous multiple organization packing problem, in: Proceedings of the 6th IFIP International Conference on Theoretical Computer Science, 2010.

[2] M. Bougeret, P.-F. Dutot, K. Jansen, C. Robenek, D. Trystram, Tight approximation for scheduling parallel jobs on identical clusters, in: 14th Workshop on Advances in Parallel and Distributed Computational Models APDCM, (IPDPS), 2012.

[3] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

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

[5] D. Ye, X. Han, G. Zhang, On-Line Multiple-Strip Packing, in: Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications (COCOA), Springer, 2009, p. 165.

[6] M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, D. Trystram, Approximation algorithm for multiple strip packing, in: Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA), 2009.

[7] U. Schwiegelshohn, A. Tchernykh, R. Yahyapour, Online scheduling in grids, in: IEEE International Symposium on Parallel and Distributed Processing (IPDPS), 2008, pp. 1–10.

[8] R. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal of Applied Mathematics 17 (1969) 263–269.

[9] D. Friesen, M. Langston, Evaluation of a multifit-based scheduling algorithm, Journal of Algorithms 7 (1) (1986) 35–59.

[10] 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 (2009) 562–573.

[11] P.-F. Dutot, K. Jansen, C. Robenek, D. Trystram, A $(2+\epsilon)$-approximation for scheduling parallel jobs in platforms (2013).

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

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

[14] D. Hochbaum, D. Shmoys, A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach, SIAM Journal on Computing 17 (3) (1988) 539–551. doi:http://dx.doi.org/10.1137/0217033.

[15] M. Bougeret, P.-F. Dutot, K. Jansen, C. Robenek, D. Trystram, Tight approximation for scheduling parallel jobs on identical clusters, lIRMM research report 12001 http://hal-lirmm.ccsd.cnrs.fr/lirmm-00656780.