Discrete Optimization 6 (2009) 310–323

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Rectangle packing with one-dimensional resource augmentation

Klaus Jansen [a], Roberto Solis-Oba [b,*]

[a] Institut für Informatik, Universität zu Kiel, Kiel, Germany

[b] Department of Computer Science, The University of Western Ontario, London, Canada

ARTICLE INFO

Article history:
Received 21 January 2008
Received in revised form 31 March 2009
Accepted 2 April 2009
Available online 7 May 2009

Keywords:
Approximation algorithms
Rectangle packing
Strip packing
Optimization

ABSTRACT

In the rectangle packing problem we are given a set R of rectangles with positive profits and the goal is to pack a subset of R into a unit size square bin $[0, 1] \times [0, 1]$ so that the total profit of the rectangles that are packed is maximized. We present algorithms that for any value $\epsilon > 0$ find a subset $R' \subseteq R$ of rectangles of total profit at least $(1 - \epsilon)OPT$, where OPT is the profit of an optimum solution, and pack them (either without rotations or with 90° rotations) into the augmented bin $[0, 1] \times [0, 1 + \epsilon]$.

This algorithm can be used to design asymptotic polynomial time approximation schemes (APTAS) for the strip packing problem without and with 90° rotations. The additive constant in the approximation ratios of both algorithms is 1, thus improving on the additive term in the approximation ratios of the algorithm by Kenyon and Rémila (for the problem without rotations) and Jansen and van Stee (for the problem with rotations), both of which have a much larger additive constant $O(1/\varepsilon^2)$, $\varepsilon > 0$.

1. Introduction

Recently, there has been a lot of interest in two and three-dimensional packing problems, like strip packing [1–4], two-dimensional bin packing [5–9], rectangle packing [10,1,11–14], and three-dimensional strip packing [15,16]. These problems play an important role in diverse applications like cutting stock, VLSI layout, image processing, internet advertisement, and multiprocessor scheduling. In this paper, we study the strip packing problem and the problem of packing rectangles with profits into a rectangular bin. Let $R = \{R_1, \ldots, R_n\}$ be a set of n rectangles; each rectangle R_i has width $w_i \in (0, a]$, length $\ell_i \in (0, b]$ and profit $p_i \in \mathbb{R}^+$. The *rectangle-packing problem* is to pack a maximum profit subset $R' \subseteq R$ of rectangles into a unit size square bin B, $[0, 1] \times [0, 1]$. When packing the rectangles we only allow orthogonal packings; this means that the rectangles must not overlap and their sides must be parallel to the sides of the bin. In the *strip packing* problem the goal is to pack all the rectangles R into a strip of unit width and minimum length. We consider two variants of these problems: without rotations (the rectangles cannot be rotated) and with 90° rotations (the rectangles can be rotated 90° before they are packed).

The rectangle packing problem (with and without rotations) is known to be strongly NP-hard even for the restricted case of packing squares with unit profits [17]. The one-dimensional version of the problem is equivalent to the knapsack problem. The knapsack problem is weakly NP-hard, and several fully polynomial time approximation schemes (FPTAS) for it have been designed [18]. In contrast, since our problem is strongly NP-hard it has no FPTAS unless P = NP. Baker et al. [10] presented in 1983 a (4/3)-approximation algorithm to pack squares with unit profits into a rectangular bin. In 2004 Jansen and Zhang [14] designed a polynomial time approximation scheme (PTAS) for packing squares with unit profits in a rectangular bin and a

* Corresponding address: Department of Computer Science, The University of Western Ontario, Middlesex College Building, N6A 5B7 London, Ontario, Canada. Tel.: +1 519 661 2111; fax: +1 519 661 3515.

E-mail addresses: kj@informatik.uni-kiel.de (K. Jansen), solis@csd.uwo.ca (R. Solis-Oba).

1572-5286/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2009.04.001

$(2+\epsilon)$-approximation algorithm for packing rectangles with arbitrary profits [13], for any $\epsilon > 0$. Recently, Fishkin et al. [12] proposed an algorithm for the so-called rectangle packing problem with resource augmentation, that packs a subset of rectangles with profit at least $(1-\epsilon)OPT$ into an augmented square bin $[0, 1+\epsilon] \times [0, 1+\epsilon]$, where OPT is the maximum profit of any subset of rectangles that can be packed into a unit size square bin. For the rectangle packing problem restricted to squares, the best known algorithm [19] has performance ratio $\frac{5}{4} + \epsilon$, for any $\epsilon > 0$.

The strip packing problem (without and with rotations) is also strongly NP-hard [20]. The currently best approximation algorithms for the strip packing problem without rotations have absolute performance ratio 2 [3,4] and asymptotic performance ratio $1 + \epsilon$, for any $\epsilon > 0$ [2]. For strip packing with rotations the best approximation algorithm has absolute performance ratio 2 [4] and asymptotic performance ratio $1 + \epsilon$, for any $\epsilon > 0$ [21]. The asymptotic fully polynomial time approximation schemes (AFPTAS) by Kenyon and Rémila [2] and Jansen and van Stee [21] compute, respectively, a strip packing without rotations and with 90° rotations of total length at most $(1 + \epsilon)OPT + O(1/\epsilon^2)$ for any $\epsilon > 0$. Both algorithms run in time polynomial in n and $1/\epsilon$.

The first main result presented in this paper is the following:

Theorem 1.1. *Let $R = \{R_1, \ldots, R_n\}$ be a set of rectangles with lengths $\ell_i \leq 1$, widths $w_i \leq 1$ and profits $p_i \in \mathbb{R}^+$. For every constant value $\epsilon > 0$, there are polynomial time algorithms that select and pack a subset $R' \subseteq R$ of rectangles (without rotations and with 90° rotations) into an augmented rectangular bin $[0, 1] \times [0, 1+\epsilon]$. The profit of R' is at least $(1-\epsilon)OPT$, where OPT is the maximum profit of any subset of rectangles that can be packed into a unit size square bin $[0, 1] \times [0, 1]$.*

A remarkable feature of the algorithms presented in this paper for the rectangle packing problem is that they only need to augment the length of the bin, while its width does not need to be changed. We consider this result to be an important step towards the solution of 2-dimensional packing problems without resource augmentation. Not being allowed to increase the width of the bin poses some challenging technical problems, as we need to select a set of rectangles of nearly optimum profit and whose widths are not larger than the widths of the rectangles in an optimum solution. Furthermore, we need to be able to find a packing for these rectangles without constraining their possible positions across the bin. Two key ideas used to address these issues are: First, the discovery that all wide-short rectangles in a near optimum solution can be packed into a constant number of rectangular "containers"; these containers have the interesting property that they all have the same length and only a small number of different possible widths. Second, a transformation of a special case of the rectangle packing problem into a scheduling problem of parallel tasks on a constant number of machines.

A related result was recently obtained by Bansal and Sviridenko [6] for two dimensional bin packing. Here, the problem is to pack a set of rectangles into the minimum number of unit size square bins. They designed an algorithm that packs the rectangles into $OPT + O(1)$ bins of size $[0, 1] \times [0, 1+\epsilon]$, for any value $\epsilon > 0$, where OPT is the minimum number of unit size square bins needed to pack the rectangles. The algorithm in [6] rounds up the widths and lengths of large rectangles in a similar fashion as in [2]. This rounding causes, both an increase in the length of the bins, and an increase in the number of bins needed to pack the augmented rectangles. For our rectangle packing problem we cannot round the width and length of the large rectangles as this would require us to increase both dimensions of the bin.

Using our rectangle packing algorithm, we obtain the following result for the strip packing problem without rotations and with 90° rotations.

Theorem 1.2. *There are asymptotic polynomial time approximation schemes (APTAS) for the strip packing problem without rotations and with 90° rotations that produce solutions of length at most $(1 + \epsilon)OPT_{SP} + 1$, where OPT_{SP} is the value of an optimum solution for each problem.*

The additive constant 1 in the lengths of the solutions produced by our algorithms is very small and independent of $1/\epsilon$. Therefore, it greatly improves on the additive constant $O(1/\epsilon^2)$ of the approximation schemes by Kenyon and Rémila [2] and Jansen and van Stee [21]. Clearly, this is obtained at the expense of a higher running time. A similar situation happened with the bin packing problem (that is a special case of strip packing with rectangles of unit length): Fernandez de la Vega and Lueker [22] found a linear time algorithm that computes a solution using $(1 + \epsilon)OPT + 1$ bins and later Karp and Karmarkar [23] designed an algorithm that uses only $OPT + O(\log^2 OPT)$ bins, but it has a much higher running time.

We also show that the strip packing problem with 90° rotations for instances with optimum value $OPT_{SP} \geq 1$ has a PTAS.

Theorem 1.3. *There is a polynomial time approximation scheme (PTAS) for the strip packing problem with 90° rotations on instances with optimum value $OPT_{SP} \geq 1$.*

Concerning inapproximability, it has been shown in [24] that there is no approximation algorithm for the strip packing problem without rotations with an absolute approximation ratio smaller than 3/2 unless P = NP. In this paper we show that the same inapproximability bound also applies to the problem with 90° rotations.

1.1. Overview of the algorithm and organization of the paper

In Sections 2–6 we describe our algorithm for the rectangle packing problem without rotations. The algorithm for the problem with rotations is explained in Section 6.4. Finally, our algorithms for the strip packing problem are presented in Section 7. As the rectangle packing algorithm is rather complicated, we first present an overview of the algorithm.

The first main contribution of this work is a proof that there is a near optimal solution for the rectangle packing problem that has a simple structure. This is a crucial fact, as our approximation algorithm only builds packings that have this same simple structure. This proof makes use of some ideas (partitioning the set of rectangles, creating a gap in the sizes of different rectangle partitions, rounding the dimensions and positions of some of the rectangles, shifting rectangles to simplify the structure of the packing) the that have been used previously for other packing and scheduling problems [25,22,2,26].

We first consider an optimum solution S^* for the rectangle packing problem. This solution undergoes three changes, described in detail in Section 2. First, the set of rectangles R^* in this solution is partitioned by length into long, medium length, and short rectangles, and by width into wide, medium width, and narrow rectangles. This partitioning is made in such a way that the medium rectangles have small profit. Such rectangles are discarded, thus, loosing a small fraction of the total profit (but this creates a gap between the dimensions of the rectangles in the remaining partitions that will allow our algorithm to deal with each one of these groups independently). Second, the long rectangles have their lengths and horizontal positions rounded up. (This reduces the number of different possible packing that our algorithm needs to consider.) Third, most of the short-wide rectangles are packed into a constant number of rectangular regions, called containers, of fixed length. There is only a small number of possible widths for these containers (this will allow our algorithm to build containers of the same dimensions as those in the near-optimum solution), and the short-wide rectangles not packed in them have very small profit, so they are discarded.

In Sections 3–5 our rectangle packing algorithm is described. In Section 3 we indicate how to select the rectangles that will be packed in the bin. As mentioned already, the algorithm only considers packings with the simple structure detailed in Section 2. Since the long rectangles have had their lengths rounded, there is only a constant number of different lengths for them. For each possible length ℓ_i, the algorithm needs to determine the total width of the rectangles of length ℓ_i that will be packed. This step is described in Section 3.1. Then, a knapsack algorithm is used to select the actual set of long rectangles that will be placed in the bin.

As for the short rectangles, we use an algorithm for packing rectangles with large resources described in [11] to select (and actually pack) the short rectangles that will be placed in the containers. The remaining space in the bin, that will not be used by long rectangles or containers, will be used to pack short-narrow rectangles; a knapsack algorithm is used to select these rectangles, as described in Section 3.2.

Once a set of rectangles has been selected, the next step is to compute a packing for them. First, our algorithm determines the positions where the long rectangles and the containers will be placed in the bin. This is done by solving a linear program that gives the positions for the containers and yields a fractional packing for the long rectangles. The linear program is described in Section 4.

The fractional packing for the long rectangles is transformed into a feasible packing by using a simple greedy procedure. This transformation requires the removal of a small set of long rectangles of low profit. Finally, the short, narrow rectangles are packed in the bin by using the First Fit Decreasing Width algorithm, as described in Section 5. The analysis of the algorithm is presented in Section 6.

2. Existence of near-optimum packings with simple structure

The first step in the design of our rectangle packing algorithm is to show that there are near-optimum solutions with a simple structure, as described in Corollary 2.1. This fact greatly reduces the size of the search space for the problem, as our algorithm only needs to consider feasible solutions with that structure. Later we show how to explore the search space to find a solution of near optimum value in polynomial time.

2.1. Partitioning the set of rectangles

Let

$$\epsilon' = \min\{1/2, \epsilon/4\}, \tag{1}$$

where ϵ is the desired precision for the solution. We assume that $1/\epsilon'$ is integral and a multiple of 2, otherwise we can simply round ϵ' down to the largest value of the form $1/(2a) \leq \epsilon'$, for a integer. Consider an optimum solution S^* for the rectangle packing problem. Let R^* be the subset of rectangles selected by this optimum solution, and let the total profit of this solution be $OPT = \sum_{R_i \in R^*} p_i$.

Let $\sigma_1 = \epsilon'$ and $\sigma_k = (\sigma_{k-1})^{1+9/\sigma_{k-1}^2}$ for all integers $k \geq 2$. For each $k \geq 2$, we define the following sets of rectangles:

$$\mathcal{R}_k^* = \{R_i \in R^* \mid w_i \in (\sigma_k, \sigma_{k-1}] \text{ or } \ell_i \in (\sigma_k, \sigma_{k-1}]\}.$$

Each rectangle R_i belongs to at most two of these sets, so

$$\sum_{k\geq 2} \sum_{R_i \in \mathcal{R}_k^*} p_i \leq 2OPT. \tag{2}$$

Hence, there must be an index $\tau \in \{2, \ldots, 4/\epsilon' + 1\}$ such that $profit(\mathcal{R}^*_\tau) \leq (\epsilon'/2)OPT$, where $profit(\mathcal{R}^*_\tau) = \sum_{R_i \in \mathcal{R}^*_\tau} p_i$. To see this, note that if such an index τ does not exist, then $\sum_{k=2}^{4/\epsilon'+1} profit(\mathcal{R}^*_k) > \frac{4}{\epsilon'}\frac{\epsilon'}{2}OPT = 2OPT$, contradicting (2). Let

$$\delta = \sigma_\tau \quad \text{and} \quad s = 9/\delta^2 + 1. \tag{3}$$

We use these values δ and s to partition the set of rectangles in three groups according to their lengths: $\mathcal{L} = \{R_i \in R \mid \ell_i > \delta\}$, $\mathcal{H} = \{R_i \in R \mid \ell_i \leq \delta^s\}$, and $\mathcal{M}_\ell = \{R_i \in R \mid \ell_i \in (\delta^s, \delta]\}$. Then, we consider the widths of the rectangles and partition them into three additional groups: $\mathcal{W} = \{R_i \in R \mid w_i > \delta\}$, $\mathcal{N} = \{R_i \in R \mid w_i \leq \delta^s\}$, and $\mathcal{M}_w = \{R_i \in R \mid w_i \in (\delta^s, \delta]\}$. Observe that a rectangle belongs to exactly two of the above groups. The rectangles in $\mathcal{L}$, $\mathcal{H}$, $\mathcal{W}$, and $\mathcal{N}$ are called long, short, wide, and narrow, respectively (thus, there are long-wide, long-narrow, short-wide, and short-narrow rectangles). We also define $\mathcal{L}^* = \mathcal{L} \cap R^*$, $\mathcal{H}^* = \mathcal{H} \cap R^*$, $\mathcal{M}^*_\ell = \mathcal{M}_\ell \cap R^*$, $\mathcal{W}^* = \mathcal{W} \cap R^*$, $\mathcal{N}^* = \mathcal{N} \cap R^*$, and $\mathcal{M}^*_w = \mathcal{M}_w \cap R^*$.

Since $\delta = \sigma_\tau$ and $\delta^s = \sigma_{\tau+1}$, then $\mathcal{M}^*_\ell = \{R_i \in R^* \mid \ell_i \in (\sigma_{\tau+1}, \sigma_\tau]\}$, $\mathcal{M}^*_w = \{R_i \in R^* \mid w_i \in (\sigma_{\tau+1}, \sigma_\tau]\}$, and $\mathcal{M}^*_\ell \cup \mathcal{M}^*_w = \mathcal{R}^*_\tau$. Therefore, we deem these rectangles as non-important as their total profit is at most $(\epsilon'/2)OPT$. Let us discard the rectangles in $\mathcal{M}^*_\ell \cup \mathcal{M}^*_w$ from the optimum solution, creating a gap, $(\sigma_{\tau+1}, \sigma_\tau]$, between the lengths of the rectangles in $\mathcal{L}^*$ and $\mathcal{H}^*$, and between the widths of the rectangles in $\mathcal{W}^*$ and $\mathcal{N}^*$. This separation between the sets $\mathcal{L}^*$ and $\mathcal{H}^*$, and $\mathcal{W}^*$ and $\mathcal{N}^*$ is critical as it will allow us to deal independently with rectangles from different groups, as we show below.

Note that even when an optimum subset R^* of rectangles is not known, we can still assume that the value of τ is known. This is because there is only a constant number, $4/\epsilon'$, of possible values for τ. Thus, in our algorithm we simply try all these values (increasing the time complexity of the algorithm by only a constant factor); among all the solutions computed with these different values for τ, the algorithm chooses one with maximum profit.

2.2. Rounding the lengths of the long rectangles

We round up the length of each long rectangle $R_i \in R^*$ to the nearest multiple of δ^2. Then, we set the origin of a Cartesian system of coordinates at the left-bottom corner of the bin. In the optimum solution S^* we shift the rectangles horizontally to the right until all the long rectangles have their corners placed at points (x, y) where the x coordinates are multiples of δ^2 (see Fig. 1).

Since each long rectangle has length at least δ, the two above transformations increase the length of the packing by at most $\frac{1}{\delta}(2\delta^2) = 2\delta$. Accordingly, let us increase the length of the bin B to $1 + 2\delta$. This rounding and shifting limits the set of possible lengths for the long rectangles in R^* and the set of possible positions for their left sides in an optimum packing, while increasing the length of the bin by only a slight amount. These facts will aid us in selecting and packing a subset of long rectangles to produce a near-optimum solution for our problem.

2.3. Containers for short rectangles

Let us draw vertical lines across the bin B spaced by δ^2 (since the length of each long rectangle is a multiple of δ^2, the left and right sides of a long rectangle lie along two of these lines). These lines split the bin into a set of at most $(1 + 2\delta)/\delta^2$ vertical strips, that we call *slots*. A *container* C is defined as the rectangular region in a slot whose upper boundary is either the lower side of a long rectangle or the upper side of the bin, and whose lower boundary is either the upper side of a long rectangle or the lower side of the bin (see Fig. 1).

Consider a container c in the optimum solution S^* as shown in Fig. 1. Note that the two vertical sides of a container might be crossed by short rectangles. We can remove these rectangles through a shifting technique that decreases the total profit of the rectangles packed in c by only a small amount. To do this, let us first allocate all small rectangles crossing the border between two containers c and c' to the container that appears on the right. Then, we divide c into vertical strips spaced by a distance $3\delta^s$. The number of these strips is $\delta^2/(3\delta^s) > 1/(4\epsilon')$. Therefore, there must be a strip s' for which the total profit of the short rectangles completely contained in the strip is at most $\frac{\epsilon'}{4}OPT(c)$, where $OPT(c)$ is the total profit of the rectangles assigned to c in the optimum solution S^*. Let us remove all short rectangles completely contained in s', creating in c an empty vertical gap of length at least δ^s. Now, we can move all small rectangles crossing the left boundary of c to this empty gap (see Fig. 1).

By performing the above process over all containers we loose profit at most $\frac{\epsilon'}{4}OPT$, but now no short rectangle crosses the boundary of a container. Let S^*_c be the set of rectangles that remain in container c.

For the sequel, we only consider containers storing at least one short-wide rectangle inside them. Notice that any solution for the rectangle packing problem has a constant number $O(1/\delta^3)$ of such containers, since each container has length δ^2, width at least δ and, therefore, area at least δ^3.

Consider one such container C storing at least one short-wide rectangle in the optimum solution S^*. Let us momentarily forget about the short-narrow rectangles in C. Now, compute the distance from the bottom side of each short-wide rectangle in C to the bottom of C, and then sort the rectangles in non-decreasing order of these distances. Take the rectangles in this order and shift them down until they touch either the bottom of C or the top of another rectangle. After shifting the rectangles, we get a new feasible packing for them (see Fig. 2). Let $w_{\max}(C)$ be the width of this new packing, i.e., all the

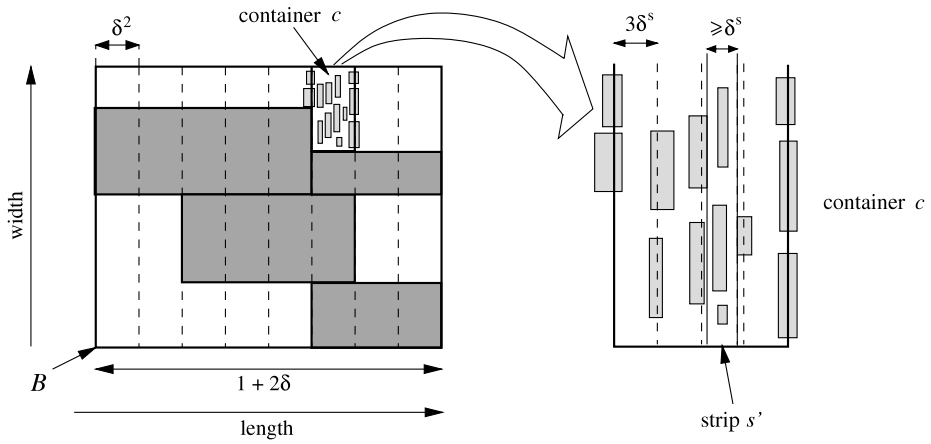

Fig. 1. Shifting the rectangles and discarding short rectangles crossing slot boundaries. Long rectangles appear in darker shade.

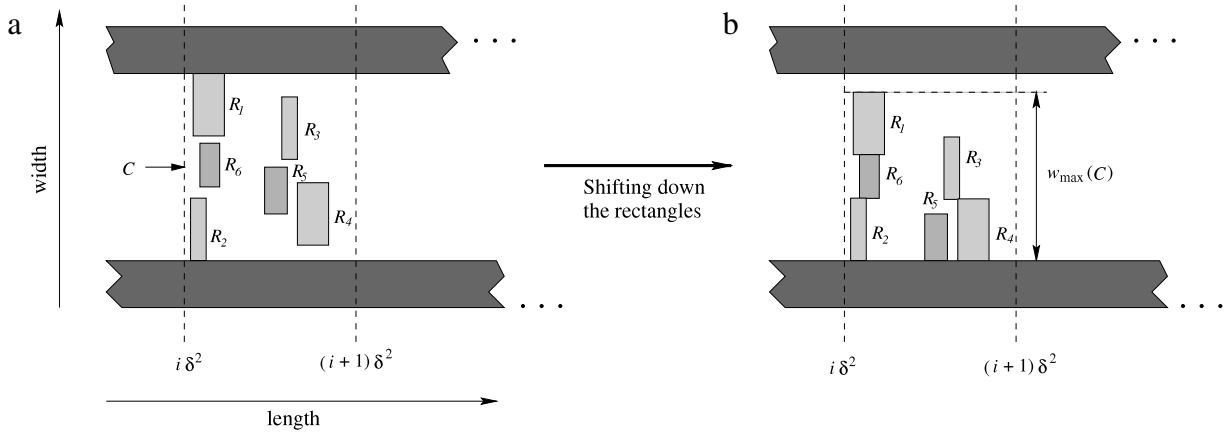

Fig. 2. (a) A container C. R_1, R_2, R_3, R_4, R_5, and R_6 are short-wide rectangles. (b) The container after shifting down the short-wide rectangles.

rectangles are contained in the bottom part of C up to width $w_{\max}(C)$. Note that $w_{\max}(C)$ is the sum of the widths of a constant number, at most $1/\delta$, of short-wide rectangles.

We now consider again all rectangles S_C^* as they were originally packed in C and, then, round the width of C down to the nearest value of the form $w_{\max}(C) + i\delta^s$, where i is an integer value. By doing this some of the rectangles from S_C^* might not fit in C anymore. In order to ensure that all these rectangles still fit, we need to increase the length of C. The following results show that the length of the container only needs to be increased by a small amount.

Lemma 2.1 (*From [2]*). *For any value $\epsilon > 0$, a set R of rectangles, each of length and width at most 1, can be packed in polynomial time in a strip of unit width and length at most $area(R)(1+\epsilon) + (4(2+\epsilon^2)^2/\epsilon^2 + 1)\Delta$, where $area(R)$ is the total area of the rectangles in R and Δ is the maximum length of the rectangles of width smaller than ϵ.*

Lemma 2.2. *The rectangles in S_C^* can be packed in a container C' of width w' and length $\delta^2 + 2\delta^4$, where w' is the width of C rounded down to the nearest value of the form $w_{\max}(C) + i\delta^s$, for an integer value $i \leq n$.*

Proof. Note that by the way in which $w_{\max}(C)$ was defined, all short-wide rectangles in S_C^* fit in the modified container C'. Therefore, we can pack all the rectangles from S_C^* in C' as follows. First we multiply by $1/w'$ the width and length of each rectangle in S_C^*. We do the same thing with the dimensions of C', so that C' has unit width. Then, by Lemma 2.1 with $\epsilon = \delta^2$, the scaled rectangles can be packed in a strip of unit width and length at most

$$area(S)(1+\delta^2) + \left(\frac{4(2+\delta^2)^2}{\delta^2} + 1\right)\delta^s/w' = area(S)(1+\delta^2) + \frac{16 + 17\delta^2 + 4\delta^4}{\delta^2}\delta^s/w'$$
$$< area(S)(1+\delta^2) + 17\delta^{s-2}/w', \quad \text{as } \epsilon' \leq 1/2 \text{ and so } \delta \leq (1/2)^{36},$$

where $area(S)$ is the total area of the scaled rectangles, which is at most $(w'+\delta^s)\delta^2/(w')^2$. By re-scaling the rectangles back to their original sizes, the width of the strip reduces to w' and its length is

$$\frac{(w'+\delta^s)\delta^2(1+\delta^2)}{w'} + 17\delta^{s-2} \leq \delta^2+\delta^4+\delta^{s+1}(1+\delta^2)+17\delta^{s-2}, \quad \text{since } w' \geq \delta$$
$$< \delta^2+2\delta^4, \quad \text{since } \delta \leq (1/2)^{36} \text{ and so, } s \geq 9\times 2^{36}. \quad \square$$

By Lemma 2.2 every container C storing short-wide rectangles can be replaced by a container C' of length at most $\delta^2+2\delta^4$ and whose width is of the form $w_{\max}(C)+i\delta^s$. By using the same shifting technique described above we can decrease the length of C' down to δ^2 while discarding only a subset of rectangles of very small profit: We split the container C' into strips of length $4\delta^4$, obtaining $\lfloor(\delta^2+2\delta^4)/(4\delta^4)\rfloor = 1/(4\delta^2)$ strips of length $4\delta^4$ and one strip of length at most $4\delta^4$. If a rectangle packed inside C' crosses the boundary between strips, we assign the rectangle to the strip that appears on the right. The other rectangles are assigned to the strip that completely contains them. There is at least one strip whose rectangles have profit at most $4\delta^2 OPT(C)$. If we eliminate this strip from C' we create a gap of length at least $4\delta^4-2\delta^s > 2\delta^4$ inside C'. Therefore, the remaining rectangles fit into the original container C of length δ^2. Note that if we do this on all containers, we loose in total profit of value at most $4\delta^2 OPT \leq (\epsilon'/4)OPT$.

Corollary 2.1. *There is a set R^+ of rectangles of total profit at least $(1-\epsilon')OPT$ and a packing S^+ for them in a bin of width 1 and length at most $1+2\delta$ such that*

- *every long rectangle in R^+ has its length rounded up to the nearest multiple of δ^2 and it is positioned so that its left side is at a position x that is a multiple of δ^2, and*
- *each container C storing at least one short-wide rectangle has length δ^2 and width $w_{\max}(C)+i\delta^s$, where $w_{\max}(C)$ is the sum of the widths of at most $1/\delta$ short-wide rectangles and $i \leq n$ is a non-negative integer value.*

3. Rectangle selection

Our algorithm for the rectangle packing problem only considers packings with the structure described in Corollary 2.1. Among all the packings produced, the algorithm selects one with highest profit. We describe in this section the first step of the algorithm: How to select the set of rectangles that is going to be packed in the bin. In subsequent sections we show how to actually pack the selected rectangles in the bin.

As described in Section 2.1, our algorithm tries all possible values $\{2, 3, \ldots, 4/\epsilon'+1\}$ for τ. Let us consider one of these values for τ and define the sets $\mathcal{L}$, $\mathcal{M}_\ell$, $\mathcal{H}$, $\mathcal{W}$, $\mathcal{M}_w$, and $\mathcal{N}$ as described above. Let us assume that for this choice of τ, $profit(\mathcal{M}_\ell \cup \mathcal{M}_w) \leq (\epsilon'/2)OPT$. The rectangles in $\mathcal{M}_\ell \cup \mathcal{M}_w$ are ignored, so our algorithm only needs to deal with 3 disjoint classes of rectangles: long ($\mathcal{L}$), short-wide ($\mathcal{H} \cap \mathcal{W}$), and short-narrow ($\mathcal{H} \cap \mathcal{N}$).

3.1. Selecting long rectangles

To describe our selection of long rectangles, let us first assume that we know an optimum solution S^* for the rectangle packing problem. Let R^* be the set of rectangles selected by this optimum solution. Since the number of long rectangles in R^* could be very large, we cannot simply enumerate all subsets of long rectangles from R to find the set $R^* \cap \mathcal{L}$ of long rectangles in the optimum solution. Instead, we show below how to select a set of long rectangles of nearly the same profit and area as the long rectangles in R^*.

We round up the length of each long rectangle to the nearest multiple of δ^2 as described in Section 2.2. For any constant value $K > 0$, let L_K^* be the set formed by the K long rectangles in R^* of largest profit. The reason for introducing this parameter K will become apparent later when we analyze the packing produced by the algorithm for the selected set of rectangles. Let $\bar{L}_K^* = (R^* \cap \mathcal{L}) \setminus L_K^*$ be the remaining long rectangles in R^*, and let $\bar{L}_{Ki}^*$ be the subset of $\bar{L}_K^*$ formed by rectangles of length $i\delta^2$, for each $i = 1/\delta, 1/\delta+1, \ldots, 1/\delta^2$. For each set $\bar{L}_{Ki}^*$, let $w(\bar{L}_{Ki}^*)$ be the total width of the rectangles in it.

For any constant value K, the set L_K^* is, of course, not known. However, since L_K^* contains only a constant number K of rectangles, our algorithm can construct in polynomial time all $O(n^K)$ subsets of K long rectangles from R; clearly, one of these sets must be equal to L_K^*. For each possible selection of L_K^*, we can find good approximations for the values $w(\bar{L}_{Ki}^*)$ and for the corresponding sets $\bar{L}_{Ki}^*$ as follows.

- If set $\bar{L}_{Ki}^*$ has at most $1/\delta^4$ rectangles, then our algorithm will simply try all $O(n^{1/\delta^4})$ different subsets of at most $1/\delta^4$ long rectangles of length $i\delta^2$. One of these sets will be $\bar{L}_{Ki}^*$.
- If $\bar{L}_{Ki}^*$ has more than $1/\delta^4$ rectangles, the algorithm first considers all subsets of R with $1/\delta^4$ rectangles of length $i\delta^2$. One of these subsets will coincide with the set S_{Ki}^* of $1/\delta^4$ widest rectangles in $\bar{L}_{Ki}^*$. Take a rectangle $R_\ell^* \in S_{Ki}^*$ with lowest profit (such a profit is at most $\delta^4\, profit(\bar{L}_{Ki}^*)$) and use as approximations for $w(\bar{L}_{Ki}^*)$ all values of the form $\varpi + kw(R_\ell^*)$, where $\varpi = \sum_{R_j^* \in S^* Ki} w(R_j^*)$, $k \in \{0, \ldots, n-1/\delta^4\}$, and $w(R_j^*)$ is the width of rectangle R_j^*. Note that $w(\bar{L}_{Ki}^*)$ must be in the interval $[\varpi + xw(R_\ell^*), \varpi + (x+1)w(R_\ell^*)]$ for some integer $0 \leq x \leq n-\delta^{-4}$. If we remove R_ℓ^* from $\bar{L}_{Ki}^*$, the total width of the rectangles in $\bar{L}_{Ki}^*$ would be at most $\varpi + xw(R_\ell^*)$ and their profit would be at least $(1-\delta^4)profit(\bar{L}_{Ki}^*)$.

Corollary 3.1. *For each $i = 1/\delta, 1/\delta + 1, \ldots, 1/\delta^2$, we can find in polynomial time a set Λ_i of $O(n^{1/\delta^4+1})$ values of the form $\varpi + xw_\ell$, where ϖ is the sum of widths of the rectangles in a set $S \subseteq R$ of at most $1/\delta^4$ rectangles of length $i\delta^2$, w_ℓ is the width of a rectangle $R_\ell \in S$ of minimum profit, and $x \in \{0, 1, \ldots, n - 1/\delta^4\}$. This set Λ_i is such that there is a value $\varpi^* + x^* w_\ell \in \Lambda_i$ such that $\varpi^* + x^* w_\ell < w(\bar{L}^*_{Ki}) \leq \varpi^* + (x^* + 1)w_\ell$, and w_ℓ is the width of a rectangle of profit at most δ^4 profit $(\bar{L}^*_{Ki})$.*

For each $\bar{L}^*_{Ki}$, set Λ_i contains $O(n^{1/\delta^4+1})$ possible bounds for its width $w(\bar{L}^*_{Ki})$. Our algorithm will try all these bounds, and since there are fewer than $1/\delta^2$ sets $\bar{L}^*_{Ki}$, the total number of possible bounds that the algorithm needs to try is $O((n^{1+1/\delta^4})^{1/\delta^2})$, which is polynomial in n. Note that by Corollary 3.1, among these bounds there is a set $\{b^*_1, b^*_2, \ldots, b^*_{1/\delta^2-1/\delta+1}\}$ of them such that for each $i = 1/\delta, 1/\delta+!, \ldots, 1/\delta^2$, $b^*_i = \varpi^* + x^*_i w^*_i \leq w(\bar{L}^*_{Ki}) \leq \varpi + (x^*_i + 1)w^*_i$.

For each possible bound b_i for $w(\bar{L}^*_{Ki})$ the algorithm needs to select a subset of rectangles of width at most b_i and high profit to pack in our solution. To do this selection we use Lawler's polynomial time approximation scheme for the knapsack problem, [18], using the set of rectangles of length $i\delta^2$ as input: the desired precision used in Lawler's algorithm is δ, the width of each rectangle is used as its size, and b_i is the capacity of the knapsack.

By Corollary 3.1, among all $O(n^{1/\delta^4+1})$ sets selected by our algorithm, at least one of them must have profit at least $(1 - \delta)(1 - \delta^4)$*profit*$(\bar{L}^*_{Ki})$ and total width no larger than $w(\bar{L}^*_{Ki})$. Therefore, among the rectangle selections made by our algorithm for all sets $\bar{L}^*_{Ki}$, one of them will include rectangles of total profit at least $(1 - \delta)(1 - \delta^4) \sum_{i=1}^{1/\delta^2}$ *profit*$(\bar{L}^*_{Ki}) \geq (1 - 2\delta) \sum_{i=1}^{1/\delta^2}$ *profit*$(\bar{L}^*_{Ki})$.

We note that in total, our algorithm must select $O(n^{K+(1+1/\delta^4)/\delta^2})$ different sets of long rectangles.

3.2. Selecting short-wide and short-narrow rectangles

Let $\mathcal{C}^+_{sw}$ be the set of containers in the near optimal solution S^+ (as described in Corollary 2.1) that store short-wide rectangles. By Corollary 2.1, $|\mathcal{C}^+_{sw}| \leq 1/\delta^3$ and each container $C \in \mathcal{C}^+_{sw}$ has length δ^2 and width of the form $w_{\max}(C) + i\delta^s$, where $w_{\max}(C)$ is the sum of widths of at most $1/\delta$ short-wide rectangles and $i \leq n$ is an integer value. Therefore, the number of different possible widths for containers in $\mathcal{C}^+_{sw}$ is $O(n^{1/\delta+1})$, which is polynomial in n.

Since we do not know the set $\mathcal{C}^+_{sw}$, our algorithm will build packings with $0, 1, 2, \ldots, \delta^{-3}$ containers, and for each container the algorithm will try all $O(n^{1/\delta+1})$ possible widths. Clearly, one of these sets of containers must be identical to $\mathcal{C}^+_{sw}$. Consider such a choice of containers along with a selection S_L of long rectangles, as described in the previous section, such that (a) the profit of S_L is at least $(1 - \epsilon)$ times the profit of the long rectangles selected in S^+ and (b) the total width of the long rectangles of length $i\delta^2$ in S_L is no larger than the total width of the corresponding long rectangles in S^+, for all $i = 1/\delta, 1/\delta + 1, \ldots, 1/\delta^2$.

Let A_{sn} be the area of the bin of width 1 and length $1 + 2\delta$ minus the area of the rectangles in S_L and minus the area of the containers in $\mathcal{C}^+_{sw}$. Note that A_{sn} is no smaller than the total area of the short-narrow rectangles packed by S^+ outside the containers from $\mathcal{C}^+_{sw}$. We choose a set of short-narrow rectangles to pack outside $\mathcal{C}^+_{sw}$ by using again Lawler's algorithm [18], with precision ϵ, and using the area of each rectangle as its size and A_{sn} as the capacity of the knapsack. The profit of this set of rectangles is at least $(1 - \epsilon)$ times the profit of short-narrow rectangles packed by S^+ outside $\mathcal{C}^+_{sw}$.

The next step is to choose short-wide and short-narrow rectangles to be packed inside the containers in $\mathcal{C}^+_{sw}$. To do this we use the algorithm of Fishkin et al. from [11] for packing problem with large resources; this algorithm can select and pack a near-optimum profit set of rectangles into a container of length at least $1/(\epsilon')^3$ times the length of any rectangle. In our case, to be able to use this algorithm, we require the length, δ^2, of each container $C \in \mathcal{C}^+_{sw}$ to be at least $\delta^s/(\epsilon')^3$. Since $\epsilon' \leq 1/2$, then $\delta \leq (1/2)^{36}$ and $s \geq 9 \times 2^{36}$; therefore, $\delta^2 \geq \delta^s/(1/2)^3 \geq \delta^s/(\epsilon')^3$, as required.

The algorithm in [11] was designed to pack rectangles in a single container, but a straightforward extension allows it to consider a constant number of containers. The total profit of the short rectangles selected by Lawler's algorithm and by the algorithm in [11] is at least $(1 - \epsilon')$ times the total profit of the short rectangles selected by S^+.

Note that in total, our algorithm will select $O(n^{(1+1/\delta)/\delta^3})$ different sets of containers and short rectangles.

4. Positioning long rectangles and containers

As described in Section 3.2, our algorithm will select $O(n^{K+(1+1/\delta^4)/\delta^2+(1+1/\delta)/\delta^3})$ different sets of containers and rectangles that it will try to pack in the bin. Let L be one of these sets of rectangles and let $\mathcal{C}$ be the chosen set of containers. In order to try to find a valid packing for L we first need to determine the positions where the rectangles will be packed in the bin. To this end, let us split the bin into vertical slots of length δ^2.

Let $C \in \mathcal{C}$ be one of the containers selected by our algorithm and let S_C be the set of short-wide and short-narrow rectangles that our algorithm packed in C, as described in Section 3.2. Let us remove from L all the rectangles in S_C and replace them with a single new rectangle R_C of the same width and length as C. The same is done for all containers in C, so now L contains only long rectangles, rectangles replacing containers of $\mathcal{C}$, and short-narrow rectangles.

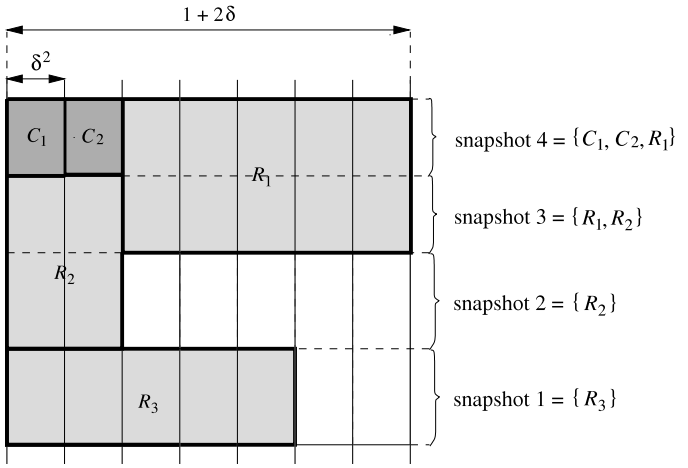

Fig. 3. Packing for a set of rectangles and containers, and the induced snapshots.

It is interesting to mention that the problem of packing the rectangles from L in the bin can be viewed as a scheduling problem: Each slot corresponds to a machine and every rectangle is a job. A long rectangle of length $i\delta^2$ is a job that requires the use of i consecutive machines. A container from $\mathcal{C}$ represents a job that needs one machine, and any set of short rectangles of total length at most δ^2 corresponds to a set of jobs that can be simultaneously processed by one machine. The processing time of each job is equal to the width of the corresponding rectangle. The goal is to minimize the makespan.

To determine the positions where long rectangles and containers will be packed, we will first compute a fractional packing for them. In this packing, we ensure that all containers and a large set of long rectangles of high profit are not split by the fractional packing. This is a crucial property of our algorithm, as we show below that we can find a fractional packing for L that splits only a small number of low profit rectangles; discarding these rectangles yields a valid packing while only slightly affecting the profit of the solution.

We use a linear program to construct the fractional packing. To understand this linear program we first need to define some concepts. Let $L' \subseteq L$ be the set formed by all rectangles corresponding to the containers of $\mathcal{C}$ and the K long rectangles of highest profit in L. A *slot assignment* for L' is a mapping $f : L' \to 2^M$, where $M = \{1, \ldots, (1 + 2\delta)/\delta^2\}$ is the set of slots, such that for each rectangle $R_j \in L'$, $f(R_j)$ is a consecutive set of γ_j slots, where $\gamma_j\delta^2$ is the length of R_j. Note that since the number, $(1 + 2\delta)/\delta^2$, of slots is constant, the number of different mappings f is constant, at most $((1 + 2\delta)/\delta^2)^{|L'|}$. The algorithm considers all these mappings, and for each one of them it tries to compute a packing for L that is consistent with the mapping. If no packing is found, then the set L is discarded and a different set of rectangles is selected as described above. At the end, the packing with the largest profit is finally selected by our algorithm.

Consider a packing S for L' according to a slot assignment f. Let us imagine a horizontal line that moves from the bottom to the top of the bin sweeping the packing S. A *snapshot* is any set of rectangles from L' that is simultaneously intersected by this line (see Fig. 3). Every rectangle $R_j \in L'$ appears in a sequence of consecutive snapshots $SHOT(\alpha_j), \ldots, SHOT(\beta_j)$, where $SHOT(\alpha_j)$ is the first snapshot and $SHOT(\beta_j)$ is the last snapshot that contains R_j. Here, we index the snapshots from the bottom to the top of the bin as shown in Fig. 3. For example, in Fig. 3, R_2 appears in snapshots 2 and 3, so $\alpha_2 = 2$ and $\beta_2 = 3$.

Partition the rectangles in $L \setminus L'$ into two groups: L_{sn} containing the short-narrow rectangles, and L_{lo} containing the long rectangles. Let $m'(i)$ be the set of slots occupied by the rectangles from L' in snapshot $SHOT(i)$. Then, $M \setminus m'(i)$ is the set of *free slots* inside snapshot $SHOT(i)$ that can be used to pack the rectangles in $L_{sn} \cup L_{lo}$. Let $\mathcal{F}$ be the family formed by all possible subsets of slots.

For each possible set $F \in \mathcal{F}$ of free slots let us define a *configuration* (SN, Π) as a tuple where SN is a subset of F and Π is a partition of $F \setminus SN$ into sets of consecutive slots. Let $c_{F,i}$, $i = 1, \ldots, n_F$, denote all possible configurations for F, and let n_F be the number of these configurations. Given a configuration $c_{F,i} = (SN_{F,i}, \Pi_{F,i})$, the first set $SN_{F,i}$ of slots will be reserved to pack rectangles from L_{sn}; every subset of slots $F' \in \Pi_{F,i}$ of cardinality ℓ will be reserved for packing long rectangles from L_{lo} of length $\ell\delta^2$. Let $n_{F,i}(\ell)$ be the number of subsets of cardinality ℓ in $\Pi_{F,i}$.

To pack the sets of rectangles L_{sn} and L_{lo}, we first use a linear program to assign them to slots. To describe this linear program, we need a variable $x_{F,i}$ for each configuration $c_{F,i}$, denoting the total width of the snapshots where the free slots are allocated according to $c_{F,i}$. Hence, the area reserved by a configuration $c_{F,i}$ to pack short-narrow rectangles is $|SN_{F,i}|\delta^2 x_{F,i}$.

Let $W_i(L_{lo})$ be the total width of the long rectangles of length $i\delta^2$ in L_{lo}, for all $i = 1/\delta, \ldots, 1/\delta^2$, and let A_{sn} be the total area of the short-narrow rectangles in L_{sn}. Since $|L'| \leq K + \delta^{-3}$, the maximum number of snapshots in any packing for L' is at most $g = 2(K + \delta^{-3})$.

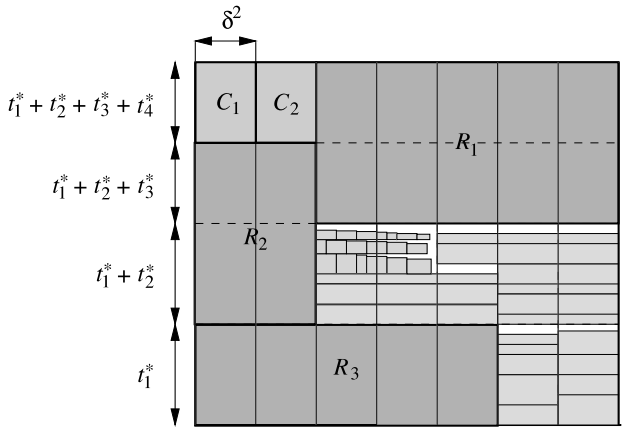

Fig. 4. Packing long rectangles and short narrow rectangles into the snapshots. $R_1, R_2, R_3 \in L'$; C_1 and C_2 are containers from $\mathcal{C}_{sw}^{+}$.

For each rectangle $R_i \in L'$ we try all possible values α_i, β_i for the first and last snapshots where R_i can be packed. Since L' has only a constant number (at most $K + \delta^{-3}$) of rectangles then there is only a constant number (at most $g^{2(K+\delta^{-3})}$) of different possible assignments of starting and ending snapshots for the rectangles in L'.

Let f be a slot assignment for L' and let α, β be assignments of starting and ending snapshots for the rectangles in L'. We use the following linear program to allocate rectangles to slots and snapshots. In the linear program, variable t_i is the sum of widths of the first i snapshots; variable e_F is the total width of the snapshots where the set of slots not occupied by rectangles from L' is F.

$$
\begin{aligned}
LP(f,\alpha,\beta): \quad & t_0 = 0, t_g \le 1 && \\
& t_i \ge t_{i-1} && i = 1,\ldots,g \\
& t_{\beta_j} - t_{\alpha_j - 1} = w_j && \forall R_j \in L' \\
& \sum_{i:F=M\setminus m'(i)} (t_i - t_{i-1}) = e_F && \forall F \in \mathcal{F} \\
& \sum_{i=1}^{n_F} x_{F,i} \le e_F && \forall F \in \mathcal{F} \\
& \sum_{F\in\mathcal{F}} \sum_{i=1}^{n_F} n_{F,i}(\ell) x_{F,i} \ge W_\ell(L_{lo}) && \ell = 1,\ldots,|M| \\
& \sum_{F\in\mathcal{F}} \sum_{i=1}^{n_F} |SN_{F,i}| \delta^2 x_{F,i} \ge A_{sn} && \\
& x_{F,i} \ge 0 && \forall F \in \mathcal{F}, i = 1,\ldots,n_F.
\end{aligned}
$$

The first three constraints ensure that the rectangles from L' are completely packed in the bin. The next three constraints guarantee that all long rectangles are (fractionally) allocated to empty space not occupied by L'. The seventh constraint makes sure that the remaining space in the bin is at least as large as the total area of the short-narrow rectangles selected for packing.

If $LP(f,\alpha,\beta)$ has no feasible solution, then we discard the slot and snapshot assignments f, α, and β.

5. Generating a packing

Let (t^*, e^*, x^*) be a feasible solution for the above linear program. Now we show how to transform this solution into a valid packing for the rectangles (for an illustration of a valid packing see Fig. 4). For simplicity, let us remove from this solution all snapshots $[t_i^*, t_{i+1}^*)$ of zero width, i.e. the snapshots for which $t_i^* = t_{i+1}^*$. Let g^* be the number of remaining snapshots. Furthermore, without loss of generality we may assume that all the configurations in which the set F of free slots is $F = M$ appear in the last snapshot $[t_{g^*-1}^*, t_{g^*}^*)$; otherwise, we can simply shift these configurations there.

5.1. Packing the long rectangles

Each rectangle $R_i \in L'$ is placed in the snapshots $f(R_i)$ so that its bottom is at distance $t_{\alpha_{i-1}}^*$ from the bottom of the bin. Notice that no rectangle from L' is split. To pack the other long rectangles L_{lo}, consider one by one the snapshots $[t_a^*, t_{a+1}^*)$, starting with $[t_0^* = 0, t_1^*)$. For each snapshot $[t_a^*, t_{a+1}^*)$, $m'(a+1)$ is the set of slots used by the rectangles in L'. For this snapshot, we consider all the configurations $c_{F,i}$ with $x_{F,i} > 0$ corresponding to the set of free slots $F = M \setminus m'(a+1)$,

ordered so that all configurations $c_{F,i} = (SN, \Pi)$ with the same set SN appear in consecutive positions. This will ensure that a contiguous block of $|SN|$ slots will be available inside the snapshot to process short-narrow rectangles.

Let $\mathcal{R}_\ell = \{R_{\ell,1}, \ldots, R_{\ell,n_\ell}\} \subseteq L_{lo}$ be the set of long rectangles of length $\ell\delta^2$, for every $\ell = 1, \ldots, 1/\delta^2$. Let y^*_{a+1} be the width of snapshot $[t^*_a, t^*_{a+1})$. Take the first configuration $c_{F,i} = (SN_{F,i}, \Pi_{F,i})$ in the above ordering, for which $x^*_{F,i} > 0$. Then, select for each set $X \in \Pi_{F,i}$ the first not-yet (completely) packed rectangle $R_{\ell,j} \in \mathcal{R}_\ell$ with $\ell = |X|$. This rectangle is packed inside snapshot $[t^*_a, t^*_{a+1})$ in the set of slots X. This can be done, since X is a consecutive set $\{x, x+1, \ldots, x+\ell-1\}$ of slots. Additional rectangles $R_{\ell,j+1}, R_{\ell,j+2}, \ldots$ are packed in the slots X inside snapshot $[t^*_a, t^*_{a+1})$ until either the total width of the selected rectangles is at least $y^* = \min(x^*_{F,i}, y^*_{a+1})$, or all rectangles in $\mathcal{R}_\ell$ are packed. If the total width of the rectangles selected is larger than y^*, then the last selected rectangle is split so that the width of the rectangles is exactly y^*.

The above process is repeated for all sets $X \in \Pi_{F,i}$. If $x^*_{F,i} < y^*_{a+1}$, we set $y^*_{a+1} \leftarrow y^*_{a+1} - x^*_{F,i}$; then we consider the next configuration $c_{F,i'}$ with $x_{F,i'} > 0$ and pack long rectangles as described above. Otherwise, we set $x^*_{F,i} \leftarrow x^*_{F,i} - y^*_{a+1}$ and then continue packing according to configuration $c_{F,i}$ in the next snapshot $[t^*_{a+1}, t^*_{a+2})$.

5.2. Packing the short-narrow rectangles

After (fractionally) packing the long rectangles L_{lo}, the next step is to place the small-narrow rectangles L_{sn} in the empty spaces remaining in the bin. Note that all configurations $c_{F,i}$ with the same first component $SN_{F,i}$ within an interval $[t^*_a, t^*_{a+1})$ leave a reserved area of total width $|SN_{F,i}|\delta^2$ for packing short-narrow rectangles. This reserved area gets split by L' and L_{lo} into at most $|M|/2+1$ rectangular blocks $B_1, B_2, \ldots, B_k$. In each block we pack a subset of short-narrow rectangles as follows.

In those blocks B_j of width $b < 4\delta^{s-1}$ it might not be possible to pack any short-narrow rectangles. Thus, our algorithm will not pack anything there. In the next section we bound the total loss in profit incurred by the algorithm by leaving these blocks empty.

Let B_j have length $d\delta^2$ and width $b \geq 4\delta^{s-1}$. Take short-narrow rectangles off L_{sn} an put them in a set S until their total area is at least $d\delta^2 b$. Since each short-narrow rectangle has area at most δ^{2s}, then the total area of S is $AREA(S) < d\delta^2 b + \delta^{2s}$. We use the First Fit Decreasing Width (FFDW) algorithm [1] to pack the rectangles of S into block B_j. The following result from [1] can be used to determine how many rectangles from S the FFDW algorithm can pack in B_j.

Lemma 5.1 ([1]). *Let S' be a set of rectangles, each of length and width at most Δ. FFDW can pack these rectangles in a rectangular bin of length 1 and width $FFDW(S') \leq AREA(S')(1+\Delta) + \Delta$.*

Since all rectangles in S have width and length at most δ^s, FFDW can pack S into a bin of length $d\delta^2$ and width at most $AREA(S)(1+\delta^s)/(d\delta^2) + \delta^s = (d\delta^2 b + \delta^{2s})(1+\delta^s)/(d\delta^2) + \delta^s \leq b + 3\delta^s$. In other words, if we use FFDW to pack S into block B_j, then a subset of rectangles of total area at most $3\delta^s \times d\delta^2$ might not fit in it. Note that the total profit of these unpacked rectangles could be large; therefore, we need to modify our packing algorithm to ensure that our solution has a large profit. First, we use FFDW to pack all the rectangles of S into a block of length $d\delta^2$ and width $b + 3\delta^s$. Then, divide the block into horizontal strips of width $4\delta^s$. This partitions S into at least $1/\delta$ disjoint groups (in this partition, if a rectangle from S intersects the top side of a strip, then we consider that it belongs to the strip that is above). One of these groups must have profit at most $\delta \times profit(S)$. This low profit group is removed from the packing, so the remaining rectangles fit in block B_j.

5.3. The complete algorithm

Algorithm `RectanglePacking` (R, ϵ)
Input: Set R of rectangles and precision ϵ
Output: A packing for a subset of rectangles of profit $(1-\epsilon)OPT$ in a bin of unit width and length $1+\epsilon$.

1. set $\epsilon' \leftarrow \epsilon/4$ and round down to largest value $1/2a \leq \epsilon'$ for a integer.
2. Repeat Steps 3–12 for each value $\tau \in \{2, \ldots, 4/\epsilon' + 1\}$.
   3. Set $\delta \leftarrow \sigma_\tau$, where σ_τ is as defined in Section 2.1 and set $|M| \leftarrow \frac{1+2\delta}{\delta^2}$. Then, increase the length of the bin to $1+2\delta$.
   4. Partition R into sets $\mathcal{L}$, $\mathcal{M}_\ell$, $\mathcal{H}$, $\mathcal{N}$, $\mathcal{M}_w$, and $\mathcal{W}$, and discard all rectangles in $\mathcal{M}_\ell \cup \mathcal{M}_w$.
   5. Round up the length of every long rectangle to the nearest multiple of δ^2.
   6. Repeat Steps 7–12 for each value $K \in \{1, 2, \ldots, (2|M|)^{|M|+\lceil 1/\epsilon' \rceil + 3}\}$.
      7. Repeat Steps 8–12 for each selection L_K of long rectangles as described in Section 3.1 (that has at least one feasible packing into the augmented bin of width 1 and length $1+2\delta$).
         8. Repeat steps 9–12 for each possible choice $\mathcal{C}^+_{sw}$ of containers and their widths, as described in Section 3.2.
            9. Select a subset of short-narrow rectangles to pack outside the containers of $\mathcal{C}^+_{sw}$, as described in Section 3.2.
            10. Use the algorithm in [11] to select and pack a subset of short rectangles in each container of $\mathcal{C}^+_{sw}$.
            11. Solve the linear programs $LP(f, \alpha, \beta)$ for each possible slot assignment f and snapshot assignments α, β for the containers of $\mathcal{C}^+_{sw}$ and L_K.
            12. If $LP(f, \alpha, \beta)$ has a solution, then pack the long rectangles as described in Section 5.1, and then pack the short-narrow rectangles as described in Section 5.2.
13. Output the packing found of largest profit.

6. Analysis of the algorithm

6.1. Split long rectangles

The number of configurations (SN, Π) is at most $(2|M|)^{|M|}$, since there are $2^{|M|}$ different subsets SN of M and there are $\mathcal{B}_M \leq |M|^{|M|}$ different partitions Π of M, where $\mathcal{B}_M$ is the Mth Bell number [27]. Observe that the algorithm described in the previous section might not produce a valid packing for the long rectangles in L_{lo} since a subset, SPLIT, of these rectangles might have been split into several pieces. Let us remove all the rectangles in SPLIT from the bin, thus obtaining a valid packing. Now, we need to ensure that the profit lost by doing this is small. To determine the profit of SPLIT, let us first bound the number of rectangles in it.

There are two cases when a long rectangle might be split: (1) inside a snapshot, when the packing algorithm changes from one configuration to another one, and (2) at the end of a snapshot. Since there are at most $(2|M|)^{|M|}$ configurations, and when changing from a configuration $c_{F,i} = (SN_{F,i}, \Pi_{F,i})$ to a different configuration $c_{F,i'}$ no more than $|M|$ rectangles can be split (as $\Pi_{F,i}$ has fewer than $|M|$ subsets), then case (1) causes at most $(2|M|)^{|M|+1}$ long rectangles to be split. Case (2) causes at most $g^*(|M|-1) \leq 2(K+\delta^{-3})(|M|-1)$ long rectangles to be split. Therefore, SPLIT has size at most $(2|M|)^{|M|+1} + 2(K+\delta^{-3})(|M|-1) \leq (2|M|)^{|M|+4} + 2K(|M|-1)$. To bound the total profit lost by the removal of these rectangles, we use the following result.

Lemma 6.1 (*[28]*). *Let $p_1 \geq p_2 \geq \cdots \geq p_m$ be a sequence of positive numbers. Let $c = 2(|M|-1)$ and $d = (2|M|)^{|M|+4}$. If $m > (\lceil 1/\epsilon' \rceil d + 1)(c+1)^{\lceil 1/\epsilon' \rceil}$, then there is a constant K, $1 \leq K \leq (2|M|)^{|M|+\lceil 1/\epsilon' \rceil+3}$, such that $p_{K+1} + \cdots + p_{K+cK+d} \leq \epsilon' \sum_{i=1}^{m} p_i$.*

We use this result as follows. If $n \leq (\lceil 1/\epsilon' \rceil d + 1)(c+1)^{\lceil 1/\epsilon' \rceil}$ then we set $K = n$, and hence there will not be any split rectangles. However, if $n > (\lceil 1/\epsilon' \rceil d + 1)(c+1)^{\lceil 1/\epsilon' \rceil}$, then for each possible value $1 \leq K \leq (2|M|)^{|M|+\lceil 1/\epsilon' \rceil+3}$ let L_K be the set of long rectangles selected by our algorithm as described in Section 3.1 and let $p_1 \geq p_2 \geq \cdots \geq p_K$ be the profits of these rectangles. By Lemma 6.1, for one of these values K, $p_{K+1} + \cdots + p_{K+(2|M|)^{|M|+4}+2K(|M|-1)} \leq \epsilon' \sum_{i=1}^{K} p_i$. With this choice of K, $profit(\text{SPLIT}) \leq \epsilon' profit(L_K)$.

Notice that the rectangles in L_K can be packed in an augmented bin of length $1+2\delta$; this is because we consider only sets L_K with a feasible packing. Therefore, L_K can be partitioned into 3 disjoint subsets, each of which can be packed in a unit size square bin (to see this, draw a vertical line through the middle of the bin; this partitions the rectangles into three sets: the rectangles to the left of the line, the rectangles intersected by the line, and the rectangles to the right of the line). Hence, $profit(L_K) \leq 3OPT$, and $profit(SPLIT) \leq \epsilon' \times profit(L_K) \leq 3\epsilon' OPT$.

6.2. Unpacked short-narrow rectangles

Short-narrow rectangles were packed in the blocks B_i as described in Section 5.2, but our algorithm does not pack anything in those blocks B_i that have width smaller than $4\delta^{s-1}$. To bound the total space wasted by the algorithm in these blocks, let us first bound the total number of such blocks.

In the solution of $LP(f, \alpha, \beta)$ each snapshot could have up to $|M|/2+1 \leq |M|$ different blocks reserved for short-narrow rectangles. Thus, there are at most $g^*|M| \leq 2(K+\delta^{-3})|M|$ such blocks. However, when packing long rectangles in the snapshots, we can create up to $|M|/2+1 \leq |M|$ additional blocks when changing from a configuration $c_{F,i} = (SN_{F,i}, \Pi_{F,i})$ to another configuration $c_{F,i'} = (SN_{F,i'}, \Pi_{F,i'})$ with $SN_{F,i} \neq SN_{F,i'}$. Since for each set $F \in \mathcal{F}$ of free slots there are at most $2^{|M|}$ different first components for the configurations $c_{F,i}$, then at most $2^{|M|}2^{|M|}|M| = 2^{2|M|}|M|$ additional blocks can be created. Therefore, in total there are at most $(2K + 2\delta^{-3} + 2^{2|M|})|M|$ blocks for packing short-narrow rectangles.

The total area of blocks of width smaller than $4\delta^{s-1}$ and, thus, the total area wasted by the algorithm in these blocks is then smaller than $4\delta^{s-1}(2K + 2\delta^{-3} + 2^{2|M|})|M|$. Using the above bound for K, this area is at most $4\delta^{s-1}[2(2|M|)^{|M|+\lceil 1/\epsilon' \rceil+3} + 2\delta^{-3} + 2^{2|M|}]|M| \leq 4(4/\delta^2)^{4/\delta^2}\delta^{s-1}$, where the last inequality follows from the definition of δ. Furthermore, $(4/\delta^2)^{4/\delta^2} = (2/\delta)^{8/\delta^2} < (1/\delta)^{9/\delta^2-2}$ since $\delta \leq (1/2)^{10}$. Hence, the total area is at most

$$4(1/\delta)^{9/\delta^2-2}\delta^{s-1} = 4\delta^{s-1-(9/\delta^2)+2} = 4\delta^{s-9/\delta^2+1} \leq \delta^{s-9/\delta^2} \leq \delta, \quad \text{as } s = 9/\delta^2 + 1.$$

This means that short-narrow rectangles of total area at most δ might not be packed by the algorithm. These rectangles can be packed to the right of the bin using the FFDW algorithm. This increases the length of the bin by at most $\delta + \delta^s < 2\delta$. The total length of the bin is, then, at most $1+4\delta$.

6.3. Rectangle packing without rotations

Here, we prove Theorem 1.1 for packing rectangles without rotations. Algorithm `RectanglePacking` produces a large number of packings, the best of which is finally selected by the algorithm. Consider this largest profit packing and the iteration of the algorithm in which it is computed. In Step 7 of the algorithm a set L_K of long rectangles of total profit at

least $(1-2\delta)p_L^+$ is selected, where p_L^+ is the profit of the long rectangles in a near-optimum solution S^+ as described in Corollary 2.1. In Step 10 the algorithm selects and packs in the containers a set of short rectangles of profit at least $(1-\epsilon')p_C^+$, where p_C^+ is the total profit of the short rectangles packed in containers in S^+. In Step 9 a set of short-narrow rectangles of profit at least p_{sn}^+ is chosen, where p_{sn}^+ is the total profit of the short-narrow rectangles in S^+ packed outside containers.

In Step 12 the long rectangles selected in Step 7 are packed in the bin and a subset of profit at most $\epsilon' p_L^+$ is discarded. Therefore, the total profit of the rectangles that are packed in the augmented bin $[0,1]\times[0,1+4\delta]$ is at least $(1-2\delta)p_L^+ + (1-\epsilon')p_C^+ + p_{sn}^+ - \epsilon' p_L^+ - \delta p_{sn}^+ \geq (1-3\epsilon')(p_L^+ + p_C^+ + p_{sn}^+) = (1-3\epsilon')profit(S^+) \geq (1-4\epsilon')OPT = (1-\epsilon)OPT$, where OPT is the profit of an optimum solution.

The complexity of the algorithm is $O(n^{(4/\delta^2)^{5/\delta^2}})$, where $\delta = O(1)$, as defined in (3).

6.4. Rectangle packing with 90° rotations

Now, let us consider the case where 90° rotations are allowed. Note that Corollary 2.1 is still valid, assuming that the rectangles have been oriented as in an optimum solution S^*. However, the selection process described in Section 3 is more complicated as now we need to determine the orientation for the rectangles.

In Sections 3.1 and 3.2 we used a knapsack algorithm (as the algorithm in [11] uses also a knapsack algorithm to select the wide rectangles to pack in a container) to select the sets of long and wide rectangles to be packed in the bin. Now, instead of a knapsack algorithm we use a multiple choice knapsack algorithm [18] to select these rectangles. In the multiple choice knapsack problem we are given a set of items i_j grouped into disjoint classes N_k. Each item i_j has a profit p_j and a weight w_j and the goal is to select a maximum profit set of items of total weight at most a given value K, such that at most one item from each class is selected. For our rectangle selection problem the set of items corresponds to the set R of rectangles. For each rectangle R_j we create a class N_j consisting of 2 items: R_j and R_j rotated 90°. The PTAS in [18] can choose a set of rectangles of nearly maximum total profit and bounded area, and it can also decide whether each selected rectangle must be rotated or not.

In Section 3.1 we need to select the set of at most $1/\delta^4$ widest rectangles in each set $\bar{L}_{Ki}^*$. We can do this by enumeration as we did in the case without rotations, but now for each rectangle R_i we consider two cases: when it is rotated and when it is not rotated.

For the short-narrow rectangles the situation is easier. Since we use an area argument to pack these rectangles (see Sections 3.2 and 5.2), a rotation of these rectangles is not necessary. Once the rectangles to be packed and their orientations have been selected, they are packed in the bin as described in Sections 4 and 5. This proves Theorem 1.1 for the case with rotations.

7. Strip packing problem

7.1. APTAS for strip packing without rotations

Our algorithm for the rectangle packing problem can be used to design an asymptotic PTAS for the strip packing problem without rotations. The best known algorithm for this problem, by Kenyon and Rémila [2], produces a solution of length at most $(1+\epsilon)OPT_{SP} + O(1/\epsilon^2)$ for any $\epsilon > 0$, where OPT_{SP} is the length of an optimum solution. Our algorithm achieves a better additive constant by producing a solution of length at most $(1+\epsilon)OPT_{SP} + 1$, but at the expense of a much higher running time. Our algorithm for strip packing works as follows. Let R be the set of rectangles that needs to be packed in the strip.

1. Use the algorithm of Steinberg [4] to pack R in a strip of length $v \leq 2OPT_{SP}$. Hence, $v/2 \leq OPT_{SP} \leq v$. Set $\varepsilon' \leftarrow \epsilon/5$.
2. Consider all values $v' = v/2, (1+\varepsilon')v/2, (1+2\varepsilon')v/2, \ldots, v$. Note that one of these values v^* is such that $OPT_{SP} \leq v^* \leq (1+\varepsilon')OPT_{SP}$. For each value v' we scale the lengths of the rectangles by multiplying them by $1/v'$ and define the profit of each rectangle as its area; then, we use our rectangle packing algorithm to pack a subset of scaled rectangles into a bin of unit width and length.

   Observe that when $v' = v^*$ there is a way of packing all scaled rectangles into an augmented bin and, thus, by Theorem 1.1 for this value of v' our algorithm must be able to pack in the bin most of the rectangles; the scaled rectangles that our algorithm cannot pack have total area at most ε'.
3. Find v^*, the smallest value v' such that in the packing produced by our algorithm the total area of the set S of un-packed (scaled) rectangles is at most ϵ.
4. Use the algorithm of Steinberg to pack S in a strip of width 1 and length at most $\epsilon + 1/v^*$.
5. In the packing produced in Step 2 for $v' = v^*$, scale the rectangles back to their original lengths and append to this packing the packing created in Step 4.

The total length of the packing produced is at most $v^*(1+2\varepsilon'+1/v^*) = (1+2\varepsilon')v^* + 1 \leq (1+\varepsilon')(1+2\varepsilon')OPT_{SP} + 1 \leq (1+5\varepsilon')OPT_{SP} + 1 = (1+\epsilon)OPT_{SP} + 1$, for $\varepsilon' \leq 1$. The running time of this algorithm is dominated by our rectangle packing algorithm. This proves Theorem 1.2 for strip packing without rotations.

7.2. APTAS and PTAS for strip packing with 90° rotations

We give now an approximation scheme for the strip packing problem with rotations. The previously best known algorithm for this problem, by Jansen and van Stee [21], produces a solution of value at most $(1+\epsilon)OPT_{SP} + O(1/\epsilon^2)$ for any $\epsilon > 0$, where OPT_{SP} is the length of an optimum solution. As before, we assume that each rectangle has width and length at most 1. First we use the Next Fit Decreasing Width (NFDW) algorithm to obtain a strip packing of length $v \leq 2\,Area(R) + \max_{R_i \in R} \min(\ell_i, w_i) \leq 3OPT_{SP}$. Note that $v/3 \leq OPT_{SP} \leq v$. We set $\varepsilon' = (\epsilon/4)^2$ and consider all values $v' = v/3, v/3(1+\varepsilon'), v/3(1+\varepsilon')^2, \ldots, v$. For one of these values v^* we have $OPT_{SP} \leq v^* \leq (1+\varepsilon')OPT_{SP}$.

For each value v' we construct a bin of unit width and length v'. Note that our rectangle packing algorithm works with minor modifications also on bins that do not have unit length. Again when $v' = v^*$ our algorithm packs almost all rectangles into an augmented bin of width 1 and length $v^*(1+\varepsilon')$; only a subset S of total area at most $v^*\varepsilon'$ is not packed. Then use NFDW to pack S in a strip of width 1 and length at most $2\varepsilon' v^* + \min(1, v^*)$ (if $v^* < 1$ then the additive term is the length of the largest rectangle). The total length of the solution is at most $v^*(1+3\varepsilon')+\min(1, v^*) \leq (1+\varepsilon')(1+3\varepsilon')OPT_{SP}+\min(1, v^*) \leq (1+4\varepsilon'+3\varepsilon'^2)OPT_{SP} + \min(1, v^*) \leq (1+7\varepsilon')OPT_{SP} + \min(1, (1+\varepsilon')OPT_{SP}) \leq (1+\epsilon)OPT_{SP} + 1$. This proves Theorem 1.2 for the strip packing problem with 90° rotations.

Now let us consider the interesting case where the optimum value $OPT_{SP} \geq 1$. Here the above set S of rectangles can be packed into a strip of width 1 and length $2\varepsilon' v^* + (\varepsilon')^{1/2} v^*$. Note that for each rectangle in S the scaled length or width is at most $(\varepsilon')^{1/2}$. Since we can rotate these rectangles (of length and width is at most 1), the original length is at most $(\varepsilon')^{1/2} v^*$ or the width is at most $(\varepsilon')^{1/2} \leq (\varepsilon')^{1/2} OPT_{SP} \leq (\varepsilon')^{1/2} v^*$. This gives a total length of at most $v^*(1+3\varepsilon'+(\varepsilon')^{1/2}) \leq (1+\varepsilon')(1+3\varepsilon'+(\varepsilon')^{1/2})OPT_{SP} \leq (1+5\varepsilon'+(\varepsilon')^{1/2})OPT_{SP} \leq (1+(8/9)\epsilon)OPT_{SP} \leq (1+\epsilon)OPT_{SP}$. This proves our Theorem 1.3.

Notice that a similar result holds when $OPT_{SP} \geq c$ and $c > 0$ is a constant. However, for the general case it is not possible to design a PTAS unless P = NP:

Lemma 7.1. *There is no approximation algorithm fsaor the strip packing problem with* 90° *rotations with absolute approximation ratio smaller than* 3/2 *unless* $P = NP$.

Proof. To prove the lemma we show that if there exists an algorithm for strip packing with 90° rotations with absolute approximation ratio smaller than 3/2, then that algorithm could be used to solve the partition problem. In the partition problem we are given a set $S = \{s_1, s_2, \ldots, s_n\}$ of $n > 1$ positive integer values and the goal is to decide whether there is a subset $S' \subset S$ such that $\sum_{s_i \in S'} s_i = B$, where $B = \frac{1}{2}\sum_{s_i \in S} s_i$. Without loss of generality we might assume that $s_i \leq B$, for all $i = 1, 2, \ldots, n$. Consider an instance S of the partition problem. For this instance we generate the following set $R = \{R_1, R_2, \ldots, R_n\}$ of rectangles, where rectangle R_i has width s_i/B and length $\ell = \min\{s_j/(8B^2) \mid s_j \in S\}$. Since $s_i \geq 1$, for all $i = 1, 2, \ldots, n$, then $\ell \geq 1/(8B^2)$ and $\ell \leq B/(8B^2) = 1/(8B) < 1/B \leq \min\{s_j/B \mid s_j \in S\}$; hence, the length of each rectangle is smaller than its width.

Assume first that there is a solution for the instance S of the partition problem, i.e. there is a subset $S' \subset S$ such that $\sum_{s_i \in S'} s_i = \sum_{s_i \in S \setminus S'} s_i = B$. In this case the rectangles in R can be packed (without having to rotate them) in a strip of unit width and length 2ℓ, as all rectangles corresponding to the elements in S' can be packed at the bottom of the strip, while the remaining rectangles can be packed on top of them.

On the other hand, if there is no solution for the instance S of the partition problem, then the rectangles must be packed in a strip of length at least 3ℓ. To see this, observe that if at least one rectangle R_i is rotated by 90°, then a strip of length at least $s_i/B \geq 1/B > (3s_i)/(8B^2) \geq 3\ell$ would be needed to pack the rectangles. If no rectangle is rotated and there is no solution for the partition problem, then the best packing must have length 3ℓ. Hence, an algorithm for the strip packing problem with 90° rotations with approximation ratio smaller than 3/2 must be always able to decide whether there is a solution for any instance of the partition problem. □

Acknowledgements

We kindly thank the anonymous referees for their valuable comments that helped us improve this paper.

The first author's research was supported in part by the EU project AEOLUS, contract number 015964 and by the DAAD German Academic Exchange Service. The second author's research was supported in part by the Natural Science and Engineering Research Council of Canada grant R3050A01.

References

[1] E.G. Coffman, M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM Journal on Computing 9 (1980) 808–826.
[2] C. Kenyon, E. Remila, A near-optimal solution to a two-dimensional cutting stock problem, Mathematics of Operations Research 25 (2000) 645–656.
[3] I. Schiermeyer, Reverse-fit: A 2-optimal algorithm for packing rectangles, in: European Symposium on Algorithms, in: LNCS, vol. 855, 1994, pp. 290–299.
[4] A. Steinberg, A strip-packing algorithm with absolute performance bound two, SIAM Journal on Computing 26 (1997) 401–409.

[5] N. Bansal, A. Caprara, M. Sviridenko, Improved approximation algorithms for multidimensional bin packing problems, in: Symposium on Foundations of Computer Science, 2006, pp. 697–708.
[6] N. Bansal, M. Sviridenko, Two-dimensional bin packing with one dimensional resource augmentation, Discrete Optimization 4 (2007) 143–153.
[7] N. Bansal, J.R. Correa, C. Kenyon, M. Sviridenko, Bin packing in multiple dimensions: Inapproximability and approximation schemes, Mathematics of Operations Research 21 (1) (2006) 31–49.
[8] A. Caprara, Packing 2-dimensional bins in harmony, in: IEEE Symposium on Foundations of Computer Science, 2002, pp. 490–499.
[9] D.R. Karger, K. Onak, Polynomial approximation schemes for smoothed and random instances of multidimensional packing problems, in: ACM–SIAM Symposium on Discrete Algorithms, 2007, pp. 1207–1216.
[10] B.S. Baker, A.R. Calderbank, E.G. Coffman, J.C. Lagarias, Approximation algorithms for maximizing the number of squares packed into a rectangle, SIAM Journal on Algebraic and Discrete Methods 4 (1983) 383–397.
[11] A.V. Fishkin, O. Gerber, K. Jansen, On efficient weighted rectangle packing with large resources, in: International Symposium on Algorithms and Computation, 2005, pp. 1039–1050.
[12] A.V. Fishkin, O. Gerber, K. Jansen, R. Solis-Oba, Packing weighted rectangles into a square, in: Symposium on Mathematical Foundation of Computer Science, in: LNCS, vol. 3618, 2005, pp. 352–363.
[13] K. Jansen, G. Zhang, On rectangle packing: Maximizing benefits, in: ACM–SIAM Symposium on Discrete Algorithms, 2004, pp. 197–206.
[14] K. Jansen, G. Zhang, Maximizing the number of packed rectangles, in: Scandinavian Workshop on Algorithm Theory, 2004, pp. 362–371.
[15] N. Bansal, X. Han, K. Iwama, M. Sviridenko, G. Zhang, Harmonic algorithm for 3-dimensional strip packing problem, in: ACM–SIAM Symposium on Discrete Algorithms, 2007, pp. 1197–1206.
[16] K. Jansen, R. Solis-Oba, An asymptotic approximation algorithm for 3D-strip packing, in: ACM–SIAM Symposium on Discrete Algorithms, 2006, pp. 143–152.
[17] J.Y.-T. Leung, T.W. Tam, C.S. Wong, G.H. Young, F.Y.L. Chin, Packing squares into a square, Journal of Parallel and Distributed Computing 10 (1990) 271–275.
[18] E. Lawler, Fast approximation algorithms for knapsack problems, Mathematics of Operations Research 4 (1979) 339–356.
[19] R. Harren, Approximating the orthogonal knapsack problem for hypercubes, in: International Colloquium on Automata, Languages and Programming, in: LNCS, vol. 4051, 2006, pp. 238–249.
[20] M.R. Garey, D.S Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.
[21] K. Jansen, R. van Stee, On strip packing with rotations, in: ACM Symposium on Theory of Computing, 2005, pp. 755–761.
[22] W. Fernandez de la Vega, G.S. Lueker, Bin packing can be solved within 1+epsilon in linear time, Combinatorica 1 (1981) 349–355.
[23] M. Karmarkar, R.M. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in: IEEE Symposium on Foundations of Computer Science, 1982, pp. 312–320.
[24] B. Johannes, Scheduling parallel jobs to minimize the makespan, Journal of Scheduling 9 (2006) 433–452.
[25] A.K. Amoura, E. Bampis, C. Kenyon, Y. Manoussakis, Scheduling independent multiprocessor tasks, Algorithmica 32 (2002) 247–261.
[26] S.V. Sevastianov, G.J. Woeginger, Makespan minimization in open shops: A polynomial time approximation scheme, Mathematical Programming 82 (1998) 191–198.
[27] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics: A Foundation for Computer Science, Addison-Wesley, 1989.
[28] K. Jansen, L. Porkolab, Improved approximation schemes for scheduling unrelated parallel machines, Mathematics of Operations Research 26 (2001) 324–338.