INSTITUT FÜR INFORMATIK

Multiple strip packing and scheduling parallel jobs in platforms

Marin Bougeret, Pierre-Francois Dutot, Klaus Jansen, Christina Robenek, Denis Trystram

Bericht Nr. 1108
Oktober 2011
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Multiple strip packing and scheduling parallel jobs in platforms

Marin Bougeret, Pierre-Francois Dutot, Klaus Jansen,
Christina Robenek, Denis Trystram

Bericht Nr. 1108
Oktober 2011
ISSN 2192-6247

e-mail:
marin.bougeret@ens-lyon.f,pierre-francois.dutot@imag.fr,
kj@informatik.uni-kiel.de, cot@informatik.uni-kiel.de,
denis.trystram@imag.fr

Parts of this results have been published as extended abstract in Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA 2009) and Computing and Combinatorics - 17th Annual International Conference (COCOON 2011). The research was supported by EU project "'AEOLUS: Algorithmic Principles for Building Efficient Overlay Computers"', EU contract number 015964, and DFG project JA612/12-1, "'Design and analysis of approximation algorithms for two- and threedimensional packing problems"', and DGA-CNRS

Abstract

We consider two strongly related problems, multiple strip packing and scheduling parallel jobs in platforms. In the first one we are given a list of n rectangles with heights and widths bounded by one and N strips of unit width and infinite height. The objective is to find a non-overlapping orthogonal packing without rotations of all rectangles into the strips minimizing the maximum height used. In the scheduling problem we consider jobs instead of rectangles, i.e. we are allowed to cut the rectangles vertically and we may have target areas of different size, called platforms. A platform P_ℓ is a collection of m_ℓ processors running at speed s_ℓ and the objective is to minimize the makespan, i.e. the latest finishing time of a job.

1 Introduction

We consider two strongly related problems, multiple strip packing and scheduling parallel jobs in platforms. Strip packing (SP) is the geometric version of the cutting-stock problem. Here we are given a list of n rectangles $L = \{r_j | j \in \{1, \ldots, n\}\}$ with heights h_j and widths w_j bounded by one and a strip of unit width and infinite height. The objective is to find a non-overlapping orthogonal packing without rotations of all rectangles into the strip minimizing the total height used. The optimum value for this height will be denoted with $\mathrm{OPT}_{SP}(L)$. Multiple strip packing (MSP) is a generalization of the strip packing problem. Here the rectangles have to be packed into N identical strips $S_1, \ldots S_N$. For each strip S_ℓ, $\ell \in \{1, \ldots, N\}$ we denote with $h(\ell)$ the height of the packing in this strip. The objective in MSP is to minimize $\max_\ell h(\ell)$. For an instance of MSP we simply write L suppressing the number of strips (that will always be N). The optimum value of MSP for a given list of rectangles L will be denoted with $\mathrm{OPT}_{MSP}(L)$.

The second problem is called scheduling parallel job in platforms (SPP). As strip packing is closely related to scheduling parallel jobs on identical machines, we consider its generalization to N *platforms* of processors. Moreover we are given n *jobs* $J = \{J_1, \ldots, J_n\}$ and N heterogeneous *platforms* P_ℓ each of them representing a set of m_ℓ processors (machines) running at speed $s_\ell \in \mathbb{R}_{>0}$. We assume the platforms to be ordered non-decreasingly by their number of processors, i.e. $m_1 \leq m_2 \leq \ldots \leq m_N$. Every job J_j is described by a pair (p_j, q_j) of the *length of a job* p_j (*number of operations*) and a *number of parallel processors* q_j (*degree of parallelism*) that J_j requires when executed. We assume $q_j \leq m_N = \max_\ell m_\ell$ for all jobs, if not there is no feasible schedule. Since sometimes we will identify jobs with rectangles, we call q_j the width of job J_j. A job J_j is only allowed to be scheduled within one platform, its processing time in platform P_ℓ is $t_j^\ell := \frac{p_j}{s_\ell}$ if $q_j \leq m_\ell$ else $t_j^\ell := \infty$. We assume furthermore (by scaling) $\min_\ell s_\ell = 1$ and define $t_{\max} := \max_{j,\ell}\{t_j^\ell | t_j^\ell < \infty\}$, which is less than $p_{\max} := \max_j p_j$ (as $\min_\ell s_\ell = 1$). Our objective is to find a non-preemptive schedule of all jobs into the platforms minimizing $C_{\max} := \max_\ell C_{\max}(\ell)$, where $C_{\max}(\ell)$ denotes the completion time of a feasible schedule in P_ℓ, i.e. the latest finishing time of a job in P_ℓ. For an instance J of *SPP* let $\mathrm{OPT}_{SPP}(J)$ denote the optimum value for $C_{\max}$.

The *size* (sometimes called *work*) of a job is $SIZE(J_j) = p_j q_j$ and for a list of jobs J we define $SIZE(J) = \sum_{j \in J} p_j q_j$. Analogously we define the size of a rectangle $SIZE(r_j) = w_j h_j$ and the size of a list of rectangles $SIZE(L) = \sum_{j \in L} w_j h_j$. The to-

tal processing time (or length) of a list of jobs J is $P(J) = \sum_{j \in J} p_j$. Similar we define the total height of a list of rectangles L as the sum over all heights $H(L)$.

The quality of an approximation algorithm is measured by its performance ratio. For an instance I of a minimization problem with optimum value $\mathrm{OPT}(I)$ as *MSP* and *SPP* we say that B has *absolute ratio* α, if $\sup_I {}^{B(I)}/_{\mathrm{OPT}(I)} \leq \alpha$, and *asymptotic ratio* α, if $\alpha \geq \limsup_{\mathrm{OPT}(I)\to\infty} {}^{B(I)}/_{\mathrm{OPT}(I)}$, respectively. A minimization problem admits an *(asymptotic) polynomial-time approximation scheme* ((A)PTAS), if there exists a family of polynomial-time approximation algorithms $\{B_\varepsilon | \varepsilon > 0\}$ of (asymptotic) $(1+\varepsilon)$-approximations. We call an approximation scheme *fully polynomial* ((A)FPTAS), if the running time of every algorithm A_ε is bounded by a polynomial in n and $\frac{1}{\varepsilon}$.

Both problems *MSP* and *SPP* are closely related where at first sight the scheduling problem offers us more possibilities to manipulate the instance to find an optimal solution while the packing problem seems to be more rigid. We will see that there are similar LP-relaxations based on fractional bin packing that can be used to give approximation algorithms for both problems.

1.1 Known results

Multiple strip packing was first considered by Zhuk [24], who showed that there is no approximation algorithm with absolute ratio better than 2, and later by Ye et. al. [23]. Both concentrated on the online case. Additionally an approximation algorithm for the offline case with ratio $2+\varepsilon$ was achieved in [23].

In [7] Coffman et al. gave an overview about performance bounds for shelf-orientated algorithms for strip packing as $NFDH$ (Next Fit Decreasing Height) and $FFDH$ (First Fit Decreasing Height). Those adopt an absolute ratio of 3, and 2.7, respectively. Schiermeyer [18] and Steinberg [21] presented independently an algorithm for SP with absolute ratio 2. Recently, Harren et al. gave an algorithm for strip packing with absolut ratio close to $5/3$ [10]. An important result is an AFPTAS for strip packing with additive constant $\mathcal{O}({}^1/_{\varepsilon^2} h_{\max})$ given by Kenyon and Rémila in [14], where $h_{\max}$ denotes the height of the tallest rectangle (i.e. the length of the longest job). This constant was improved by Jansen and Solis-Oba who presented in [12] an APTAS with additive constant $h_{\max}$.

If the jobs are assigned to processors of consecutive addresses, the problem *SPP* corresponds directly to *MSP*. But, keep in mind that in general because of the contiguity constraint algorithms for *SPP* cannot be directly applied to the generalized *MSP*, because rectangles may be cut. But the optimal value for generalized *MSP* is an upper bound for the optimal value for *SPP*. Schwiegelshohn *et al.* [20] achieved ratio 3 for scheduling parallel jobs on heterogeneous platforms with identical speeds without release times, and ratio 5 with release times. Tchernykh *et al.* presented in [22] an algorithm with absolute ratio 10 without release times. For scheduling parallel jobs on identical platforms, we proposed a low cost approximation algorithm with absolute ratio $5/2$ in [3]. Recently, we were able to improve our result to a fast $5/2$-approximation for heterogeneous platforms with identical speeds and under the additional constraint that every job can be scheduled in each platform [4].

1.2 Our results

In the first part of this article we present several results for multiple strip packing. We present an approximation algorithm with absolute ratio 2, which is an improvement of the former result of $2+\varepsilon$ by Ye et al. [23] and best possible, unless $\mathcal{P} = \mathcal{NP}$. Furthermore, we show how to use the shelf-based heuristics $NFDH$ and $FFDH$ to obtain approximation algorithms for *MSP* with the same asymptotic ratio as for strip packing. We also introduce an AFPTAS for multiple strip packing, which is a based on a generalization of an improved version of the algorithm of Kenyon and Rémila [14]. By using a different rounding technique we can reduce the additive constant of the strip packing algorithm by Kenyon and Rémila to $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$ and its running time. We generalize that algorithm to several strips and achieve the following:

Theorem 1.1. *For any instance L of* MSP *and accuracy $\varepsilon > 0$ there is an algorithm that produces a packing into N strips of unit width with height at most*

$$(1+\varepsilon)OPT_{MSP}(L) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$$

and running time polynomial in the size of the input $|L|$ and $1/\varepsilon$. For N sufficiently large, namely $N = \Omega(1/\varepsilon^2 \log(1/\varepsilon))$ the packing produced has height at most

$$(1+\varepsilon)OPT_{MSP}(L) + \mathcal{O}(1)h_{\max}.$$

This is also an improvement of the first version of the algorithm published as an extended abstract in [2].

The same algorithm can be applied to scheduling parallel jobs in identical platforms achieving the same approximation ratio. The problem with heterogeneous platforms (SPP) is more complex since the platforms may have different numbers of processors and speeds. The algorithm for *MSP* does not apply here because it is based on cutting a solution for a single strip and distributing it well-balanced. Additionally, we do not assume that every job fits in every platform. Thus, the algorithm in [4] does also not apply. For *SPP* we found a different approach which also leads to an AFPTAS. Our algorithm first transforms the problem into a 2-dimensional bin packing problem with different bin sizes and formulates an LP-relaxation for it. Solving the LP gives a fractional assignment of the jobs to the platforms. By clever rounding the solution of the LP we get an integral assignment of nearly every job. Applying the improved strip packing algorithm the preassigned jobs are repacked. This also improves the running time and the additive term of the first version of the algorithm published as an extended abstract in [5].

Theorem 1.2. *For every accuracy ε there exists an approximation algorithm that produces for every instance J of* SPP *a schedule of length at most*

$$(1+\varepsilon)OPT_{SPP}(J) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}$$

and running time polynomial in the size of the input $|J|$ and $1/\varepsilon$.

Furthermore our algorithm can easily be modified to handle malleable jobs or release times. For malleable jobs we achieve the same approximation ratio while for *SPP* with release times we present an AFPTAS with additive factor $O(1/\varepsilon^2 \log(1/\varepsilon))p_{\max}$.

1.3 Organization of the paper

In Section 2 we give generalizations of $NFDH$ and $FFDH$ for *MSP*. Section 3 contains our 2-approximation for *MSP*. We describe how to improve the algorithm of Kenyon and Rémila in Section 4 and give an AFPTAS for MSP in Section 5. Finally we present an AFPTAS for *SPP* in Section 6 and discuss its modification for malleable jobs in Section 8 and for release times in Section 9.

2 Shelf-based algorithms for MSP

We start with some elementary results for the NFDH and FFDH heuristics [7] applied to several strips that will give an understanding of the problem. The NFDH and FFDH heuristics are so-called shelf-based algorithms. A *shelf* is a row of items placed next to each other aligned by their edges. The bottom of a shelf is either the bottom of the bin or at the same height as the upper edge of the tallest item packed in the shelf below. Typically the principle of operation of those algorithms is easy to grasp, but it can be difficult to analyze them.

Theorem 2.1. *Let A be one of the shelf-based strip packing algorithms NFDH and FFDH with asymptotic ratio $\alpha > 1$, that creates for a list of rectangles L a packing into a single strip of unit width with height less than $\alpha OPT_{SP}(L) + h_{\max}$. For any $N \in \mathbb{N}$ there exists an algorithm A_N that packs a list of rectangles L into N strips with $\max_{\ell \in \{1,\dots,N\}} h(\ell) \leq \alpha OPT_{MSP}(L) + h_{\max}$.*

Proof. Consider Algorithm 1. We show that for a list of rectangles L, the packing produced by A_N has height less than $\alpha \mathrm{OPT}_{MSP}(L) + h_{\max}$. Let $t \in \mathbb{N}$ be the number of shelves produced by A in Step 1 and H_j, $j \in \{1, \dots, t\}$, the height of the jth shelf. Since there are no items intersecting the 0th line, after the last step of the algorithm the height $h(1)$ of the first strip S_1 is bounded by $h_{\max} + \frac{\sum_{j=1}^{t} H_j - h_{\max}}{N}$. For any strip S_ℓ, $\ell \in \{2, \dots, N\}$, containing items from between the $(\ell-1)$th and the ℓth line and the ones intersecting the $(\ell-1)$th line, we have

$$h(\ell) \leq \frac{\sum_{j=1}^{t} H_j - h_{\max}}{N} + h_{\max} = \frac{A(L) - h_{\max}}{N} + h_{\max}.$$

With OPT_{SP} as the optimum of SP we conclude

$$\begin{aligned} A_N(L) = \max_i h_i &\leq \frac{A(L) - h_{\max}}{N} + h_{\max} \\ &\leq \frac{\alpha \mathrm{OPT}_{SP}(L) + h_{\max} - h_{\max}}{N} + h_{\max} = \frac{\alpha \mathrm{OPT}_{SP}(L)}{N} + h_{\max}. \end{aligned}$$

Since $1/N \mathrm{OPT}_{SP}(L)$ is a lower bound for $\mathrm{OPT}_{MSP}(L)$ the proof is complete. □

Corollary 2.2. *The algorithms $FFDH_N$ and $NFDH_N$ generate packings for a set of rectangles L into N strips with height less than $1.7OPT_{MSP}(L) + h_{\max}$ and $h_{MSP} \leq 2OPT_{MSP}(L) + h_{\max}$, respectively.*

Algorithm 1 A_N

1: Pack rectangles with A in one strip S. (Keep in mind that both, NFDH and FFDH order the rectangles by height first) Let H be the height of S.
2: Cut out the first shelf and pack it into the first strip S_1.
3: **for all** $\ell \in \{0, 1, \ldots, N\}$ **do**
4: Draw a horizontal line trough S at height $\ell(H - h_{\max})/N$.
5: **end for**
6: **for all** $\ell \in \{0, 1, \ldots, N-1\}$ **do**
7: Pack all items intersecting the ℓth line and all items between the ℓth and $(\ell+1)$th lines into strip $\ell + 1$.
8: **end for**

Corollary 2.3. *Let L be an instance of MSP. In a packing generated by the above algorithm A_N we have*

$$\max_{\ell \in \{1,\ldots,N\}} |h(\ell) - A_N(L)| \leq 2h_{\max},$$

where $h(\ell)$ denotes the height of the packing in strip S_ℓ.

Proof. By construction height of the packing for the rectangles selected between the ℓth and $(\ell+1)$th line is at least $\frac{\sum_{j=1}^{t} H_j - h_{\max}}{N} - h_{\max}$ and at most $\frac{\sum_{j=1}^{t} H_j - h_{\max}}{N} + h_{\max}$. □

Another way to pack a set of rectangles with a modified version of the $NFDH$ heuristic into N strips is Algorithm 2. The packing generated by that algorithm is very smooth,

Algorithm 2

1: Order the rectangles by non-increasing height.
2: For each $\ell \in \{1, \ldots, N\}$ pack one shelf according to the $NFDH$ heuristic in strip S_ℓ, that means starting in the lower left corner pack the rectangles next to each other on the baseline of strip S_ℓ, until the next rectangle does not fit. Draw a new baseline at the height of the highest rectangle (that clearly is the first one).
3: Take the strip S^- with the current lowest height h^- (if there is more than one, take the one with the smallest index) and pack one shelf according to the $NFDH$ heuristic on top of the shelves.
4: Repeat Step 3 until all rectangles are packed.

in the sense that the heights of the strips only differ by $h_{\max}$.

Lemma 2.4. *For a set of rectangles $L = \{r_1, \ldots, r_n\}$ Algorithm 2 generates a packing into N strips, so that*

$$\max_{\ell,k \in \{1,\ldots,N\}} |h(k) - h(\ell)| \leq h_{\max}.$$

Proof. Let $a_1, \ldots, a_r$ denote the shelves created by the algorithm and let b_i denote the height of the first rectangle placed in shelf a_i, $i \in \{1, \ldots, r\}$, clearly $b_1 \geq \ldots \geq b_r$. We show per induction on i that the claim is true after creating shelf a_i.
During step 2 the assertion is obviously true. Let $1 \leq i_0 < r$ and assume that the assertion is true for the current packing with shelves a_i, $i \leq i_0$. Let S^- be the strip with

the lowest current height h^- and h^+ the height of the currently highest strip S^+ after packing shelf a_{i_0}. The value $h(\ell)^*$ denotes the height of strip S_ℓ after packing the next shelf a_{i_0+1}. Then $h^+ > h^- + b_{i_0+1}$ or $h^+ \leq h^- + b_{i_0+1}$. In the first case we conclude $h(\ell)^* \in [h^-, h^+]$ for all $\ell \in \{1, \ldots, N\}$. Since $|h^+ - h^-| \leq h_{\max}$ by induction hypothesis, the assertion follows. In the second case the assertion is true, because since $b_{i_0+1} \leq h_{\max}$ we have $h(\ell)^* \in [h^-, h^- + h_{\max}]$ for all $\ell \in \{1, \ldots, N\}$. □

This leads to a further result about rectangles with bounded width.

Theorem 2.5. *For a set of rectangles $L = \{r_1, \ldots, r_n\}$ with width bounded by ε we obtain by Algorithm 2 a packing into N strips with height less than*

$$\frac{1}{1-\varepsilon} OPT_{MSP}(L) + 2h_{\max}. \tag{1}$$

Proof. Let $SIZE(S_\ell)$ denote the total area of the rectangles packed into S_ℓ. We consider the strip $S_{\min}$ with $SIZE(S_{\min}) = \min_{\ell \in \{1,\ldots,N\}} SIZE(S_\ell)$. Let $a_1, \ldots, a_r$ be the ordered sequence of shelves constructed by algorithm 2. Furthermore, let b_i and b'_i be the heights of the first and last rectangle placed in shelf $a_i, i \in \{1, \ldots, r\}$. We have $b_1 \geq b'_1 \geq \ldots \geq b_r \geq b'_r$. A shelf is closed when the next rectangle does not fit completely on the shelf. Notice that all narrow rectangles on shelf a_i have heights $\geq b'_i$. For $i \in \{1, \ldots, r-1\}$ the total width of the rectangles packed on shelf a_i is larger than $(1-\varepsilon)$. Therefore on these shelves a_i an area of $(1-\varepsilon)b'_i$ is fully covered by rectangles and thus $SIZE(S_{min}) \geq \sum_{i=1}^{r-1}(1-\varepsilon)b'_i$. Let $h(S_{\min})$ denote the height of strip $S_{\min}$. With $SIZE(S_{\min}) \leq {}^{SIZE(L)}/_{N}$ and Lemma 2.4 we conclude

$$\begin{aligned} h_{MSP} \leq h(S_{\min}) + h_{\max} &= \sum_{l=1}^{r} b_r + h_{\max} \leq \sum_{l=1}^{r-1} b'_r + 2h_{\max} \\ &\leq \frac{SIZE(S_{\min})}{1-\varepsilon} + 2h_{\max} \leq \frac{SIZE(L)}{N(1-\varepsilon)} + 2h_{\max} \leq \frac{\mathrm{OPT}_{MSP}(L)}{1-\varepsilon} + 2h_{\max}. \end{aligned}$$

□

3 A 2-Approximation for MSP

Since there is no approximation algorithm for MSP with absolute ratio smaller than 2 (unless P=NP), a 2-approximation is best possible for MSP. In this section we show the following theorem.

Theorem 3.1. *For any $N \in \mathbb{N}$ there is a polynomial-time algorithm for MSP with absolute ratio two.*

To handle different sizes of N different subroutines are used to obtain a 2-approximation. For $N = 1$ *MSP* is equivalent to strip packing and we can use the algorithm of Steinberg [21] or Schiermeyer [18] with absolute performance bound 2.

Theorem 3.2 (Steinberg [21]). *Let $L = \{r_1, \ldots, r_n\}$ be a set of rectangles with heights h_i and widths w_i and Q be a rectangle with width u and height v. Let $h := \max_{i\in\{1,\ldots,n\}} h_i$ and $w := \max_{i\in\{1,\ldots,k\}} w_i$. If the following inequalities hold,*

$$w \leq u, \qquad h \leq v, \qquad 2SIZE(L) \leq uv - (2w-u)_+(2h-v)_+ \tag{2}$$

then it is possible to pack L into the rectangle Q.(As usual, $x_+ = \max(x, 0)$.)

If N is sufficiently large, the algorithm of Caprara [6] for 2-dimensional bin packing (2DBP) with asymptotic ratio 1.69... gives us a 2-approximation:
Two-dimensional bin packing is the 2-dimensional generalization of bin packing, where a set of rectangles with widths and heights bounded by one has to be packed into a minimum number of unit squares, called bins. Moreover in [19] it is shown that Caprara's algorithm already gives a 2-approximation if $OPT_{2DBP}(I) \geq 1446$ for an instance I. Given an instance L of MSP we transform it into an instance of 2-dimensional bin packing $\tilde{L}$ with $\mathrm{OPT}_{DBP}(\tilde{L}) = N$ by scaling the height of any item with $1/\mathrm{OPT}_{MSP}(L)$. The value $\mathrm{OPT}_{MSP}(L)$ can be found via binary search in time $\mathcal{O}(\log(nh_{\max})$ (see Lemma 3.3 below for more details). For $N \geq 1446$ the algorithm of Caprara gives a packing for $\tilde{L}$ into at most $2N$ unit square bins in time $\mathcal{O}(n\log(n)) + T$ where T is the running time of an AFPTAS for bin packing. Stacking every two bins on top of each other and rescaling gives us a 2-approximation for *MSP*.

Lemma 3.3. *Let $L = \{r_1, \ldots, r_n\}$ be an instance of* MSP*, so that we have for the heights of the rectangles $h_i \in \mathbb{Q}$. Binary search finds in polynomial time the height of an optimal solution.*

Proof. For each height h_i there exist $q_i, p_i \in \mathbb{N}$ with $h_i = {}^{p_i}/_{q_i}$. For $i \in \{1, \ldots, n\}$ we have $Qh_i \in \mathbb{N}$, where $Q = \prod_{i=1}^{n} q_i$. Since $OPT(L)$ is equal to the sum of heights of rectangels from L, we also have $Q\mathrm{OPT}(L) \in \mathbb{N}$. So for the height v of an optimal solution we conclude that $Qh_{\max} \leq Qv \leq Qnh_{\max}$. Since $log_2(Qnh_{\max}) = \sum_{i=1}^{n} log_2(q_i) + log_2(n) + log_2(h_{\max}) \leq |L|$, where $|L|$ is the length of the input, Binary Search finds in polynomial time the value Qv and so the value v. □

In case $2 \leq N < 1446$ there is something more to do. Here we use a PTAS for rectangle packing with area maximization (RPA) found by Bansal et al. [1]. In RPA we are given a set of rectangles $L = \{r_1, \ldots, r_n\}$ with widths w_i and heights h_i and a bin of unit size. The objective is to find a feasible packing of a subset L' of the rectangles into the bin while maximizing the total area of the rectangles in L'. Let us first consider $N = 2$. Algorithm 3 gives us a 2-approximation in this case.

If $2 < N < 1446$ we use an extended version of the PTAS for RPA in [1] for several strips. Moreover, one can show the following

Theorem 3.4. *Given a constant number N of bins, a fixed value ε and a set of rectangles $L = \{r_1, \ldots, r_n\}$ there is a polynomial time algorithm $A_{N,\varepsilon}$ that finds a subset $L' \subset L$ with total area at least $(1-\varepsilon)$ times the optimal value and a packing for L' into N bins or decides that no such subset exists.*

For a detailed proof we refer to [19]. Together with the next assertion we can state that Algorithm 4 that gives us a 2-approximation.

Algorithm 3 2-Approximation for MSP if N=2

1: Guess the height of an optimal solution for *MSP* and denote it with v.
2: Scale the heights of the rectangles in L by $1/v$ so that the corresponding packing fits into one bin of height 2 and width one.
3: The set of resulting rectangles L_v is now considered as an instance of RPA with $\mathrm{OPT}_{RPA}(L) = SIZE(L_v)$, where $SIZE(L_v)$ is the total area of all rectangles in L_v. Apply the algorithm in [1] with accuracy $\varepsilon = 1/4$ and find a packing of a subset $L'_v \subset L_v$ with total area at least $(1-\varepsilon)SIZE(L_v)$ into a bin of height 2. By rescaling the rectangles of L'_v get a packing for the first strip with height at most $2v$.
4: Since $SIZE(L_v) \leq 2$ the remaining items in $L_v \backslash L'_v$ have total area $SIZE(L_v \backslash L'_v) \leq \varepsilon SIZE(L_v) \leq 1/2$. Therefore we can pack them with Steinberg's algorithm into a strip of height at most 2. Rescaling gives us a second strip of height at most $2v$.

Lemma 3.5. *Let $N \geq 3$ and L be an instance of 2DBP with total area $SIZE(L) \leq N/4$. There exists a packing of L into N bins.*

Proof. Since for $h, w, u = 1$ and $v = N/2$ the inequalities (2) hold, we can apply Steinberg's algorithm. By this we get a solution for SP with height at most $N/2$. By drawing $\lceil N/2 \rceil + 1$ horizontal lines with distance one through the strip starting at the bottom we divide the strip into $\lceil N/2 \rceil$ bins of height one and $\lceil N/2 \rceil - 1$ sets of cut items. Packing each set of fractional items into an extra bin we use at most $\lceil N/2 \rceil + \lceil N/2 \rceil - 1 \leq N$ bins. □

Algorithm 4 2-Approximation for MSP if $2 < N < 1446$

1: Guess an optimal height for MSP and denote it with v.
2: Scale the heights of the rectangles of L by $1/v$ so that the corresponding packing fits into N bins of height and width one.
3: The set of resulting rectangles L_v is now considered as an instance of RPA with $\mathrm{OPT}_{RPA}(L) = SIZE(L_v)$. Extend the PTAS of Bansal et al. [1] to N bins of unit size and find for an accuracy $\varepsilon \leq 1/4$ a packing for a subset $L'_v \subset L_v$ with total area $(1-\varepsilon)SIZE(L_v)$. By rescaling the rectangles of L'_v we get N bins of height v.
4: For the total area of the remaining rectangles in $L_v \backslash L'_v$ we have $SIZE(L_v \backslash L'_v) = \varepsilon SIZE(L_v) \leq N/4$. Pack those rectangles according to Lemma 3.5 into N bins and rescale the rectangles. This results again in N bins of height at most v.
5: Stack every two bins on top of each other and get a solution with bins of height at most $2v$.

The running time for both Algorithms 3 and 4 dominated by the running time of the Algorithm for RPA, which is doubly exponential in $1/\varepsilon = 4$, i.e. $\mathcal{O}(n^{256})$.

4 Improved Strip Packing Algorithm

In this section we show how to improve the additive constant in the algorithm of Kenyon and Rémila from $\mathcal{O}(1/\varepsilon^2)h_{\max}$ to $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$.

Theorem 4.1 (Kenyon & Rémila [14]). *For a list $L = \{r_1, \ldots, r_n\}$ of rectangles with widths and heights ≤ 1 and accuracy $\varepsilon > 0$ the algorithm A_ε^{KR} in [14] generates a packing into one strip with height at most $(1+\varepsilon)OPT_{SP}(L) + (4(\frac{2+\varepsilon}{\varepsilon})^2 + 1)h_{\max}$.*

By applying a different rounding technique than in the original algorithm in [14] we reduce the additive term. Roughly described the algorithm by Kenyon and Rémila works as follows:
For some value $\varepsilon' \in \Theta(\varepsilon)$ the rectangles in L are partitioned into $L_{wide} := \{r_j \in L | w_j > \varepsilon'\}$ and $L_{narrow} := \{r_j \in L | w_j \leq \varepsilon'\}$. Then L_{wide} is rounded to an instance L_{sup} with $O(1/\varepsilon^2)$ different widths. The grouping and rounding technique used for the wide rectangles called "'geometric rounding"' and is described by Algorithm 5 and Fig 1.

Algorithm 5 Rounding I

1: Put the wide rectangles ordered by non-increasing widths and pack them left-aligned on a stack with height $H := H(L_{sup})$.
2: Let $M := (1/\varepsilon')^2$
3: Draw $M-1$ horizontal lines through the stack with distance H/M starting at the bottom. Therefore we get M so-called *threshold* rectangles. A rectangle is a threshold rectangle if it either with its interior or with its lower edge intersects a line at height iH/M, $i \in \{1, \ldots, M-1\}$.
4: **for all** $i \in \{1, \ldots, M-1\}$ **do**
5: Round up the width of each rectangle between the lines iH/M and $(i+1)H/M$ to the width of the ith threshold rectangle. The widths of the rectangles below the first line are rounded up to the width of the undermost rectangle in the stack.
6: **end for**
7: Let L_{sup} denote the list of rounded rectangles.

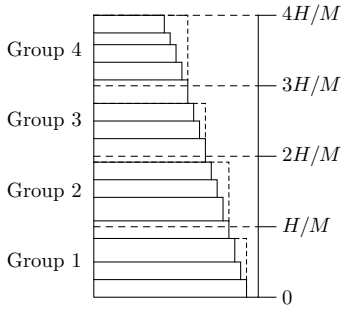

Figure 1: Rounding the rectangles in $L_{\sup}$.

The main part of the algorithm is to produce a fractional packing for the rectangles in L_{sup} using a linear program for fractional bin packing. In doing so we build configurations $C_j := \{\alpha_{ij} : w_i | i \in \{1, \ldots, M\}\}$, i.e. multisets of widths where α_{ij} denotes the number of rectangles of width w_i in C_j^ℓ and $\sum_{i=1}^{M} \alpha_{ij} w_i \leq 1$. Then the following LP is solved.

$$\begin{aligned} \min \sum_{j=1}^{q} x_j \\ \text{s.t.} \sum_{j=1}^{q} \alpha_{ij} x_j \geq \beta_i \text{ for all } i \in \{1, \ldots, M\} \\ x_j \geq 0 \text{ for all } j \in \{1, \ldots, q\}. \end{aligned} \tag{3}$$

The variable x_j indicates the height of configuration C_j, β_i is the total height of rectangles of width w_i in L_{sup} and q denotes the number of possible configurations. A feasible solution of the LP corresponds to a fractional strip packing. Note that $\text{rank}(\alpha_{ij})_{ij} \leq M$ and hence a basic solution x of (3) has at most M nonzero entries. The fractional packing can be converted into an integral one increasing the height only by $2Mh_{\max}$. Moreover, the rounded instance of wide rectangles has nice properties:

Lemma 4.2. *[14] The rounded instance L_{sup} has the following properties*

1. *The number of different width in L_{sup} is bounded by $M = \mathcal{O}(1/\varepsilon^2)$.*
2. *$FSP(L_{sup}) \leq FSP(L_{wide}) \left(1 + \frac{1}{M\varepsilon'}\right)$*
3. *$SIZE(L_{sup}) \leq SIZE(L_{wide}) \left(1 + \frac{1}{M\varepsilon'}\right)$*

Finally, the narrow rectangles in L_{narrow} are added in the remaining space next to the integral packing and on top of it with Next Fit Decreasing Height heuristic.

4.1 Modified geometric rounding

Our improved algorithm uses a different geometric rounding technique. We choose a value $\varepsilon' := \varepsilon/2$ and partition the rectangles into wide L_{wide} and narrow ones L_{narrow} as before. Then we apply geometric grouping with parameter k introduced by Karmarkar and Karp in [13] for fractional bin packing and obtain an instance $J \cup J'$ with only $O(1/\varepsilon \log(1/\varepsilon))$ different widths. Therefore we may assume that $\varepsilon SIZE(L_{wide}) > 2(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}$. Otherwise we can achieve a packing for L_{wide} with height less than $2SIZE(L_{wide}) + h_{\max} \leq {}^4/_\varepsilon(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max} + h_{\max} = \mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$ using NFDH heuristic.

For arbitrary lists L'', L' we define a partial order $\leq_g$, so that $L'' \leq_g L'$, if and only if the stack of L'' can be geometrically included into the one of L'. For a list L of rectangles let $FSP(L)$ denote the height of an optimum fractional strip packing for L. Moreover we have $L_{wide} \leq_g L_{sup}$.

Lemma 4.3. *The rounded instance $J \cup J'$ has the following properties*

(i) The number of different width is bounded by $M = O(1/\varepsilon \log(1/\varepsilon))$.

(ii) $SIZE(L_{wide}) \leq SIZE(J \cup J') \leq (1 + \varepsilon)SIZE(L_{wide})$.

(iii) $FSP(L_{wide}) \leq FSP(J \cup J') \leq (1 + \varepsilon)FSP(L_{wide})$.

Algorithm 6 Rounding II

1: Let $k := \left\lfloor \frac{SIZE(L_{wide})\varepsilon}{(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}} \right\rfloor$
2: **for all** $t \in \{0, 1 \ldots, \lfloor \log(2/\varepsilon) \rfloor\}$ **do**
3: partition L_{wide} into $\lfloor \log(2/\varepsilon) \rfloor + 1$ lists $W_t := \{r_j \in L_{wide} | w_j \in (2^{-(t+1)}, 2^{-t})\}$.
4: For each list W_t order the rectangles by non-increasing widths and pack them left-aligned on a stack.
5: Starting at the baseline draw horizontal lines at height $(ik2^t)h_{\max}$ for $i = \{0, \ldots, \lfloor h(W_t)/(k2^t h_{\max}) \rfloor\}$ through the stack. For every rectangle whose interior is cut by such a line we introduce two new rectangles, so that the stack is divided into $q(t) = \lfloor h(W_t)/(k2^t h_{\max}) \rfloor + 1$ groups $G_1(t), \ldots, G_{q(t)}(t)$ all having total height exactly $k2^t h_{\max}$ except maybe the last group $G_{q(t)}(t)$ of the narrowest rectangles having height $< k2^t h_{\max}$.
6: In each group $G_i(t)$ we round up the width of every rectangle to the width of the widest rectangles contained in this group and obtain $G'_i(t)$.
7: **end for**
8: Define $J_t := \bigcup_{i=2}^{q(t)} G'_i(t)$. Further let $J := \bigcup_t J_t$, $J' := \bigcup_t G'_1(t)$

Proof. We have that $SIZE(W_t) \geq 2^{-(t+1)} h(W_t) \geq 2^{-(t+1)}(q(t)-1)k2^t h_{\max} \geq {kh_{\max}(q(t)-1)}/{2}$ and thus $q(t) \leq \frac{2SIZE(W_t)}{kh_{\max}} + 1$ for all t. With $\varepsilon SIZE(L_{wide}) > 2(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}$ we conclude

$$
\begin{aligned}
M = \sum_{t=0}^{\lfloor \log(2/\varepsilon) \rfloor} q(t) &\leq \sum_{t=0}^{\lfloor \log(2/\varepsilon) \rfloor} \frac{2SIZE(W_t)}{kh_{\max}} + 1 \\
&\leq \frac{2}{kh_{\max}} SIZE(L_{wide}) + \lfloor \log(2/\varepsilon) \rfloor + 1. \\
&\leq \frac{2SIZE(L_{wide})}{h_{\max} \left\lfloor \frac{\varepsilon SIZE(L_{wide})}{(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}} \right\rfloor} + \lfloor \log(2/\varepsilon) \rfloor + 1 \\
&\leq \frac{2SIZE(L_{wide})}{h_{\max} \left(\frac{\varepsilon SIZE(L_{wide})}{(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}} - 1 \right)} + \lfloor \log(2/\varepsilon) \rfloor + 1 \\
&= \frac{2SIZE(L_{wide})(\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max}}{h_{\max}(\varepsilon SIZE(L_{wide}) - (\lfloor \log(2/\varepsilon) \rfloor + 1)h_{\max})} + \lfloor \log(2/\varepsilon) \rfloor + 1 \\
&< \frac{2SIZE(L_{wide})(\lfloor \log(2/\varepsilon) \rfloor + 1)}{\frac{\varepsilon}{2} SIZE(L_{wide})} + \lfloor \log(2/\varepsilon) \rfloor + 1 \\
&= 4/\varepsilon(\lfloor \log(2/\varepsilon) \rfloor + 1) + \lfloor \log(2/\varepsilon) \rfloor + 1 \\
&\leq 5/\varepsilon(\lfloor \log(2/\varepsilon) \rfloor + 1)
\end{aligned}
$$

The total height of the rectangles in every group $G_1(t)$ is at most $k2^t h_{\max}$. Since each of the rectangles in $G_1(t)$ has width at most 2^{-t} we have $SIZE(G_1(t)) \leq kh_{\max}$. Then $SIZE(L_{wide}) \leq SIZE(J \cup J') \leq SIZE(J) + (\lfloor \log(2/\varepsilon) \rfloor + 1)kh_{\max}$, since there are $\lfloor \log(2/\varepsilon) \rfloor + 1$ groups $G_1(t)$ in J'. With our choice of k assertion (ii) follows.

For all $t \in \{0, 1 \ldots, \lfloor \log(2/\varepsilon) \rfloor\}$ we have by construction

$G'_1(t) \geq_g G_1(t) \geq_g G'_2(t) \geq_g \ldots \geq_g G_{q(t)}(t)$ and thus

$$J \cup J' = \bigcup_{i=1}^{q(t)} G'_i(t) \geq_g W_t \geq_g \bigcup_{i=2}^{q(t)} G'_i(t) = J. \tag{4}$$

We have $FSP(J \cup J') \leq FSP(J) + FSP(J')$. Since at least 2^t of the rectangles in $G'_1(t)$ fit next to each other into the strip we conclude also $FSP(G'_1(t)) \leq kh_{\max}$. Thus, we have

$$FSP(J') \leq (\lfloor \log(2/\varepsilon) \rfloor + 1)kh_{\max} \leq \varepsilon SIZE(L_{wide}) \leq \varepsilon FSP(L_{wide}).$$

We conclude $FSP(J \cup J') \leq FSP(J) + \varepsilon FSP(L_{wide}) \leq (1+\varepsilon)FSP(L_{wide})$. □

Algorithm 7 Improved Strip Packing

1: For accuracy $\varepsilon > 0$ partition L into $L_{wide} := \{r_j \in L | w_j > \varepsilon/2\}$ and $L_{narrow} := \{r_j \in L | w_j \leq \varepsilon/2\}$.
2: Round the widths of the rectangles in L_{wide} with Algorithm 5 and obtain $J \cup J'$ with only $M = O(1/\varepsilon \log(1/\varepsilon))$ different widths.
3: Solve the linear program $LP(J \cup J')$.
4: Construct a feasible solution for $J \cup J'$ with height at most $FSP(J \cup J') + 2Mh_{\max}$.
5: Use modified $NFDH$ to pack the rectangles in L_{narrow} into the remaining space and on top of the strips.

With the tuned rounding technique the additive factor of the AFPTAS for strip packing improves:

Theorem 4.4. *Algorithm 7 produces for any accuracy $\varepsilon > 0$ and list of rectangles L produces a packing of the rectangles into N strips of unit width with height less than $(1+\varepsilon)OPT_{SP}(L) + (4M+1)h_{\max}$, where $M = \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$. The running time is polynomial is in $\mathcal{O}(\varepsilon^{-6} + n \log n)$.*

The proof uses similar techniques in [11]. The difference is mainly in converting the fractional solution into an integral one. Techniques used for the converting process can be found later in Theorem 5.5. We can solve the linear program used in the algorithm via max-min resource sharing. Thus, since $M \leq 5/\varepsilon(\lfloor \log(2/\varepsilon) \rfloor + 1)$ and according to [11] the fractional stip packing can be solved in time

$$\mathcal{O}(M(1/\varepsilon^2 + \log(M)) \max\{M + 1/\varepsilon^3, M \log\log(M/\varepsilon)\}) = \mathcal{O}(\varepsilon^{-6} \log(1/\varepsilon)).$$

This gives a total running time of $\mathcal{O}(\varepsilon^{-6} \log(1/\varepsilon) + n \log n)$. This also improves the original algorithm with $\mathcal{O}(1/\varepsilon^2)$ different widths and running time $\mathcal{O}(\varepsilon^{-7} + n \log n)$.

5 An AFPTAS for MSP

In this section we present an AFPTAS for MSP. Interestingly, this result also applies to scheduling parallel jobs in identical platforms. The algorithm is a generalization of our improved version of the AFPTAS for strip packing by Kenyon and Rémila [14]. We show

that there exists an algorithm for multiple strip packing achieving the same asymptotic approximation ratio with additive term $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$ for an arbitrary number N of strips. For instances with N sufficient large, namely $N \in \Omega(1/\varepsilon^2 \log(1/\varepsilon))$, we show the existence of an algorithm that adopts an improved additive constant of $\mathcal{O}(1)$.

As in Section 2 by dividing a packing for one strip into N parts of nearly the same height and distributing them among N strips we get an algorithm for MSP. We define an Algorithm A_ε as follows:

Algorithm 8 A_ε

1: Pack the sorted rectangles with Algorithm 7 into a single strip S.
2: **for all** $\ell \in \{0, 1, \ldots, N\}$ **do**
3: Draw a horizontal line through S at height $\ell A_\varepsilon^{KR}(L)/N$.
4: **end for**
5: **for all** $\ell \in \{0, 1, \ldots, N-1\}$ **do**
6: Pack all items intersecting the ℓth line and all items between the ℓth and $(\ell+1)$th lines into strip $S_{\ell+1}$.
7: **end for**

Theorem 5.1. *For a list $L = \{r_1, \ldots, r_n\}$ of rectangles with widths and heights ≤ 1 and accuracy $\varepsilon > 0$ Algorithm A_ε generates a packing into $N \geq 2$ strips with height less than $(1+\varepsilon)OPT_{MSP}(L) + O(1/\varepsilon \log(1/\varepsilon))h_{\max}$ and running time $\mathcal{O}(\varepsilon^{-6} \log(1/\varepsilon) + n \log n)$.*

Proof. By Step 2 every strip S_ℓ, $\ell \in \{1, \ldots, N\}$ has height

$$\begin{aligned} h_i &\leq \frac{A_\varepsilon^{KR}(L)}{N} + h_{\max} \\ &\leq \frac{(1+\varepsilon)OPT_{SP}(L) + (4M+1)h_{\max}}{N} + h_{\max} \\ &\overset{N\geq 2}{\leq} \frac{(1+\varepsilon)OPT_{SP}(L)}{N} + (2M+1)h_{\max} + h_{\max} \\ &\leq (1+\varepsilon)OPT_{MSP}(L) + (2M+2)h_{\max}, \end{aligned}$$

where the last inequality holds because $OPT_{SP}(L)/N$ is a lower bound for $OPT_{MSP}(L)$. The running time is dominated by the time used for Algorithm 7. □

5.1 Instances with a large number of strips

Now let us consider instances with a large number of strips. In this case it is possible to improve the additive constant to $\mathcal{O}(1)$ by balancing the configurations. Choose $\delta := \frac{\varepsilon}{4+\varepsilon}$ and $N > 10/\delta^2(\lfloor \log(2/\delta) \rfloor + 1) = \Omega(1/\varepsilon^2 \log(1/\varepsilon))$. We divide the list of rectangles L into a list of narrow rectangles $L_{narrow} := \{r_j \in L | w_j \leq \delta/2\}$ and a list of wide rectangles $L_{wide} := \{r_j \in L | w_j > \delta/2\}$. Then we round L_{wide} to an instance $J \cup J'$ with $M \leq 5/\delta(\lfloor \log(2/\delta) \rfloor + 1) = \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$ (compare Lemma 4.3) different widths using Algorithm 6. Our first objective is to create a fractional packing for the rounded wide rectangles into N strips. To do this we solve approximately (see Section 6.2) the linear program (3) for the rounded instance of wide rectangles ($J \cup J'$). Let $h_0 := FSP(J \cup J')/N$.

We first show there is a fractional packing for $J \cup J'$ that contains only $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))$ different configurations and has height at most $(1+\delta)h_0$.

Lemma 5.2. *Let $x = (x_1, \ldots, x_q)$ be a solution of $LP(J \cup J')$ with at most $m \leq M$ nonzero entries $x_1, \ldots, x_m$. For $N > {}^{10}/_{\delta^2}(\lfloor \log(2/\delta) \rfloor + 1)$ we get a fractional packing into N strips with height at most $(1+\delta)h_0$ and at most $m' \leq 2M$ different configurations.*

Proof. We first pack the rectangles fractionally into the configurations. Imagine each configuration C_j as a bin with height x_j and width c_j and divide it into α_{ij} columns of widths w_i and height x_j. Pack the rectangles in $J \cup J'$ of width w_i in a Greedy manner fractionally into the columns of width w_i until exactly height x_j, starting with $i = 1$. In this way each column contains a sequence of rectangles, which completely fit inside the column and possibly the top part of a rectangle, that started in a previous column and the bottom part of a rectangle, that is too tall to fit into this column. Since $\sum_{j=1}^{m} \alpha_{ij} x_j \geq \beta_i$, the total height of the rectangles of width w_i, there will be maybe more than enough space for the rectangles of width w_i in the configurations. In this case we distribute the rectangles among the columns and delete the additional space. So we split a configuration C_j into two parts, one of the old type where the columns of width w_i are completely filled and one without columns of width w_i. This case may happen only M times. So we have in total $m' = m + M \leq 2M$ configurations $C_1, \ldots C_{m'}$ with nonzero heights $x_1, \ldots, x_{m'}$. Assume w.l.o.g. $w_1 \geq w_2 \geq \ldots \geq w_M$. We start filling the columns for each width with the rounded fraction of the wide rectangle in L_{wide} that is split into a rectangle of width w_i and the next wider width w_{i-1} by the rounding Algorithm 5, if such a fraction exists. We finish packing each width w_i with the fraction of the wide rectangle in L_{wide} that is split into a rectangle of width w_i and the next smaller width w_{i+1}, if such a fraction exists. Notice that there exist configurations with height larger or equal h_0, since if not we conclude $\sum_{j=1}^{m'} x_j < m' h_0 \leq 2M h_0 = {}^{10}/_{\delta}(\lfloor \log(2/\delta) \rfloor + 1)h_0 < N h_0$, which is a contradiction. Consider a configuration C_j, $j \in \{1, \ldots, m'\}$. If $x_j \geq h_0$ we allocate $\lfloor {}^{x_j}/_{h_0} \rfloor$ empty strips with height h_0 for C_j. If then ${}^{x_j}/_{h_0} - \lfloor {}^{x_j}/_{h_0} \rfloor \leq \delta h_0$, we assign to C_j additional space with height $({}^{x_j}/_{h_0} - \lfloor {}^{x_j}/_{h_0} \rfloor)$ in a strip, that has already height h_0. If ${}^{x_j}/_{h_0} - \lfloor {}^{x_j}/_{h_0} \rfloor > \delta h_0$, we divide $({}^{x_j}/_{h_0} - \lfloor {}^{x_j}/_{h_0} \rfloor)$ into at most ${}^{1}/_{\delta}$ stripes with height less or equal δh_0. So assign to C_j additional space of height δh_0 in no more than ${}^{1}/_{\delta}$ strips, which are already occupied until height h_0. In the same way as the remaining stripes we handle configurations of height less than h_0. Since there are at most $2M$ configurations with nonzero height, we get at most ${}^{2M}/_{\delta} = {}^{10}/_{\delta^2}(\lfloor \log(2/\delta) \rfloor + 1) < N$ additional assignments of height δh_0, which can be distributed to N strips. Thus by this assignment policy, where the configurations are balanced, each strip has allocated area of height at most $(1+\delta)h_0$ for at most 2 different configurations □

The next Lemma shows how to get from a fractional packing to a feasible integral packing.

Lemma 5.3. *Let $x = (x_1, \ldots, x_q)$ be a solution of $LP(J \cup J')$ with at most $m' \leq 2M$ nonzero entries $x_1, \ldots, x_{m'}$. For $N > {}^{10}/_{\delta^2}(\lfloor \log(2/\delta) \rfloor + 1)$ we can convert x to a feasible packing for the rectangles in $J \cup J'$ and with height at most $(1+\delta)h_0 + 2h_{\max}$ and at most 2 different configurations per strip.*

Proof. Consider a strip S_ℓ. Emanating from the above fractional solution with $m' \leq 2M$ completely filled configurations we have in S_ℓ at most 2 configurations. Between height zero and h_0 we have a configuration C_j of height $x_{j_\ell} = h_0$ and between height h_0 and $(1+\delta)h_0$ we have a configuration C_k of height $x_{k_\ell} = \delta h_0$. First we assign to each configuration additional space of height $2h_{\max}$. This increases the total height of the packing by $2h_{\max}$. Consider now the fractional packing in the proof of Lemma 5.2. We convert this packing to a feasible integral packing for the rectangles in $J \cup J'$ in the following way:
Consider the columns of width w_i starting in the first strip where w_i appears. In each column, but the last one for w_i there are a sequence of rectangles and possibly two fractional rectangles. One is the top part of a rectangle, which does not fit completely into the previous column and the other one is the bottom part of a rectangle that is too tall to fit into this column. In each column we pack all rectangles belonging to the sequence and the completed fractional rectangle from the top. If the considered column is the last one for width w_i we have only one fractional rectangle at the bottom and possibly a rectangle that is a rounded fraction of a rectangle in L_{wide} that was split by rounding Algorithm 6 between width w_i and the next smaller width w_{i+1}. Here we add the fraction of width w_{i+1} belonging to the same rectangle in L_{wide}.

So we can guarantee that each configuration C_j with height x_{j_ℓ} in strip S_ℓ is filled up to height at least $x_{j_\ell} - h_{\max}$ with rectangles of $J \cup J'$. □

Since we can guarantee that there are at most 2 different configurations per strip, the additive constant will be improved to $\mathcal{O}(1)h_{\max}$, while the running time remains the same. For a smaller value of N the balancing argument for the configurations can not be applied. For $N = 1$ we can guarantee a feasible packing for the wide rectangles with height at most $h_0 + 2Mh_{\max}$.

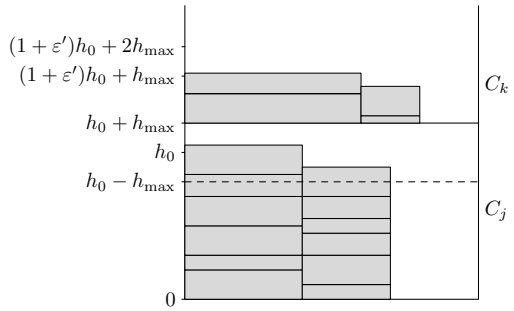

Figure 2: S_ℓ with C_j and C_k.

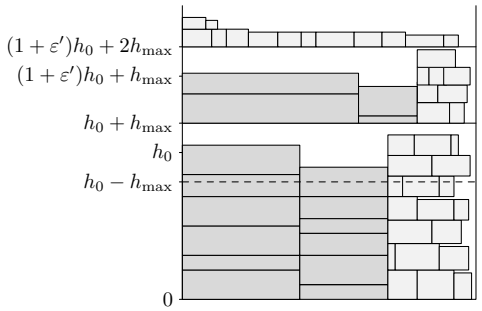

Figure 3: S_ℓ after packing the narrow rectangles.

Our last step is to pack the narrow rectangles. We use a modified version of the NFDH algorithm: For strip S_ℓ as above we pack narrow rectangles with NFDH into the empty space next to the configurations until the total height is at most $(1+\delta)h_0 + 2h_{\max}$. After that we repeat the process for strip $S_{\ell+1}$. When all strips are filled in this way, we draw a horizontal line at height $(1+\delta)h_0 + 2h_{\max}$ in each strip and pack the remaining narrow rectangles with Algorithm 2 on top (see Fig 2 and 3). Thus we can ensure by Lemma 2.4 that the maximum difference of the heights of two arbitrary strips is at most $h_{\max}$ (see Fig 3). Let h_{final} denote the height of the packing after adding the narrow rectangles.

Lemma 5.4. *Let $N > {}^{10}/\delta^2(\lfloor \log(2/\delta) \rfloor + 1)$. If $h_{final} \geq (1+\delta)h_0 + 2h_{\max}$, then we have*

$$h_{final} \leq \frac{SIZE(J \cup J' \cup L_{narrow})}{N(1-\delta/2)} + \delta h_0 + 6h_{\max}.$$

Proof. We consider the strip $S_{\min}$ with $SIZE(S_{\min}) = \min_{\ell \in \{1,\ldots,N\}} SIZE(S_\ell)$ and with configurations C_j and C_k with heights $x_{j_\ell} = h_0$ and $x_{k_\ell} \leq \delta h_0$ and widths c_j, c_k, respectively. Let $a_1 < \ldots < a_r$ be the ordered sequence of shelves constructed by modified $NFDH$ in $S_{\min}$, such that $a_k \geq (1+\delta)h_0 + 2h_{\max}$ or $0 \leq a_k \leq h_0 - h_{\max}$ for all $k \in \{1,\ldots,r\}$. Let $a_{s_1} < \ldots < a_{s_{r'}}$ be the subsequence of shelves with at least one rectangle. Furthermore, let b_{s_i} and b'_{s_i} be the heights of the first and the last rectangle placed in shelf $a_{s_i}, i \in \{1,\ldots,r'\}$. A shelf is closed, when the next narrow rectangle does not fit completely on the shelf. Notice that all narrow rectangles on shelf a_{s_i} have height $\geq b'_{s_i}$. Let $a_{\bar{s}_1} < \ldots < a_{\bar{s}_{\bar{r}}}$ be the subsequence of $a_{s_1} < \ldots < a_{s'_r}$ such that either $a_{\bar{s}_i} \geq (1+\delta)h_0 + 2h_{\max}$ or $0 \leq a_{\bar{s}_i} + b'_{\bar{s}_i} \leq h_0 - h_{\max}$. Keep in mind that the region $[0, c_j] \times [0, x_{j_\ell} - h_{\max}]$ of $S_{\min}$ is completely filled with rectangles from $J \cup J'$. We consider three cases for shelf a_k with $k < r$:

Case 1: On shelf a_k is at least one narrow rectangle and $a_k \geq (1+\delta)h_0 + 2h_{\max}$ or $0 \leq a_k + b'_k \leq h_0 - h_{\max}$. In this case there exists $i \in \{1,\ldots,\bar{r}-1\}$ with $k = \bar{s}_i$. Therefore, an area of at least $b'_{\bar{s}_i}(1-\delta/2)$ is covered by wide and narrow rectangles from $J \cup J' \cup L_{narrow}$.

Case 2: On shelf a_k is at least one narrow rectangle with $a_k + b'_k > h_0 - h_{\max}$ and $a_k \leq h_0 - h_{\max}$. In this case $(h_0 - h_{\max} - a_k)(1-\delta/2)$ is covered by wide and narrow rectangles.

Case 3: On shelf a_k is no narrow rectangle and $a_k \leq h_0 - h_{\max}$. This case may happen, when the wide rectangles in configuration C_j have already a total width larger than $1-\varepsilon'$. In this case an area of at least $(h_0 - h_{\max} - a_k)(1-\delta/2)$ is covered by wide rectangles in $J \cup J'$. Cases 2 and 3 may happen only once per strip.

Let

$$X := \sum_{k \notin \{\bar{s}_1,\ldots,\bar{s}_{\bar{r}-1}\}} (h_0 - h_{\max} - a_k)\,.$$

Then $(1-\delta/2)(X + \sum_{i=1}^{\bar{r}-1} b'_{\bar{s}_i})$ is a lower bound for the total size $SIZE(S_{\min})$ which itself is bounded from above by ${}^1/_N SIZE(J \cup J' \cup L_{narrow})$. On the other hand, the height of the final packing is at most $X + \sum_{i=1}^{\bar{r}} b_{\bar{s}_i} + 4h_{\max} + \delta h_0$ plus $h_{\max}$ by Lemma 2.4 by packing the rectangles on top. This gives

$$\begin{aligned}
h_{final} &\leq X + \sum_{i=1}^{\bar{r}} b_{\bar{s}_i} + \delta h_0 + 5h_{\max} \\
&\leq X + \sum_{i=1}^{\bar{r}-1} b_{\bar{s}_i} + \delta h_0 + 6h_{\max} \\
&\leq \frac{SIZE(S_{\min})}{1-\delta/2} + \delta h_0 + 6h_{\max} \\
&\leq \frac{SIZE(J \cup J' \cup L_{narrow})}{N(1-\delta/2)} + \delta h_0 + 6h_{\max}.
\end{aligned}$$

□

Theorem 5.5. *If $N > {}^{10}/_{\delta^2}(\lfloor \log(2/\delta) \rfloor + 1)$ the Algorithm 9 generates for an instance L of MSP a packing of height at most $(1+\varepsilon)OPT_{MSP}(L) + \mathcal{O}(1)$ and running time $\mathcal{O}(\varepsilon^{-6}\log(1/\varepsilon) + n\log(n))$.*

Proof. If $h_{final} > (1+\delta)h_0 + 2h_{\max}$ we conclude with Lemma 5.4

$$\begin{aligned}
h_{final} &< \frac{SIZE(J \cup J') + SIZE(L_{narrow})}{N(1-\delta/2)} + \frac{\delta FSP(J \cup J')}{N} + 6h_{\max} \\
&\overset{\text{L } 4.3}{\leq} \frac{(1+\delta)SIZE(L_{wide}) + SIZE(L_{narrow})}{N(1-\delta/2')} + \frac{\delta(1+\delta)FSP(L_{wide})}{N} + 6h_{\max} \\
&\leq \frac{1+\delta}{1-\delta/2} OPT_{MSP}(L) + \delta(1+\delta)OPT_{MSP}(L) + 6h_{\max} \\
&\leq \frac{1+3\delta}{1-\delta} OPT_{MSP}(L) + 6h_{\max} \\
&= (1+\varepsilon)OPT_{MSP}(L) + 6h_{\max},
\end{aligned}$$

where the third inequality holds because $SIZE(L)/N$ and $FSP(L_{wide})/N$ are lower bounds for $OPT_{MSP}(L)$. On the other hand, it follows immediately from 5.3 and 4.3 that

$$\begin{aligned}
h_{final} &\leq (1+\delta)\frac{FSP(J \cup J')}{N} + 2h_{\max} \\
&\leq \frac{(1+\delta)^2 FSP(L_{wide})}{N} + 2h_{\max} \\
&\leq (1+\varepsilon)OPT_{MSP}(L) + 2h_{\max}.
\end{aligned}$$

□

Algorithm 9

1: Set $\delta := {}^{\varepsilon}/_{(4+\varepsilon)}$
2: Partition L into $L_{wide} := \{r_j \in L | w_j > \delta/2\}$ and $L_{narrow} := \{r_j \in L | w_j \leq \delta/2\}$.
3: Round the widths of the rectangles in L_{wide} with Algorithm 5 and obtain $J \cup J'$ with only $M = \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$ different widths.
4: Solve the linear program $LP(J \cup J')$.
5: Construct a feasible solution for $J \cup J'$ by balancing the configurations.
6: Use modified $NFDH$ to pack the rectangles in L_{narrow} into the remaining space and on top of the strips.

Corollary 5.6. *There is an algorithm for scheduling parallel jobs in identical platforms that for any input J of SPP and $\varepsilon > 0$ produces a schedule of length at most*

$$(1+\varepsilon)OPT_{SPP}(J) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}$$

adn running time $\mathcal{O}(\varepsilon^{-6}\log(1/\varepsilon) + n\log(n))$. For $N = \Omega(1/\varepsilon^2 \log(1/\varepsilon))$ the length of the schedule improves to

$$(1+\varepsilon)OPT_{SPP}(J) + \mathcal{O}(1)p_{\max}.$$

6 An AFPTAS for *SPP*

The algorithm for the more general scheduling problem is based on an LP-relaxation. This allows migration and preemption of jobs. That is a job is allowed to be split into fractions that are executed in different platforms (if they fit). Emanating from the solution of the LP we compute a unique assignment of almost all jobs to the platforms. This is done by grouping the jobs similar as in Algorithm 6 and then rounding the fractions of jobs using a result of Lenstra et al. [16] ; i.e. the number of remaining fractional jobs per platform will be bounded by $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))$. Remarkably, the rounding technique needs except an (approximate) solution of the LP no extra information about the speed values. For each platform we reschedule the obtained integral jobs with Algorithm 7, our improved version of for strip packing of Kenyon and Rémila. Finally, the fractional jobs are scheduled behind. An overview of the algorithm is given below Algorithm 10.

Algorithm 10

1: Solve a linear program relaxation of the problem (5) and get a fractional schedule where preemption and migration are allowed.
2: Group the fractional jobs corresponding to the LP-solution as described in Algorithm 11 according their widths and for every platform P_ℓ obtain sets L^ℓ_{wide} and L^ℓ_{narrow} of wide and narrow fractional rectangles, respectively.
3: Via a general assignment problem (10) round the fractional rectangles and obtain sets of rounded rectangles $\tilde{L}^\ell_{wide}$, $\tilde{L}^\ell_{narrow}$ and fractional rectangles F^ℓ for $\ell \in \{1, \ldots, N\}$.
4: **for all** $\ell \in \{1, \ldots, N\}$ **do**
5: Pack $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ with the strip packing subroutine 7 into platform P_ℓ.
6: Schedule the fractional jobs in F^ℓ greedily on top of the schedule corresponding to the packing obtained before.
7: **end for**

6.1 Relaxed Schedule

Let J be an instance of SPP and let T be the makespan of an optimum schedule for J. To simplify the structure of the schedule instead of handling the specific processing times t^ℓ_j we consider each platform as a two-dimensional bin of width m_ℓ and height Ts_ℓ and schedule the jobs concerning their lengths p_j within this bin. Furthermore, we abandon the constraint that a job has to be scheduled non-preemptively and within only one platform. We represent the schedule of a job $J_j = (p_j, q_j)$ as a (finite) sequence of pairs $(I_i, Q_i)_{i \in I(j)}$, $I(j) \subset \mathbb{N}$, where every $I_i \subset [0, T]$ is a time interval and every Q_i is a set of processors so that there is a uniquely defined platform $P_{\ell_i} \in \{1, \ldots, N\}$ with $Q_i \subset P_{\ell_i}$ and $|Q_i| = q_j$. Additionally, we assume that the following conditions hold:

(i) the time intervals for job J_j within the same platform do not overlap except maybe at the endpoints, i.e. for all $\ell \in \{1, \ldots, N\}$
$\bigcup_{\substack{i,i' \in I(j), i \neq i' \\ \ell_i = \ell = \ell_{i'}}} \left(\overset{\circ}{I}_i \cap \overset{\circ}{I}_{i'} \right) = \emptyset$, where $\overset{\circ}{A}$ denotes the interior of a set A.

(ii) $\sum_{\ell=1}^{N} s_\ell \sum_{\{i \in I(j) | Q_i \subset P_\ell\}} |I_i| \geq p_j$ (covering constraint).

(iii) at any time for every processor there is at most one job running on it.

Keep in mind that under this constraints a job is allowed to be split among the platforms and may be executed in two different platforms at the same time, but never in parallel with itself within the same platform (except for a discrete time, when one piece starts and another ends). It can be executed on two different (not necessary disjoint) subsets of processors within the same platform during different time intervals, where only the endpoints of the time intervals may overlap. An example how such a relaxed schedule can look like is given in Figure 4: Assume that $T = 10/s_{\ell_1}$ and job J_j needs to be scheduled on $q_j = 3$ processors for $p_j = 7.5$ operations. So in P_{ℓ_1} it is scheduled on processors $\{7, 8, 9\}$ during time $[0, 1/s_{\ell_1}]$ and on processors $\{2, 3, 4\}$ during time $[5/s_{\ell_1}, 7/s_{\ell_1}]$. In P_{ℓ_2} it is scheduled on processors $\{1, 2, 3\}$ during time $[0, 3/s_{\ell_2}]$ and in P_{ℓ_3} it is scheduled on processors $\{3, 4, 5\}$ during time $[3.5/s_{\ell_3}, 5/s_{\ell_3}]$. This gives $1 + 2 = 3$ operations in P_{ℓ_1}, 3 operations in P_{ℓ_2} and 1.5 operations in P_{ℓ_3} (this fulfills the covering constraint).

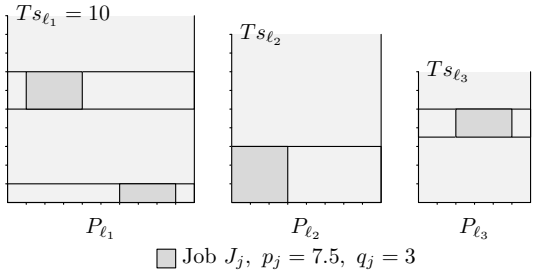

Figure 4: Relaxed schedule

The relaxed schedule can be formulated via the linear program below: For each platform in P_ℓ, $1 \leq \ell \leq N$ we introduce configurations C^ℓ. A configuration C^ℓ is a function $C^\ell : \{1, \ldots, n\} \longrightarrow \{0, 1\}$, so that $\sum_{\{j \in \{1,\ldots,n\} | C^\ell(j)=1\}} q_j \leq m_\ell$. It tells us which jobs can be scheduled in parallel in platform P_ℓ. By definition, the number $q(\ell)$ of different configurations for P_ℓ is bounded by 2^n. Let $\mathcal{C}^\ell = \{C^\ell_1, \ldots, C^\ell_{q(\ell)}\}$ denote the set of all configurations for a platform P_ℓ. In the LP below the variable $x_{C^\ell_k}$ indicates the length of configuration C^ℓ_k. That means that the jobs in $\{j \in \{1, \ldots, n\} | C^\ell_k(j) = 1\}$ are executed in platform P_ℓ during $x_{C^\ell_k}$ operation steps.

$$\begin{aligned} \sum_{k=1}^{q(\ell)} x_{C^\ell_k} &= s_\ell T \quad \ell \in \{1, \ldots, N\} \\ \sum_{\ell=1}^{N} \sum_{\{k \in \{1,\ldots,q(\ell)\} | C^\ell_k(j)=1\}} x_{C^\ell_k} &\geq p_j \; j \in \{1, \ldots, n\} \\ x_{C^\ell_k} &\geq 0 \quad k \in \{1, \ldots, q(\ell)\}, \; \ell \in \{1, \ldots, N\} \end{aligned} \tag{5}$$

The first N constraints ensure that the makespan $C_{\max}(\ell)$ in each platform P_ℓ does not exceed T. The next n constraints are covering constraints for the n jobs. They make sure that every job is executed sufficiently long.

Lemma 6.1. *If T is the makespan of an optimum schedule for* SPP(J)*, the linear program above (5) is a relaxation of* SPP(J).

Proof. Consider an optimum solution with makespan T for $SPP(J)$. Then we construct a solution for LP (5) in the following way:
For each platform P_ℓ we consider the finishing and starting times $t_1, \ldots, t_{2\tilde{n}(\ell)} \in [0, T]$ of the $\tilde{n}(\ell) \leq n$ jobs assigned to this platform. They give a partition of $[0, T]$ (not minding empty intervals). We can assume w.l.o.g. $0 = t_1 \leq \ldots \leq t_{2\tilde{n}(\ell)}$ and denote $t_{2\tilde{n}(\ell)+1} = T$. For each non-empty interval $[t_i, t_{i+1}]$, $i \in \{1, \ldots, 2\tilde{n}(\ell)\}$, we store a vector $v_i^\ell \in \{0,1\}^n$ and set $v_i^\ell(j) = 1$ if job J_j is scheduled in platform P_ℓ during this interval, else $v_i^\ell(j) = 0$. Furthermore we store a value $x_i^\ell := s_\ell(t_{i+1} - t_i)$ for v_i^ℓ. Note that if $t_{2\tilde{n}(\ell)} < T$ the vector $v_{2\tilde{n}(\ell)}^\ell$ is equal to $0_{\{0,1\}^n}$. Note that we store at most $2n$ vectors for every platform, since for $\tilde{n}(\ell) = n$ we conclude $t_{2n} = T$. Let V^ℓ denote the set of stored vectors. As long as there are two identical vectors $v_i^\ell = v_j^\ell$ with $i \neq j$ (w.l.o.g. $i < j$) in V^ℓ we reset $x_i^\ell = x_i^\ell + x_j^\ell$ and discard v_j^ℓ. Finally we define for every $\ell \in \mathbb{N}$ and $1 \leq k \leq q(\ell)$

$$\bar{x}_{C_k^\ell} := \begin{cases} x_i^\ell \text{ if } C_k^\ell = v_i^\ell \text{ for } i \in 1, \ldots, 2\tilde{n}(\ell) \\ 0 \; else. \end{cases}$$

We claim that $\bar{x} := (\bar{x}_{C_k^\ell})_{k \in \{1,\ldots,q(\ell)\}, \ell \in \{1,\ldots,N\}}$ is a solution of (5). Clearly, we have that $\bar{x}_{C_k^\ell} \geq 0$ and $\sum_{k=1}^{q(\ell)} \bar{x}_{C_k^\ell} = s_\ell T$ for $\ell \in \{1, \ldots, N\}$, since it is the sum of the values $s_\ell(t_{i+1} - t_i)$, $i \in \{1, \ldots, 2\tilde{n}(\ell)\}$. Since we started with a feasible schedule, by definition of $\bar{x}$ we already have $\sum_{\{k \in \{1,\ldots,q(\ell)\} | C_k^\ell(j)=1\}} \bar{x}_{C_k^\ell} = \sum_{\{v_i^\ell \in V^\ell | v_i^\ell(j)=1\}} x_i^\ell = p_j$ for all jobs assigned to platform P_ℓ, so the covering constraint in (5) is fulfilled. □

6.2 Solving the LP

The problem above is a feasibility problem with an exponential number of variables and $N + n$ constraints. Let $PS(J)$ denote the makespan of an optimal preemptive schedule with the properties as described above. Clearly, $PS(J) \leq \text{OPT}_{SPP}(J)$. It is sufficient to solve the *LP* approximately obtaining a value T with $PS(J) \leq T \leq (1+\varepsilon)PS(J) \leq (1+\varepsilon)\text{OPT}_{SPP}(J)$. Let $d_{\min} := \frac{\max p_j}{\max s_\ell}$. Since in our LP-relaxation a job can be scheduled in parallel with itself, when scheduled in different platforms we obtain a lower bound $L := {}^{d_{\min}}/_N$ and an upper bound $U := nt_{\max}$ for $PS(J)$. Now there must be a value $L + j\varepsilon L \in [L, U]$ for $j \in \left\{0, \ldots \left\lfloor \frac{U-L}{\varepsilon L} \right\rfloor\right\}$, so that $PS(J) \in [L + j\varepsilon L, L + (j+1)\varepsilon L]$. Thus we have $PS(j) \leq L + (j+1)\varepsilon L \leq (1+\varepsilon)PS(J)$. Consequently, using binary search to find a proper value T takes time in $\mathcal{O}(\log(\frac{U-L}{\varepsilon L})) = \mathcal{O}(\log(nt_{\max}N(\varepsilon d_{\min})^{-1}))$. Since $t_{\max} \leq \frac{\max p_j}{\min s_\ell}$, we have $t_{\max}/d_{\min} \leq \max s_\ell$ (remember that $\min s_\ell = 1$) and therefore we find T in time $\mathcal{O}(\log(nN\varepsilon^{-1} \max s_\ell))$.

We can now formulate the problem described in (5) as a max-min resource sharing problem and solve (5) by using binary search on the optimum value $PS(J)$ and testing in each step the feasibility of a system of (in-)equalities for a given $T \in [L, U]$. Let

$(x_{C_k^\ell}) := (x_{C_k^\ell})_{k\in\{1,\dots,q(\ell)\},\ell\in\{1,\dots,N\}}$. The system of inequalities is given as

$$\sum_{\ell=1}^{N}\frac{1}{p_j}\sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}} x_{C_k^\ell}\geq 1,$$

for $j\in\{1,\dots,n\}$, $x\in B(T)$, where

$$B(T) := \left\{(x_{C_k^\ell})|x_{C_k^\ell}\geq 0, \sum_{k=1}^{q(\ell)} x_{C_k^\ell} = s_\ell T,\ \ell\in\{1,\dots,N\}\right\}.$$

We test the feasibility of the system by computing an approximate solution for

$$\lambda^\star = \max\{\lambda|\sum_{\ell=1}^{N}\frac{1}{p_j}\sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}} x_{C_k^\ell}\geq\lambda, \quad 1\leq j\leq n,\ x\in B(T)\}. \tag{6}$$

This problem can be considered as a fractional covering problem with convex set $B(T)$ and n covering constraints. We present the n covering constraints by $Ax\geq\lambda$. According to [8] we can compute an $(1-\delta)$-approximate solution for (6) in $\mathcal{O}(n(1/\delta^2+\ln n))$ iterations, where an iteration includes roughly summarized the following steps:
We start with an initial solution $\bar{x}\in B(T)$ and compute a certain price vector $y=y(\bar{x})\in\mathbb{R}_+^n$ depending on $\bar{x}$. Then we consider the so called block problem, Max $\{y^TAx|x\in B(T)\}$ and compute a $(1-\delta')$-approximate solution $\tilde{x}\in B(T)$ for it with $\delta'=\delta/6$ (for details see the next paragraph). Set $\bar{x} := (1-\tau)\bar{x}+\tau\tilde{x}$ for a certain $\tau\in(0,1)$. The algorithm stops after $\mathcal{O}(n(1/\delta^2+\ln n))$ iterations with a vector $\bar{x}$ so that

$$\begin{aligned}\sum_{\ell=1}^{N}\frac{1}{p_j}\sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}}\bar{x}_{C_k^\ell}&\geq(1-\delta)\lambda^\star\\ &\qquad j\in\{1,\dots,n\}\\ \sum_{k=1}^{q(\ell)}\bar{x}_{C_k^\ell}&=Ts_\ell\ \ell\in\{1,\dots,N\}\\ \bar{x}_{C_k^\ell}\geq 0\ k\in\{1,\dots,q(\ell)\},\ \ell&\in\{1,\dots,N\}\end{aligned}$$

After $\mathcal{O}(n(1/\delta^2+\ln n))$ iterations for a given value $T\in[L,U]$ we either

1) get a solution $\bar{x}\in B(T)$ such that for all $j\in\{1,\dots,n\}$ we have $\sum_{\ell=1}^{N}\frac{1}{p_j}\sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}}\bar{x}_{C_k^\ell}\geq(1-\delta)$

2) or conclude that there is no solution $\bar{x}$ with $\sum_{\ell=1}^{N}\frac{1}{p_j}\sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}}\bar{x}_{C_k^\ell}\geq 1$ for $\ell\in\{1,\dots,N\}$.

For $x \in B(T)$ define $\lambda(x) := \min_j \sum_{\ell=1}^{N} \frac{1}{p_j} \sum_{\{k \in \{1,\ldots,q(\ell)\} | C_k^\ell(j)=1\}} x_{C_k^\ell}$. If $\lambda^\star \geq 1$ we have $\lambda(\bar{x}) \geq (1-\delta)$, so in case 1) we select a smaller value for T. In case 2) we have $\lambda(\bar{x}) < 1-\delta$ and therefore $\lambda^\star < 1$. So we know that the value T was chosen too small and chose a larger one.

Solving the Block Problem. Consider the block problem $\text{Max}\{y^T Ax | x \in B(T)\}$. The set $B(T)$ can be written as a Cartesian product of N convex sets

$$B_\ell(T) := \left\{ (x_{C_k^\ell})_{k \in \{1,\ldots,q(\ell)\}} | \sum_{k=1}^{q(\ell)} x_{C_k^\ell} = s_\ell T,\ x_{C_k^\ell} \geq 0 \right\}.$$

Each set $B_\ell(T)$ is a simplex. So the block problem can be re-written as

$$\begin{aligned} \text{Max} \sum_{\ell=1}^{N} \sum_{j=1}^{n} \frac{y_j}{p_j} \sum_{\{k \in \{1,\ldots,q(\ell)\} | C_k^\ell(j)=1\}} x_{C_k^\ell}, \\ x \in \bigtimes_{\ell=1}^{N} B_\ell(T). \end{aligned} \tag{7}$$

Thus, it is sufficient to solve N independent smaller block problems of the form

$$\begin{aligned} \text{Max} \sum_{j=1}^{n} \frac{y_j}{p_j} \sum_{\{k \in \{1,\ldots,q(\ell)\} | C_k^\ell(j)=1\}} x_{C_k^\ell}, \\ (x_{C_k^\ell})_{k \in \{1,\ldots,q(\ell)\}} \in B_\ell(T). \end{aligned} \tag{8}$$

For each $\ell \in \{1, \ldots, N\}$ we find the optimum of (8) at a vertex $\tilde{x}$ of B_ℓ(T). Such a vertex corresponds to a configuration $C_{\tilde{k}}^\ell$ with $\tilde{x}_{C_{\tilde{k}}^\ell} = s_\ell T$ and $\tilde{x}_{C_k^\ell} = 0$ for $C_k^\ell \neq C_{\tilde{k}}^\ell$. Thus, we have to find a configuration $C_{\tilde{k}}^\ell$ of jobs that can be executed in parallel in P_ℓ with largest profit, where the profit value of a configuration C_k^ℓ is given as $\sum_{\{j \in \{1,\ldots,n\} | C_k^\ell(j)=1\}} \frac{y_j}{p_j}$. This results in the following knapsack problem:

$$\begin{aligned} \text{Max} \sum_{j=1}^{n} \frac{y_j}{p_j} x_j \\ \text{s.t} \sum_{j=1}^{n} q_j x_j \leq m_\ell \\ x_j \in \{0,1\} \end{aligned}$$

Lawler showed in [15] that a $(1-\delta')$-approximate solution for this knapsack problem can be computed in time $\mathcal{O}(n \log(1/\delta') + 1/\delta'^4)$. Summing up all near optimal solutions for (8), $\ell = 1, \ldots, N$, gives a $(1-\delta')$-approximate solution for (7).

Constructing a Schedule. Now we construct a schedule based on an approximate solution. Choose $\delta := \frac{3\varepsilon}{20}$ and assume $\varepsilon \leq 1$. By binary search on T we can achieve a

solution $\tilde{x}$ so that for every $\ell \in \{1,\dots,N\}$ we have $\sum_{k=1}^{q(\ell)} \tilde{x}_{C_k^\ell} = Ts_\ell \le (1+\varepsilon)s_\ell PS(J)$ and $\sum_{\ell=1}^{N} \frac{1}{p_j} \sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}} \tilde{x}_{C^\ell} \ge (1-\delta)$ for all $j \in \{1,\dots,n\}$. We slightly extend the length of each configuration setting $x_{C_k^\ell} := \tilde{x}_{C_k^\ell}(1+4\delta)$ and conclude

$$\begin{aligned}\sum_{\ell=1}^{N} \frac{1}{p_j} \sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}} x_{C_k^\ell} &\ge (1-\delta)(1+4\delta)\\ &= (1+3\delta-4\delta^2)\\ &\ge 1,\end{aligned} \tag{9}$$

since $\delta \le \frac{3}{4}$. Consequently, in each platform the length of our generated schedule is also extended to

$$\begin{aligned}\sum_{k=1}^{q(\ell)} x_{C_k^\ell} &= (1+4\delta)\sum_{k=1}^{q(\ell)} \tilde{x}_{C_k^\ell}\\ &\le (1+4\delta)(1+\varepsilon)s_\ell PS(J)\\ &\le (1+3\varepsilon)s_\ell PS(J) \le (1+3\varepsilon)s_\ell \mathrm{OPT}_{SPP}(J).\end{aligned}$$

6.3 Rounding the Fractional Solution.

In this section we round the jobs in order to get a unique assignment of every job to a subset of processors of a platform. Consider an approximate solution $(x_{C_k^\ell})$ of the LP-relaxation. We introduce a new variable $x_j^\ell \in [0, p_j]$ that indicates the length of the fraction of job J_j that is scheduled on P_ℓ. Formally this is $x_j^\ell = \sum_{\{k\in\{1,\dots,q(\ell)\}|C_k^\ell(j)=1\}} x_{C_k^\ell}$, the sum of the length of all configurations in P_ℓ in which J_j appears. We can assume for all jobs J_j the equality $\sum_{\ell=1}^{N} x_j^\ell = p_j$, if not we simply delete job J_j from appropriate configurations or replace a configuration by two "shorter" configurations (one with job J_j and one without, their total length is the same as the one of the original configuration). For all fractions x_j^ℓ of a platform P_ℓ we build rectangles (x_j^ℓ, q_j) of height x_j^ℓ and width q_j. We use Algorithm 11 to group the rectangles of every platform P_ℓ geometrically.

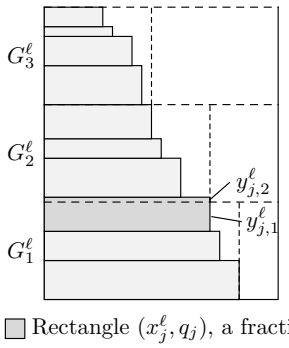

Figure 5: Constructing L'^ℓ_{wide}

If we were able to round the variables $z_{j,i}^\ell$ to integer values $\{0,1\}$ (without losing too much), this would imply a unique assignment of every rectangle to exactly one group of

Algorithm 11 Grouping

1: **for all** $1 \leq \ell \leq N$ **do**
2: Choose $\delta := \varepsilon/(4+\varepsilon)$ and partition the rectangles into wide and narrow rectangles, $L^\ell_{wide} := \{(x^\ell_j, q_j) | q_j > (\delta/2)m_\ell\}$ and $L^\ell_{narrow} := \{(x^\ell_j, q_j) | q_j \leq (\delta/2)m_\ell\}$.
3: Group the rectangles in L^ℓ_{wide} with Step 1 to 5 of Algorithm 6 into $M \leq 5/\delta(\lfloor \log(2/\delta) \rfloor + 1) = O(1/\varepsilon \log(1/\varepsilon))$ groups G^ℓ_i. Denote the resulting list of rectangles with L'^ℓ_{wide} and let $y^\ell_{j,i} \in [0, p_j]$ denote the fraction of job j that is assigned to G^ℓ_i. Let $z^\ell_{j,i} = y^\ell_{j,i}/p_j \in [0,1]$ denote the scaled fraction.
4: Compute $SIZE(L^\ell_{narrow}) = \sum_{(x^\ell_j, q_j) \in L^\ell_{narrow}} x^\ell_j q_j$ and locate the corresponding rectangles on top of the stack as group G^ℓ_0. Let $y^\ell_{j,0} \in [0, p_j]$ denote the fraction of a narrow job J_j that is assigned to G^ℓ_0 and and let $z^\ell_{j,0} = y^\ell_{j,0}/p_j \in [0,1]$.
5: **end for**

a platform. Re-identifying the rectangles with jobs, where we identify the height of a rectangle with the length of a job, this would also imply a unique assignment of every job to a platform. We achieve such a rounding of the variables $z^\ell_{j,i}$ via the following general assignment problem, so that there remain at most $M+1$ fractionally assigned rectangles per platform.

$$\begin{aligned} \sum_{j=1}^{n} z^\ell_{j,0} p_j q_j &\leq SIZE(L^\ell_{narrow}) \ \ell \in \{1, \ldots, N\} \\ \sum_{j=1}^{n} z^\ell_{j,i} p_j &\leq H(G^\ell_i) \ i \in \{1, \ldots, M\}, \ell \in \{1, \ldots, N\} \\ \sum_{\ell=1}^{N} \sum_{i=0}^{M} z^\ell_{j,i} &\geq 1 \ j \in \{1, \ldots, n\} \\ z^\ell_{i,j} &\in [0,1] \end{aligned} \tag{10}$$

The above formulation is related to the problem of scheduling jobs on parallel unrelated machines with $(M+1)N$ machines. Each group G^ℓ_i corresponds to a machine. Lenstra et al. showed in [16] that a feasible solution $(z^\ell_{i,j})$ of this problem can be rounded to a feasible solution $(\tilde{z}^\ell_{i,j})$ of the corresponding integer program formulation in polynomial time, so that there remains at most one fractional job $\tilde{z}^\ell_{i,j} < 1$ per machine. Hence, we get a unique assignment of almost all rectangles to the platforms P_ℓ except at most $M+1$ fractionally assigned rectangles per platform. Let F^ℓ denote the set of rectangles with fractional variables $\tilde{z}^\ell_{j,i}$ after the rounding. We will execute the corresponding jobs at the end of the schedule; their total processing time is bounded by $(M+1)t_{\max}$. From now on we consider for each platform P_ℓ an instance of strip packing containing a set of wide rectangles $\tilde{L}^\ell_{wide} := \{(\tilde{z}^\ell_{j,i} p_j, q_j) | \tilde{z}^\ell_{j,i} = 1, i > 0\}$ and a set of narrow rectangles $\tilde{L}^\ell_{narrow} := \{(\tilde{z}^\ell_{j,0} p_j, q_j) | \tilde{z}^\ell_{j,0} = 1\}$. In every platform we repack the pre-assigned rectangles in $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ using Algorithm 7.

7 Analysis

In the end we re-identify the rectangles with jobs, i.e. their widths with q_j and their heights with p_j. Note that a packing of the rectangles of total height $h(\ell)$ in platform P_ℓ corresponds to a schedule with makespan $h(\ell)/s_\ell$. Then the fractional jobs in F^ℓ are scheduled on top. To directly apply strip packing results we scale the widths of all rectangles in $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ by $1/m_\ell$. Furthermore, we consider platform P_ℓ as a strip of width 1 and infinite height. As we consider each platform and the allocated jobs independently, this has no impact on the solution.

7.1 Analyzing the Output

Let $(x_{C^\ell_k})$ be an approximate solution of (5) and let $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ contain the rectangles that have to be repacked in Step 5 of Algorithm 10 with the strip packing subroutine above.

Let L^ℓ_{round}, L'^ℓ_{round} and $\tilde{L}^\ell_{round}$ denote the instances obtained from L^ℓ_{wide}, L'^ℓ_{wide} and $\tilde{L}^\ell_{wide}$ via rounding algorithm 6. Remember that the difference between L^ℓ_{wide} and L'^ℓ_{wide} is that in L'^ℓ_{wide} every rectangle is uniquely assigned to one group G^ℓ_i since we introduced two new rectangles for border rectangles before. Thus $L^\ell_{round} = L'^\ell_{round}$. Moreover, we can show the following:

Lemma 7.1. *For all $\ell \in \{1, \dots, N\}$ we have*

a) $FSP(\tilde{L}^\ell_{round}) \leq (1+\delta)^2 FSP(L^\ell_{wide})$

b) $SIZE(\tilde{L}^\ell_{round}) \leq (1+\delta)^2 SIZE(L^\ell_{wide})$.

Proof. First notice that $FSP(L^\ell_{wide}) = FSP(L'^\ell_{wide})$ since we are comparing fractional solutions. By Lemma 4.3 we have $(1+\delta)FSP(L^\ell_{wide}) \geq FSP(L^\ell_{round}) = FSP(L'^\ell_{round})$. During building the the lists $\tilde{L}^\ell_{wide}$ from lists L'^ℓ_{wide} in step 3 of Algorithm 10 via the general assignment problem we do not increase the total height of any group G^ℓ_i and we do not exceed the largest width of a rectangle that appears in it. Therefore we have $L'^\ell_{round} \geq_g \tilde{L}^\ell_{wide}$ and thus $FSP(L'^\ell_{round}) \geq FSP(\tilde{L}^\ell_{wide})$. Again applying Lemma 4.3 to $\tilde{L}^\ell_{wide}$ and $\tilde{L}^\ell_{round}$ gives a). In a similar way assertion b) follows. □

Let h^ℓ_{round} denote the height of the packing produced by converting the fractional solution of 3 for the instance $\tilde{L}^\ell_{round}$ into an integral one. This is done by adding for each configuration appearing with height > 0 in the fractional solution the maximum height of a rectangle. Each basic solution of (3) has at most M non-zero entries. Thus, there are effectively at most $2M$ different configurations after filling the configurations as in Lemma 5.2 (since we consider only one strip we do not divide any configuration). Consequently we achieve a feasible integral packing for $\tilde{L}^\ell_{round}$ with height $h^\ell_{round} \leq FSP(\tilde{L}^\ell_{round}) + (1 + 2M)\max\{p_j | (p_j, q_j) \in \tilde{L}^\ell_{sup}\}$.

Now let $h(\ell)$ denote the height after adding the narrow rectangles in $\tilde{L}^\ell_{narrow}$ to our packing in platform P_ℓ, $\ell \in \{1, \dots, N\}$. We bound $h(\ell)$ in the following way:

Lemma 7.2. *For all $\ell \in \{1, \dots, N\}$ we have*
$h^\ell \leq (1+7\varepsilon)PS(J)s_\ell + \mathcal{O}(1/\varepsilon \log(1/\varepsilon)) \max\{p_j | (p_j, q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\}$.

Proof. First note that by construction we have that the height of an optimal fractional strip packing for the wide and narrow rectangles in $L^\ell_{wide} \cup L^\ell_{narrow}$ into platform P_ℓ, is less than the length of the schedule corresponding to the approximate solution of (5) constructed in step 1, that is

$$FSP(L^\ell_{wide} \cup L^\ell_{narrow}) \leq s_\ell(1+3\varepsilon)PS(J). \tag{11}$$

We consider two different cases:
If $h(\ell) > h^\ell_{round}$, Kenyon and Rémila showed that

$$h(\ell) \leq \frac{SIZE(\tilde{L}^\ell_{round} \cup \tilde{L}^\ell_{narrow})}{(1-\delta/2)} + (4M+1)\max\{p_j|(p_j,q_j) \in \tilde{L}^\ell_{round} \cup \tilde{L}^\ell_{narrow}\}.$$

Since $SIZE(\tilde{L}^\ell_{narrow}) \leq SIZE(L^\ell_{narrow})$ by construction Lemma 7.1 b) implies

$$SIZE(\tilde{L}^\ell_{round} \cup \tilde{L}^\ell_{narrow}) \leq (1+\delta)^2 SIZE(L^\ell_{wide} \cup L^\ell_{narrow}). \tag{12}$$

With (11) and since we scaled the widths of the rectangles by $1/m_\ell$ we have $SIZE(L^\ell_{wide} \cup L^\ell_{narrow}) \leq s_\ell(1+3\varepsilon)PS(J)$. With $\delta = \frac{\varepsilon}{4+\varepsilon}$

$$\begin{aligned}
h(\ell) &\overset{(12)}{\leq} A(L^\ell_{wide} \cup L^\ell_{narrow})\frac{(1+\delta)^2}{(1-\delta/2)} \\
&\quad + (4M+1)\max\{p_j|(p_j,q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\} \\
&\leq \frac{(1+3\varepsilon)PS(J)s_\ell(1+\delta)^2}{(1-\delta/2)} \\
&\quad + (4M+1)\max\{p_j|(p_j,q_j) \in L^\ell_{round} \cup L^\ell_{narrow}\} \\
&\leq (1+7\varepsilon)PS(J)s_\ell + (4M+1)\max\{p_j|(p_j,q_j) \in L^\ell_{round} \cup L^\ell_{narrow}\}
\end{aligned}$$

If $h(\ell) \leq h^\ell_{round}$ the converting process from fractional to integral gives us

$$\begin{aligned}
h^\ell_{round} &\leq FSP(\tilde{L}^\ell_{round}) \\
&\quad + (1+2M)\max\{p_j|(p_j,q_j) \in \tilde{L}^\ell_{wide}\} \\
&\overset{7.1\ a)}{\leq} FSP(L^\ell_{wide})(1+\delta)^2 \\
&\quad + (1+2M)\max\{p_j|(p_j,q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\} \\
&\leq FSP(L^\ell_{wide} \cup L^\ell_{narrow})(1+\delta)^2 \\
&\quad + (1+2M)\max\{p_j|(p_j,q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\} \\
&\overset{(11)}{\leq} s_\ell(1+3\varepsilon)PS(J)(1+\delta)^2 \\
&\quad + (1+2M)\max\{p_j|(p_j,q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\} \\
&\leq (1+6\varepsilon)PS(J)s_\ell + (1+2M)\max\{p_j|(p_j,q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\}
\end{aligned}$$

According to Lemma 4.3 we have $M \leq 5/\delta(\lfloor\log(2/\delta)\rfloor + 1) = \frac{5(4+\varepsilon)}{\varepsilon}(\lfloor\log(\frac{2(5+\varepsilon)}{\varepsilon})\rfloor + 1) = O(1/\varepsilon\log(1/\varepsilon))$ □

The packing in each platform P_ℓ corresponds to a schedule with length (referring to p_j) at most $(1+7\varepsilon)PS(J)s_\ell + \mathcal{O}(1/\varepsilon \log(1/\varepsilon)) \max\{p_j | (p_j, q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\}$, thus its completion time (referring to t^ℓ_j) is bounded by $(1+7\varepsilon)PS(J) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))t_{\max}$. The remaining jobs in F^ℓ have total processing time bounded by
$(M+1)t_{\max} \in \mathcal{O}(1/\varepsilon \log(1/\varepsilon)t_{\max}) \leq \mathcal{O}(1/\varepsilon \log(1/\varepsilon)p_{\max})$, since $t_{\max} \leq p_{\max}$ as $\min s_\ell = 1$. Adding the remaining jobs in F^ℓ to the schedule does not change the magnitude of the additive factor. With rescaling ε and since $PS(J) \leq \mathrm{OPT}_{SPP}(J)$ we obtain that the makespan of the produced schedule in each platform P_ℓ is less than $C_{\max}(\ell) \leq (1+\varepsilon)\mathrm{OPT}_{SPP}(J) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon)p_{\max})$ and conclude our main Theorem 1.2. Since during the repacking process we considered jobs as rectangles, we assigned every job to a set of processors with consecutive addresses. Thus we also obtain an AFPTAS for multiple strip packing for strips with different widths (in this case we have $s_\ell = 1$ for all $\ell \in \{1, \ldots, N\}$).

7.2 Running Time of the Algorithm

The time needed for solving (5) approximately via max-min resource sharing in step 1 is 'binary search on T' $\times$ 'number of iterations' $\times N \times$ 'solving the knapsack' which is less than $\mathcal{O}(\log(nN\varepsilon^{-1} \max s_\ell)) \times \mathcal{O}(n(\varepsilon^{-2} + \log n)) \times N \times \mathcal{O}(n \log(1/\varepsilon) + \varepsilon^{-4}) \leq \mathcal{O}(Nn^2\varepsilon^{-6} \log^2(n) \log^2(1/\varepsilon) \log(N \max s_\ell))$. The number of non-zero configurations in the final solution is bounded by the number of iterations $\mathcal{O}(n(\varepsilon^{-2} + \log n))$ [8], since in each iteration there is at most one new configuration included. So step 2 takes time $\mathcal{O}(Nn^2(\varepsilon^{-2} + \log n) \log(n^2(\varepsilon^{-2} + \log n))$
$= \mathcal{O}(Nn^2\varepsilon^{-2} \log^2(n) \log(1/\varepsilon))$, since there are at most $n^2(\varepsilon^{-2} + \log n)$ rectangles in each platform that have to be sorted. We represent the assignment problem in step 3 as a weighted bipartite graph $G = (V_1, V_2, E)$, where V_1 corresponds to the $N(M+1)$ machines (parts of the stacks), V_2 to the jobs. There is an edge between the node representing part i of the stack for P_ℓ and the node representing job J_j if $z^\ell_{j,i} > 0$. This assignment problem can be converted in time $\mathcal{O}(|E||V_1|) = \mathcal{O}(|V_1|^2|V_2|) = \mathcal{O}(\varepsilon^{-2} \log^2(1/\varepsilon)N^2n)$ into another assignment problem, whose corresponding graph is a forest [17]. Applying the rounding technique in [16] to the new assignment takes time in $\mathcal{O}(|V_1| + |V_2|) = \mathcal{O}(\varepsilon^{-1} \log(1/\varepsilon)N + n)$. So step 3 takes time in $\mathcal{O}(\varepsilon^{-2} \log^2(1/\varepsilon)N^2n)$. In step 5 it is sufficient to solve the corresponding linear program (3) approximatively with accuracy $\Theta(\varepsilon)$ also via a max-min resource sharing problem. This can be done in time $\mathcal{O}(M(\varepsilon^{-2} + \ln M) \ln(\varepsilon^{-1}) \max\{M + \varepsilon^{-3}, M \ln\ln(M\varepsilon^{-1})\})$ for every platform [11]. Since $M \in \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$ this gives for step 5 a total running time in $\mathcal{O}(N\varepsilon^{-6})$. The overall running time sums up to $\mathcal{O}(\varepsilon^{-6}N^2n^2 \log^2(n) \log^2(1/\varepsilon) \log(N \max s_\ell))$.

8 Malleable Jobs

One can also obtain an AFPTAS for scheduling malleable jobs non-preemptively by only adding a few modifications to the algorithm. To get a better overview we do not consider the platform speeds here. But remember that one can easily add speeds here by considering bins of height $s_\ell T$ instead of T, where T denotes an optimum value for the makespan for scheduling malleable jobs in platforms. In the following we give a short instruction how to adjust our algorithm:

In malleable scheduling a job J_j is described by a function $p_j : \{1, \ldots, m_N\} \longrightarrow \mathcal{Q}^+ \cup \infty$, where $p_j(k)$ is the length of job j running on k parallel processors of a platform. We introduce a configuration as a map $f_\ell : \{1, \ldots, m_\ell\} \longrightarrow \{0\} \cup \{1, \ldots, n\}$ that assigns a processor to a job (0 for idle time). Instead of solving (5) we can solve in a similar way the following linear program:

$$\begin{aligned} \sum_{f_\ell \in \mathcal{F}^\ell} x_{f_\ell} &= T \;\; \ell \in \{1, \ldots, N\} \\ \sum_{\ell=1}^{N} \sum_{k=1}^{m_\ell} \frac{1}{p_j(k)} \sum_{f_\ell \in \mathcal{F}^\ell, |f^{-1}(j)=k|} x_{f_\ell} &\geq 1 \;\; j \in \{1, \ldots, n\} \\ x_{f_\ell} &\geq 0. \end{aligned} \tag{13}$$

Here the upper and lower bounds for binary search on T, U and L, respectively, are $U := nd_{\min}$ and $L := {}^{d_{\min}}/_{N}$ where $d_{\min} := \max_\ell d^\ell$ and
$d^\ell := \max_j \min_{1 \leq k \leq m_\ell} \{p_j(k) | p_j$ can be scheduled on $P_\ell\}$. The block problem also splits into N smaller block problems, where each of them corresponds to a multiple choice knapsack. Lawler showed in [15] that those knapsack problems can be solved approximately in fully polynomial time. To guarantee that we have chosen the right number of processors for a job we replace 11 by Algorithm 12 in step 2 of Algorithm 10

Algorithm 12 Grouping for malleable jobs

1: **for all** $1 \leq \ell \leq N$ **do**
2: Choose $\delta := {}^{\varepsilon}/_{(4+\varepsilon)}$ and partition the rectangles into wide and narrow rectangles, $L^\ell_{wide} := \{(x^\ell_j, q_j) | q_j > (\delta/2)m_\ell\}$ and $L^\ell_{narrow} := \{(x^\ell_j, q_j) | q_j \leq (\delta/2)m_\ell\}$.
3: Group the rectangles in L^ℓ_{wide} with Step 1 to 5 of Algorithm 6 into $M \leq 5/\delta(\lfloor \log(2/\delta) \rfloor + 1) = O(1/\varepsilon \log(1/\varepsilon))$ groups G^ℓ_i. Denote the resulting list of rectangles with L'^ℓ_{wide}. Let a^ℓ_i, b^ℓ_i be the smallest and the largest width of a rectangle in group G^ℓ_i and let $W^\ell_{i,j}$ be the set of widths job J_j adopts in G^ℓ_i.
4: For $i \in \{1, \ldots, M\}$ and $w \in W^\ell_{i,j}$ let $y^\ell_{j,i}(w)$ denote the fraction of job j of width w that is assigned to G^ℓ_i. Let $z^\ell_{j,i} = \sum_{w \in W_{i,j}} y^\ell_{j,i}(w)$ be the complete fraction of job j in G^ℓ_i.
5: For each group $i \in \{1, \ldots, M\}$ and job j with $|W^\ell_{j,i}| \geq 2$ compute $k^\ell_{j,i} := \arg\min_{k \in [a^\ell_i, b^\ell_i]} p^\ell_j(k)$ and replace the rectangles corresponding to job j in G^ℓ_i by $(z^\ell_{j,i} p_j(k^\ell_{j,i}) k^\ell_{j,i})$. Note that $p_j(k^\ell_{j,i})$ is the smallest processing time among all processor numbers $k \in [a^\ell_i, b^\ell_i]$.
6: For each job j with $|W^\ell_{j,0}| \geq 2$ compute $k^\ell_{j,0} := \arg\min_{k \in [0, \varepsilon' m_\ell]} p^\ell_j(k)k$ and replace all rectangles corresponding to job j in G^ℓ_0 by $(z^\ell_{j,0} p_j(k^\ell_{j,0}), k^\ell_{j,0})$.
7: **end for**

Including different speed values we define the processing time of job J_j in platform P_ℓ as $t^\ell_j(k) = \frac{p_j(k)}{s_\ell}$. Note that $t^\ell_j(k) = \infty$ is possible. We define $p_{\max} := \max_{j,k} \{p_j(k) | p_j(k) < \infty\}$ and $t_{\max} := \max_{j,k,\ell} \{t^\ell_j(k) | t^\ell_j(k) < \infty\}$. To include speed values in the linear program we change the first N constraints of LP (13) into $\sum_{f_\ell \in \mathcal{F}^\ell} x_{f_\ell} = s_\ell T$, since different speeds can

be considered as providing length $s_\ell T$ instead of T for the schedule. The block problem also splits into N multiple choice knapsack problems with bin sizes $s_\ell T$ and can be solved in a similar way. The rounding step and the repacking process remain the same and finally we achieve the following theorem.

Theorem 8.1. *There is an AFPTAS for scheduling non-preemptive malleable jobs in heterogeneous platforms with different speeds with additive factor* $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}$.

9 Release Times

For a better overview we describe here the idea for the proof when all platforms run with the same speed, i.e. $s_\ell = 1$ for all $\ell \in \{1, \dots, N\}$. The general case can be derived from it.

Theorem 9.1. *There is an AFPTAS for scheduling parallel jobs in heterogeneous platforms with different speeds and release times with additive factor* $\mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))p_{\max}$.

Proof. Let T denote the optimum value of the makespan for $SPP(\mathrm{J})$ with release times $r_1, \dots, r_n$. Let $\Phi := \max_j r_j$, $\varepsilon > 0$ and assume that $1/\varepsilon \in \mathbb{N}$ or round it up to the next integer. Clearly we have $\Phi \leq T \leq \Phi + n \max_j p_j$. If $\Phi \leq \varepsilon T$, we can apply Algorithm 10 to the instance ignoring the release times and shift the constructed schedule in the end by εT. So in this case we achieve for every accuracy ε an algorithm with output less than $(1+\varepsilon)\mathrm{OPT}(J) + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}$. Thus, we assume $\Phi > \varepsilon T$. As in [9] we round down the release time to the next multiples of $i\varepsilon T$ $i \in \{0, 1, \dots, 1/\varepsilon\}$ and obtain new release times $\tilde{r}_1 \dots, \tilde{r}_n$ with at most $R := 1/\varepsilon + 1 \in \mathcal{O}(1/\varepsilon)$ different values $\rho_1, \dots, \rho_R$. To recover the loss we made by rounding down we will shift the final schedule by εT in the end. For every platform P_ℓ we consider R new platforms $\tilde{P}_{\ell,i}$, $i \in \{1, \dots, R\}$, with m_ℓ processors and create a new instance $\tilde{J}_R$ of *SPP* (without release times) with RN platforms and n jobs. A job J_j can now be scheduled in platform $\tilde{P}_{\ell,i}$ if it fits and if it is already released, i.e. $q_j \leq m_\ell$ and $\tilde{r}_j \leq \rho_i$. For each of the new platforms $\tilde{P}_{\ell,i}$ the value of an optimal fractional schedule is at most εT. Now we slightly modify the concept of a configuration: A configuration for a platform $\tilde{P}_{\ell,i}$ is a function $C^{\ell,i} : \{1, \dots, n\} \longrightarrow \{0, 1\}$, so that

- $\tilde{r}_j \leq \rho_i$ for all $j \in \{1, \dots, n\}$ with $C^{\ell,i}(j) = 1$
- and $\sum_{\{j \in \{1,\dots,n\} | C^\ell(j)=1\}} q_j \leq m_\ell$.

We can apply Algorithm 10 (using the new concept of configurations in the LP-relaxation) to $\tilde{J}_R$ obtaining in each platform $\tilde{P}_{\ell,i}$ a feasible schedule with length $(1+\varepsilon)\varepsilon T + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}$. Summing up the schedules for $\tilde{P}_{\ell,1}, \dots, \tilde{P}_{\ell,R}$ (in the right order of course) we create a schedule for the original platform P_ℓ with length $1/\varepsilon[(1+\varepsilon)\varepsilon T + \mathcal{O}(1/\varepsilon \log(1/\varepsilon))p_{\max}] = (1+\varepsilon)T + \mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))p_{\max}$. Correcting the mistake we made in the beginning and shifting the whole schedule by εT we obtain in every platform a schedule with length

$$(1+2\varepsilon)T + \mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))p_{\max}.$$

The running time of the algorithm is in
$O(\varepsilon^{-8} N^2 n^2 \log^2(n) \log^2(1/\varepsilon) \log(\max\{N, 1/\varepsilon\}))$ as we apply Algorithm 10 to $N := N/\varepsilon$ platforms and $s_\ell = 1$ for all $\ell \in \{1, \dots, N\}$. □

References

[1] N. Bansal, A. Caprara, K. Jansen, L. Prädel, and M. Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC 2009), LNCS 5878*, pages 77–86, 2009.

[2] M. Bougeret, P. F. Dutot, K. Jansen, C. Otte, and D. Trystram. Approximation algorithms for multiple strip packing. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA 2009), LNCS 5893*, pages 37–48, 2009.

[3] 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)*, pages 316–327, 2010.

[4] M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, and D. Trystram. A fast 5/2-approximation algorithm for hierarchical scheduling. In *Proceedings of the 16th International Euro-Par Conference- Parallel Processing Part I (Euro-Par 2010), LNCS 6272*, pages 157–167, 2010.

[5] M. Bougeret, P.-F. Dutot, K. Jansen, C. Robenek, and D. Trystram. Scheduling jobs on heterogeneous platforms. In *Computing and Combinatorics - 17th Annual International Conference (COCOON 2011)*, pages 271–283, 2011.

[6] A. Caprara. Packing d-dimensional bins in d stages. *Math. Oper. Res.*, 33:203–215, February 2008.

[7] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980.

[8] M. D. Grigoriadis, L. G. Khachiyan, L. Porkolab, and J. Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM J. Optim.*, 11(4):1081–1091, 2001.

[9] L. A. Hall and D. B. Shmoys. Approximation schemes for constrained scheduling problems. In *30th Annual Symposium on Foundations of Computer Science (FOCS 1989)*, pages 134–139, 1989.

[10] R. Harren, K. Jansen, L. Prädel and R. van Stee. A $(5/3 + \varepsilon)$-Approximation for Strip Packing. To appear in *Algorithms and Data Structures - 12th International Symposium (WADS 2011)*, pages 475–487, 2011.

[11] K. Jansen. *Efficient Approximation and Online Algorithms*, chapter Approximation algorithms for min-max and max-min resource sharing problems and applications, LNCS 3484, pages 156–202. Springer, 2006.

[12] K. Jansen and R. Solis-Oba. Rectangle packing with one-dimensional resource augmentation. *Discrete Optimization*, 6(3):310–323, 2009.

[13] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 312–320, 1982.

[14] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000.

[15] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Math. Oper. Res.*, 4(4):339–356, 1979.

[16] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[17] É. Tardos S. A. Plotkin, D. B. Shmoys. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

[18] I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the 2nd European Symposium on Algorithms (ESA 1994), LNCS 855*, pages 290–299, 1994.

[19] Ulrich M. Schwarz. *Approximation Algorithms for Scheduling and Two-Dimensional Packing Problems (Dissertation).* 2010.

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

[21] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.*, 26(2):401–409, 1997.

[22] A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk. Two level job-scheduling strategies for a computational grid. In *Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005), LNCS 3911*, pages 774–781, 2005.

[23] D. Ye, X. Han, and G. Zhang. On-line multiple-strip packing. *The 3rd Annual International Conference on Combinatorial Optimization and Applications (COCOA 2009), LNCS 5573*, 2009.

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