INSTITUT FÜR INFORMATIK

How to Maximize the Total Area of Rectangles Packed into a Rectangle?

Klaus Jansen, Lars Prädel

Bericht Nr. 0908
March 2009

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

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

How to Maximize the Total Area of Rectangles Packed into a Rectangle?

Klaus Jansen, Lars Prädel

Bericht Nr. 0908
March 2009

e-mail: kj@informatik.uni-kiel.de, lap@informatik.uni-kiel.de

Work 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"

Abstract

We study an interesting geometric optimization problem. We are given a set of rectangles and a rectangular target area called bin. The goal is to find a feasible packing of a subset of the given rectangles into the bin, i.e. an orthogonal packing without rotation and overlap. The objective is to maximize the total area of rectangles packed. This problem is strongly $\mathcal{NP}$-hard even for squares, therefore there is no fully polynomial time approximation scheme (FPTAS) for this problem, unless $\mathcal{P} = \mathcal{NP}$. The previously best result is a $(1/2 - \varepsilon)$-approximation by Jansen & Zhang for our problem. We present a polynomial time approximation scheme (PTAS) for this problem, i.e. a family of algorithms which compute for any accuracy $\varepsilon > 0$ in polynomial time a solution with ratio $(1 - \varepsilon)$.

1 Introduction

We study a problem called Rectangle Packing with Area Maximization (RPA). Here we are given a set of rectangles $L = \{r_1, \ldots, r_n\}$ of specified widths w_i and heights h_i and a larger rectangle called *bin*. By applying some scaling we can assume w.l.o.g. that the bin has height and width 1. The goal is to find a feasible packing, i.e. an orthogonal non-rotational arrangement where rectangles do not overlap.

RPA is a generalization of the Subset Sum problem, which is one of the most fundamental and well-known problems in combinatorial optimization [11, 15, 10]. RPA is a generalization of the Square Packing problem which is strongly $\mathcal{NP}$-hard [14]. In this problem, we have to determine whether there is an orthogonal packing of a given set of squares.

The RPA problem has many practical applications. In VLSI layout many objects have to be placed on a chip. In the advertisement placement problem rectangular advertisements or articles have to be placed in a newspaper or on a website. A further industrial application is the cutting problem. Here we have to cut some items out of a given material.

Since RPA is $\mathcal{NP}$-hard there exists no efficient optimal algorithm, unless $\mathcal{P} = \mathcal{NP}$; hence we focus on approximation algorithms. Let $A(I)$ be the output of a polynomial-time algorithm A and $Opt(I)$ the optimal value for an instance I of a maximization problem X. The absolute approximation ratio of A is $\sup_I \frac{A(I)}{Opt(I)}$. The asymptotic approximation ratio of A is $\limsup_{Opt(I)\mapsto\infty} \frac{A(I)}{Opt(I)}$. X admits a polynomial-time approximation scheme (PTAS) if there is a family of algorithms $\{A_\varepsilon | \varepsilon > 0\}$ such that for any $\varepsilon > 0$ and any instance I of X, A_ε produces a $(1-\varepsilon)$-approximate solution in time polynomial in the encoding length of the instance. A fully polynomial-time approximation scheme (FPTAS) is a PTAS where additionally A_ε has run-time polynomial in $1/\varepsilon$. Asymptotic (fully) polynomial-time approximation schemes (APTAS, AFPTAS) are similarly defined in terms of asymptotic approximation ratio.

Results: RPA is strongly $\mathcal{NP}$-hard so it does not admit an FPTAS, unless $\mathcal{P} = \mathcal{NP}$. Hence we focus on a PTAS. The Rectangle Packing problem (RP) is a generalization of RPA. Here the rectangles additionally have arbitrary profits and the goal is to maximize the total profit of the packed rectangles. For RP an $(1/2 - \varepsilon)$-approximation was found by Jansen & Zhang [9]. Instances which contain only squares were discussed by Harren [4] obtaining a $(4/5 - \varepsilon)$-approximation and by Jansen & Solis-Oba [6] obtaining a PTAS.

For maximizing the number of squares in a bin there is an asymptotic $(3/4)$-approximation algorithm by Baker et al. [1] and a PTAS found by Jansen & Zhang [8]. Fishkin et al. [3] obtained a PTAS for RPA (maximizing the area) restricted to instances which contain squares.
A different problem which is closely related is the Strip Packing problem. Here the goal is to find a feasible packing of all of the given rectangles into a strip of width 1 and infinite height in order to minimize the height of the packing. The classical approximation algorithms by Coffman et al. [2], Next-Fit Decreasing Height (NFDH) and First-Fit Decreasing Height (FFDH), pack rectangles on shelves. The authors showed that NFDH has an absolute approximation ratio of 3 and FFDH one of 2.7. Independently, Schiermeyer [16] and Steinberg [17] developed algorithms with an absolute worst-case ratio of 2. Kenyon & Rémila [12] found an AFPTAS with additive constant of $O(1/\varepsilon^2)$, later Jansen & Solis-Oba [5] found an APTAS with additive constant of 1. The latter algorithm uses a PTAS for RP where the bin is augmented in one direction by ε. We present the following result that improves the previous best $(1/2 - \varepsilon)$-approximation algorithm for RPA:

Theorem 1. *There is a polynomial-time approximation scheme (PTAS) for RPA.*

Techniques: We study two cases depending on whether the optimal value is small or large. We discuss the case with small objective value in the appendix. In the second case we partition the instance into big, long, wide, medium and small rectangles. We are able to delete the medium rectangles and lose only a little amount in the objective function. For the remaining arrangement we are able to establish a suitable gap structure. After deleting the small and wide rectangles we construct gap-rectangles for the wide rectangles. In a second step we construct long gap-rectangles by repacking the rounded wide gap-rectangles and removing the long rectangles. In general, the number of wide and long gap-rectangles could be linear in the number of rectangles in the instance. The main geometric method of our paper is to reduce the number of wide and long gap-rectangles to a constant. This can be done by considering horizontal strips of height δ, which depends on ε. Then we analyze the geometric packing in the horizontal strips and using an inductive argument to obtain the desired bound. Finally, we use a modified version of Kenyon & Rémila [12] to place the original long and wide rectangles into the gap-rectangles.
Organization of the paper: In Section 2 we partition the instance in big, wide, long, small and medium rectangles. In this section we prove that we are able to delete the medium rectangles and get an instance of rectangles with either large or small sides. We modify an optimal solution in Section 3, there we build so-called gap-rectangles which replace the long and wide rectangles. These gap-rectangles can be merged so that a constant number of them remains in the solution, which can then be guessed. In Section 4 we pack these gap-rectangles with selected long and wide rectangles. In Section 5 the small rectangles are packed into the remaining gaps with a NFDH algorithm.

2 Partitioning

Here we partition the rectangles of L in big, long, wide, small and medium rectangles and discard the medium rectangles. We have to ensure that the medium rectangles used in a fixed optimal solution have a small total area. We achieve this similar to the partitioning in [5]. Let for the fixed precision ε

$$\varepsilon' \leq \min\{1/2, \varepsilon/4\}$$

be the largest value such that $1/\varepsilon'$ is an even integer.
Define a sequence $\sigma_1, \ldots, \sigma_{\frac{4}{\varepsilon'}+1}$ by

$$\sigma_1 = \varepsilon' \text{ and } \sigma_k = \sigma_{k-1}^{12/\sigma_{k-1}+8}$$

for $k \in \{2, \ldots, 4/\varepsilon' + 1\}$. For a given $k \in \{2, \ldots, 4/\varepsilon' + 1\}$ we partition the instance into

- $L_{B_k} = \{r_i | r_i \in L \wedge h_i > \sigma_k \wedge w_i > \sigma_k\}$ the set of *big* rectangles,
- $L_{W_k} = \{r_i | r_i \in L \wedge h_i \leq \sigma_{k+1} \wedge w_i > \sigma_k\}$ the set of *wide* rectangles,
- $L_{L_k} = \{r_i | r_i \in L \wedge h_i > \sigma_k \wedge w_i \leq \sigma_{k+1}\}$ the set of *long* rectangles,
- $L_{S_k} = \{r_i | r_i \in L \wedge h_i \leq \sigma_{k+1} \wedge w_i \leq \sigma_{k+1}\}$ the set of *small* rectangles and
- $L_{M_k} = \{r_i \in L | w_i \in (\sigma_{k+1}, \sigma_k] \vee h_i \in (\sigma_{k+1}, \sigma_k]\}$ the set of *medium* rectangles.

For the analysis we need a fixed optimal solution, called OPT, of the instance. For a given sublist S of rectangles in L we denote by $S^* = S \cap \text{OPT}$ the list of rectangles which belong to S and the optimal solution OPT. Let $x_i, y_i \in [0,1]$ denote the coordinates of the lower left corners of the rectangles $r_i \in S^*$. Furthermore let $A(S)$ be the total area of the rectangles in S. Hence $A(L^*)$ is the optimal value $Opt(L)$ of the instance L.

Lemma 1. *There exists a $k \in \{2, \ldots, 4/\varepsilon' + 1\}$ such that $A((L_{M_k})^*) \leq \frac{\varepsilon'}{2} Opt(L)$.*

Proof. Each rectangle in L^* belongs to at most two sets $(L_{M_k})^*$, so we have

$$\sum_{k \in \{2, \ldots, 4/\varepsilon'+1\}} A((L_{M_k})^*) \leq 2 \cdot Opt(L).$$

Assume $A((L_{M_k})^*) > \frac{\varepsilon'}{2} Opt(L)$ for all $k \in \{2, \ldots, 4/\varepsilon' + 1\}$. Then we would obtain

$$\sum_{k \in \{2, \ldots, 4/\varepsilon'+1\}} A((L_{M_k})^*) > \frac{4}{\varepsilon'} \cdot \frac{\varepsilon'}{2} Opt(L) = 2 \cdot Opt(L),$$

which is a contradiction. □

This shows that the area loss of one particular L_{M_k} set is small enough. We try all $4/\varepsilon'$ possibilities for the value k for deleting the set L_{M_k}. Let k' be the value where the output of our algorithm has the largest area. Furthermore let

$$\delta := \sigma_{k'} \text{ and } s := \frac{12}{\delta} + 8.$$

For simplicity we call the sets L_B, L_W, L_L, L_S, L_M instead of $L_{B_{k'}}, L_{W_{k'}}, L_{L_{k'}}, L_{S_{k'}}, L_{M_{k'}}$ and the optimal subsets $L_B^*, L_W^*, L_L^*, L_S^*, L_M^*$ instead of $(L_B)^*$,$(L_W)^*$,$(L_L)^*$,$(L_S)^*$,$(L_M)^*$. We are able to guess a value δ, depending on ε, and to delete medium rectangles where the area loss is at most $\frac{\varepsilon'}{2} Opt(L)$. Furthermore we are able to guess the optimal set of big rectangles, because each of them has an area of at least δ^2 and there are at most $1/\delta^2$ big rectangles in the optimal solution OPT. We enumerate all combinations of at most $1/\delta^2$ big rectangles. This can be done polynomial in time $n^{O(1/\delta^2)}$.

3 Gap-Rectangles

In contrast to the big rectangles it is not possible to guess the wide and long rectangles in the optimal solution OPT. Instead we use so-called *gap-rectangles* in which we pack long and wide rectangles.
We have the following steps: We construct gap-rectangles, round them, repack them and merge some of them.
Constructing: Consider the fixed optimal solution OPT without the small rectangles S_L^*. Take the wide rectangles out of the solution and draw a horizontal line at the bottom and top edge of each long and each big rectangle, until it hits another rectangle. These lines build rectangles $g_1, \ldots, g_m$ which we call *wide gap-rectangles* (see Figure 1).
W.l.o.g. we assume that $g_1, \ldots, g_m$ are sorted in non-increasing order of widths. We eliminate all gap-rectangles with width less than δ, because we cannot pack wide rectangles into them. With this elimination we have wide gap-rectangles $g_1, \ldots, g_{m'}$ left.
We repack the wide rectangles into the solution. Each wide rectangle is contained in wide gap-rectangles. It is possible that some wide rectangles are contained in more than one gap-rectangle. Therefore we have to cut the wide rectangles horizontally into pieces.

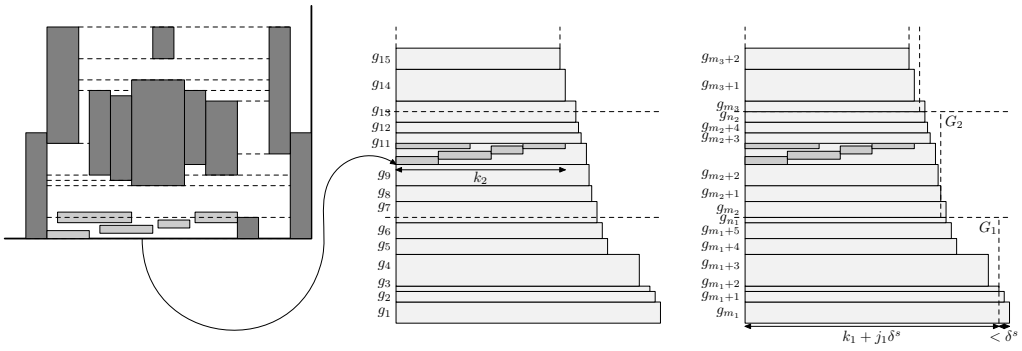

Figure 1: Constructing and rounding of gap-rectangles

Rounding: Take the gap-rectangles out of the solution and pack them on a stack sorted

in non-increasing order of their widths. Using the rounding technique from [12], we divide the stack into $1/\delta^2$ groups of equal height, possibly splitting gap-rectangles in that process. W.l.o.g. we assume that gap-rectangles $g_{m_i}, \ldots, g_{n_i}$ are contained in group i. The stack has a height of at most $1/\delta$, because each gap-rectangle has width at least δ and it would otherwise cover an area of more than 1. Using this bound, each group has height at most δ.

We round the width of all gap-rectangles in one group to the same value as described below. To this end we shift the wide rectangles inside the gap-rectangles horizontally to the left side. Hence the width of the used area inside the gap-rectangles is a combination of the widths of at most $1/\delta$ wide rectangles. We compute for each gap-rectangle the maximal width of the used area of the wide rectangles. The maximal value of these widths in each group i is called k_i (see Figure 3). Then we compute for each group i a value j_i so that $k_i + j_i \cdot \delta^s \leq w_{m_i} < k_i + (j_i + 1) \cdot \delta^s$ whereas w_{m_i} is the width of the widest gap-rectangle in this group. Furthermore we round the width of each gap-rectangle in this group to $k_i + j_i \cdot \delta^s$ (see Figure 1). This generates an area loss on the right side of at most $\delta^s \cdot 1/\delta$, because the stack has a height of at most $1/\delta$.

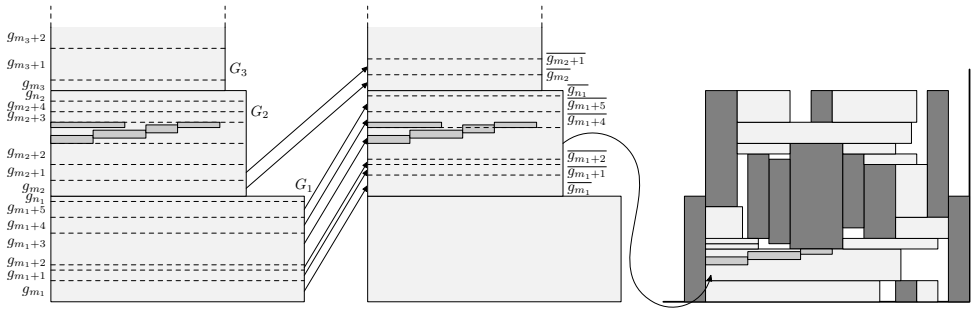

Figure 2: Repacking of gap-rectangles

Repacking: We merge the gap-rectangles in each group to get $1/\delta^2$ group-rectangles $G_1, \ldots, G_{1/\delta^2}$ (see Figure 2).

In the next section we explain how to create a solution of wide rectangles which are not fractionalized inside the group-rectangles. But for now we only handle the gap-rectangles. We cut the group-rectangles again so that the newly created gap-rectangles fit into the solution. To this end we delete the widest group-rectangle G_1 and repack only the remaining group-rectangles $G_2, \ldots, G_{1/\delta^2}$. The deleted group-rectangle has an area of at most δ, because its height is at most δ and its width is at most 1.

We cut each remaining group-rectangle G_i horizontally into $n_{i-1} - m_{i-1}$ parts so that part ℓ has height of gap-rectangle g_ℓ, for $\ell \in \{m_{i-1}, \ldots, n_{i-1}\}$. This new gap-rectangle $\overline{g_\ell}$ has the same height but a width smaller than or equal to the width of g_ℓ. Hence we are able to repack $\overline{g_\ell}$ at the coordinates of g_ℓ (see Figure 2).

With these rounding steps our algorithm is able to guess the widths of each group-rectangle. To this end we have to select for each group at most $1/\delta$ wide rectangles for computing the value k_i and try $1/\delta^s$ values for computing j_i.

We do the same steps for the long rectangles. This means that we take the long rectangles out of the solution and build long gap-rectangles by drawing vertical lines at the right

and left edge of each wide gap-rectangle and big rectangle. Then we round the heights of these gap-rectangles in the same way.
Merging: After constructing, rounding and repacking the wide and long gap-rectangles we want to merge as many gap-rectangles of the same group as possible. For this we try to shift wide gap-rectangles, so that two rectangles on top of each other start at the same x-coordinate. Then we merge them if they are in the same group. In the same way we merge and shift two wide gap-rectangles with different widths, if we are able to round up the width of the shorter gap-rectangle to the width of the other gap-rectangle. Similarly we shift long gap-rectangles vertically and round the heights if possible.
After these steps we shift all long gap-rectangles vertically down so that their y-coordinates are positioned on top of one big rectangle or a wide gap-rectangle. This is possible because two long gap-rectangles are not on top of each other.
With this discussion we obtain the following result.

Lemma 2. *We construct and round gap-rectangles which contain horizontally fractionalized wide rectangles and vertically fractionalized long rectangles by losing an area of at most* $2 \cdot \delta + 2 \cdot \delta^{s-1}$. *The algorithm is able to guess the rounded widths of the wide gap-rectangles and the rounded heights of the long gap-rectangles, by trying all possible values.*

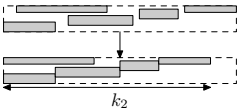

Figure 3: Shifting the wide rectangles

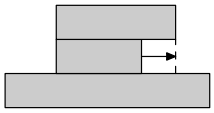

Figure 4: Rounding up in the merging step

Now we estimate the total number of gap-rectangles in this solution. To this end we divide the bin in horizontal *strips* of height δ and width 1. Then we divide these strips into *squares* of side length δ. In the following we count the number of bottom-left corners of gap-rectangles in each strip. We say that a gap-rectangle *starts* in some position if the bottom-left corner of this rectangle is in this position. It *ends* in some position if the top-right corner is in this position.
We bound the number of wide gap-rectangles in a $\delta \times \delta$ square in the lowest strip. Then we estimate the largest possible number of long gap-rectangles starting there. With this bound we estimate the largest possible number of wide gap-rectangles in the strip above and so on.
We denote the number of different widths of the wide gap-rectangles by K, so $K := 1/\delta^2 - 1$ and the number of different heights of the long gap-rectangles by K' with the same value. By rounding into the K different widths we obtain the following lemma.

Lemma 3. *Let R be a $\delta \times \delta$ square in the lowest strip. There are at most K wide gap-rectangles starting and ending in R.*

Proof. Wide gap-rectangles are horizontally non-adjacent because they originally touch a long or big rectangle at their construction. Hence wide gap-rectangles are positioned on

top of each other and not side by side.
The wide gap-rectangles in R are positioned completely on top of each other forming a tower (see Figure 5). Assume a wide gap-rectangle is on top of another wide gap-rectangle with smaller width. Then there is a gap below the upper wide gap-rectangle where no long gap-rectangle or big rectangle fits. Hence we round up the width of the lower gap-rectangle to the width of the other one and merge them. Assume two wide gap-rectangles on top of each other with the same width would start at different x-coordinates. Then there is again a gap below the upper gap-rectangle and hence we are able to shift the lower gap-rectangle to the x-coordinate of the upper gap-rectangle and merge them.
By this construction each tower contains one gap-rectangle of each width. So there are at most K wide gap-rectangles starting and ending in each $\delta \times \delta$ square. □

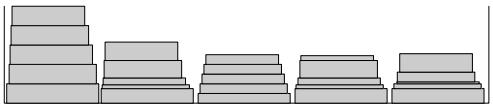

Figure 5: Wide gap-rectangles in the lowest strip

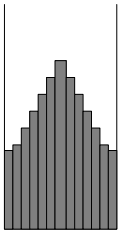

Figure 6: Starting long gap-rectangles on an edge

This result implies that at most $\frac{K}{\delta}$ wide gap-rectangles are in the lowest strip. Now we estimate the largest possible number of long gap-rectangles starting on an edge of width less or equal than δ. This edge could be the bottom of the bin, the top edge of a wide gap-rectangle or the top edge of a big rectangle.

Lemma 4. *Let E be a horizontal edge of width less or equal than δ. There are at most $2K'$ long gap-rectangles starting on E.*

Proof. Long gap-rectangles starting on an edge of width less or equal than δ form a mountain with one peak, where each side consists of at most K' long gap-rectangles (see Figure 6). For each length there start at most two long gap-rectangles and so there start at most $2K'$ long gap-rectangles in total. In case of a long gap-rectangle placed between two taller long gap-rectangles there is a gap between the two taller gap-rectangles where no wide gap-rectangle or big rectangle fits. Hence there is an empty space and we can round up the height of the shorter gap-rectangle and merge it with one of its neighbours. □

Now assume that a fixed number of wide gap-rectangles are in a $\delta \times \delta$ square starting or ending there. We estimate the largest possible number of long gap-rectangles which start in this square.

Lemma 5. *Let R be a $\delta \times \delta$ square with at most M starting or ending wide gap-rectangles. Then there are at most $K'(2M + 6)$ long gap-rectangles starting in R.*

Proof. Each top edge of the M wide gap-rectangles could form an edge of width less than δ on which long gap-rectangles could start. Hence we obtain by Lemma 4 at most $2K'$ long gap-rectangles starting on each of these M wide gap-rectangles. Furthermore there are at most two big rectangles in this square on which long gap-rectangles could start. Then there is at most one wide gap-rectangle which intersects this square without starting and ending on which long gap-rectangles could start or the bottom of the bin. Hence we have at most three more edges. In total we obtain at most $M + 3$ edges with width less or equal than δ and hence at most $2K' \cdot (M + 3) = K' \cdot (2M + 6)$ long gap-rectangles starting in this square. □

With the next lemma we bound the total number of long gap-rectangles starting in the whole strip.

Lemma 6. *Let S be a strip with at most $M > K$ wide gap-rectangles. There are at most $5M \cdot K'$ long gap-rectangles starting in S.*

Proof. Let M_i denote the number of wide gap-rectangles which start or end in the i-th $\delta \times \delta$ square R_i of the strip. We have $\sum_{i=1}^{1/\delta} M_i = 2M$. By using Lemma 5 for each R_i there are at most $K' \cdot (2M_i + 6)$ starting long gap-rectangles. In total there are at most $\sum_{i=1}^{1/\delta} K' \cdot (2M_i + 6) = K' \cdot \left(4M + \frac{6}{\delta}\right)$ starting long gap-rectangles in this strip. Since $M \geq K = \frac{1}{\delta^2} - 1 > \frac{6}{\delta}$ we bound this number by $K' \cdot \left(4M + \frac{6}{\delta}\right) < 5M \cdot K'$. □

In addition we can bound the number of wide gap-rectangles in one strip, if we know the number of long gap-rectangles which intersect the strip from below. This means that we know the number of long gap-rectangles in this strip except long gap-rectangles which start there. We try to shift all wide gap-rectangles in a strip so that they horizontally touch a long gap-rectangle which starts below this strip, a big rectangle or the wall of the bin. This is not always possible because there might be wide gap-rectangles which horizontally touch on both sides long gap-rectangles starting in this strip. Nevertheless we are able to bound this number.

Lemma 7. *Let S be a strip which intersects at most $M' \geq K'$ long gap-rectangles starting below this strip. Then there are at most $5KM'$ wide gap-rectangles in S.*

Proof. It is possible that wide gap-rectangles horizontally touch only long gap-rectangles which start in this strip S. These wide gap-rectangles have to build towers similar to the wide gap-rectangles in the first strip (see Lemma 3). If they do not form towers then it is possible to round up or shift some of these wide gap-rectangles and merge them. Hence we obtain similar to Lemma 3 that the number of all wide gap-rectangles in S which horizontally touch only long gap-rectangles starting in S is bounded by K/δ.
The remaining wide gap-rectangles in S touch to the left or right either a long gap-rectangle starting below this strip, a big rectangle, or the wall of the bin. Let M'_i denote the number of long gap-rectangles which start in a strip below and intersect the ith square of S. On the right side of these M'_i long gap-rectangles could start at most $2K$ wide gap-rectangles by symmetrically using Lemma 4. Furthermore on the left side of these M'_i long gap-rectangles could end at most $2K$ wide gap-rectangles. This leads us to at most $4M'_iK$ wide gap-rectangles starting and ending on these M'_i long gap-rectangles.

There are at most two big rectangles or the wall of the bin in this square. So we have two additional edges of height less or equal than δ. On these edges start or end at most $2K$ additional wide gap-rectangles. We obtain that there are at most $K\,(4M'_i + 4)$ wide gap-rectangles starting or ending on the right or left side of long gap-rectangles which starts from a strip below, big rectangles or the wall of the bin.
In total we have at most $\frac{K}{\delta} + \sum_{i=1}^{1/\delta} K\,(4M'_i + 4) = K \cdot \left(4M' + \frac{5}{\delta}\right)$ wide gap-rectangles in S. Since $M' \geq K' = \frac{1}{\delta^2} - 1 > \frac{5}{\delta}$ we bound this number by $5KM'$. □

Now we are able to bound the number of wide gap-rectangles in one specified strip.

Lemma 8. *In the i-th strip there are at most $\frac{K}{\delta} \cdot (26KK')^{i-1}$ wide gap-rectangles.*

Proof. Let $n_i(w)$ be the number of wide gap-rectangles in strip i and let $n_i(h)$ be the number of long gap-rectangles starting in strip i. By Lemma 6 we obtain $n_i(h) \leq n_i(w) \cdot 5 \cdot K'$ and by Lemma 7 we obtain $n_i(w) \leq \sum_{j=1}^{i-1} n_j(h) \cdot 5K$ since at most $\sum_{j=1}^{i-1} h_i$ long gap-rectangles end in the i-th strip.
We prove by induction over the strips that $n_i(w) \leq (26KK')^{i-1} \cdot n_1(w)$.
By Lemma 6 we obtain $n_1(h) \leq n_1(w) \cdot 5K'$, by Lemma 7 we obtain

$$n_2(w) \leq n_1(h) \cdot 5K \leq n_1(w) \cdot 5K' \cdot 5K = 25KK' \cdot n_1(w) \leq 26KK' \cdot n_1(w).$$

Hence there are at most $26KK'$ wide gap-rectangles in the second strip.
By using an inductive argument we suppose $n_j(w) \leq (26KK')^{j-1} \cdot n_1(w)$ for all $j \in \{2, \ldots, i-1\}$.
For $i \geq 3$ we have

$$\begin{aligned} n_i(w) &\leq \sum_{j=1}^{i-1} n_j(h) \cdot 5K = n_{i-1}(h) \cdot 5K + \sum_{j=1}^{i-2} n_j(h) \cdot 5K \leq n_{i-1}(w) \cdot 5K \cdot 5K' + n_{i-1}(w) \\ &= n_{i-1}(w) \cdot (25KK' + 1) \leq (26KK')^{i-2} \cdot n_1(w) \cdot 26KK' = (26KK')^{i-1} \cdot n_1(w). \end{aligned}$$

By Lemma 3 is $n_1(w) \leq \frac{1}{\delta} \cdot K$ therefore there are at most $\frac{K}{\delta} \cdot (26KK')^{i-1}$ wide gap-rectangles in the i-th strip. □

The same bound holds for the number of long gap-rectangles. By using geometric sum arguments we obtain:

Theorem 2. *We have at most $\frac{1}{\delta^{8/2-2}}$ gap-rectangles in the modified optimal solution OPT.*

Proof. By Lemma 8 we obtain at most

$$\sum_{i=1}^{1/\delta} \frac{K}{\delta} \cdot (26KK')^{i-1} = \frac{K}{\delta} \sum_{i=0}^{1/\delta - 1} (26KK')^i$$

wide gap-rectangles in the solution. Since $K = K'$ we have the same number for long gap-rectangles by symmetrically using Lemma 8. Thus we have with the geometric progression

$$\begin{aligned} \frac{2K}{\delta} \sum_{i=0}^{1/\delta - 1} (26KK')^i = \frac{2K}{\delta} \sum_{i=0}^{1/\delta - 1} \left(26K^2\right)^i &\leq \frac{2K}{\delta} \cdot \frac{(26K^2)^{1/\delta} - 1}{(26K^2) - 1} \\ &\leq \frac{2K}{\delta} \cdot \left(\left(26K^2\right)^{1/\delta} - 1\right) \leq \frac{2K}{\delta} \cdot \left(26K^2\right)^{1/\delta}. \end{aligned}$$

Using $s = \frac{12}{\delta} + 8$ and $\delta < \frac{1}{26}$, we get

$$\frac{2K}{\delta} \cdot \left(26K^2\right)^{1/\delta} \leq \frac{2}{\delta^3} \cdot \left(\frac{26}{\delta^4}\right)^{1/\delta} \leq \frac{1}{\delta^4} \cdot \left(\frac{1}{\delta^5}\right)^{1/\delta} = \left(\frac{1}{\delta}\right)^{5/\delta+4}$$
$$\leq \left(\frac{1}{\delta}\right)^{6/\delta} \leq \left(\frac{1}{\delta}\right)^{\frac{1}{2}\cdot(s-4)-2} \leq \left(\frac{1}{\delta}\right)^{\frac{1}{2}\cdot(s-4)} = \frac{1}{\delta^{s/2-2}}.$$

□

As our main result we have bounded the total number of gap-rectangles by a constant. Interestingly we can round the height of each wide gap-rectangle and the width of each long gap-rectangle to a multiple of δ^s. In this step we lose for each wide gap-rectangle the amount of the rounding, which is at most δ^s, plus the maximal height of rectangles inside the gap-rectangles, which is also δ^s. Similar this bound holds for rounding the widths of the long gap-rectangles. To this end we need a solution of rectangles which are not fractionalized inside the gap-rectangles. We consider this in the next section. In total we have an area loss of at most $\delta^{2-s/2} \cdot 2\delta^s = 2\delta^{s/2+2}$. With this we obtain the following lemma which gives the complete area loss in this section.

Lemma 9. *By creating gap-rectangles with rounded widths and heights we lose a total area of at most* 3δ*.*

Proof. In Lemma 2 we lose an area of at most $2\delta + 2\delta^{s-1}$. In rounding the heights of the wide gap-rectangles and the widths of the long gap-rectangles we have an area loss of at most $2\delta^{s/2+2}$. Altogether we lose an area of at most

$$2\delta + 2\delta^{s-1} + 2\delta^{s/2+2} \leq 2\delta + \delta^{s/2+2} + 2\delta^{s/2+2} \leq 2\delta + \delta = 3\delta.$$

□

Now our algorithm is able to guess the number of gap-rectangles in the solution and the widths and heights of them. In total we are able to guess the whole structure of the modified optimal solution with the gap-rectangles and the big rectangles. To do so we guess a bottom-left orientated solution, where we allocate coordinates to each gap-rectangle and each big rectangle. In the next section we take a closer look at the rectangles which are replaced by the gap-rectangles. We construct in this section a solution of long and wide rectangles which are not fractionalized inside the gap-rectangles.

4 Wide and Long Rectangles

We construct a solution of non-fractionalized wide and long rectangles inside the gap-rectangles in two steps. In the first step we construct a solution of them inside the group-rectangles. The group-rectangles were constructed by the rounding step of the gap-rectangles (see Section 3).
In the second step we have to delete all cut rectangles inside the group-rectangles when we split them to construct the rounded gap-rectangles. We have at most $\frac{1}{\delta^{s/2-2}}$ gap-rectangles

in the solution. Hence we have at most $\frac{1}{\delta^{s/2-2}}$ cutting lines inside the group-rectangles. We delete the rectangles split by these cutting lines. The wide rectangles are split horizontally and the long rectangles are split vertically. Therefore by deleting the fractionalized rectangles we lose an area of at most $\frac{2}{\delta^{s/2-2}} \cdot \delta^s$. The first step is more complicated.
The next steps will be explained for the wide rectangles. The construction for the long rectangles is analogous. Consider the group-rectangles $G_1, \ldots, G_{1/\delta^2}$ which contain a fractional solution of wide rectangles. We round the wide rectangles similarly to the gap-rectangles. This means that we take all wide rectangles out of the group-rectangles and put them on a stack sorted in non-increasing order of widths (see Figure 1). We cut the stack into $m := 1/\delta^2$ parts, denoted by $P_1, \ldots, P_m$. The rectangles which are cut in this step will be deleted. This generates an area loss of at most $m \cdot \delta^s$. Next we round the widths of all rectangles in part P_i to the width of the widest rectangle, denoted by w_{P_i}. It is possible that several parts have same widths. We merge them and hence we have $m' \leq m$ parts left.
Now repack the rounded wide rectangles into the group-rectangles. Since we have rounded widths we have to delete the rectangles in the widest part P_1 so that the remaining rectangles fit.
We are able to guess the width of the rounded rectangles. To this end we have to select m rectangles out of n. Furthermore we are able to round the height of each part. We delete the topmost rectangles in part P_i such that the height is a multiple of δ^s denoted by h_{P_i}. This generates an area loss in each part of at most $2 \cdot \delta^s$ because we round down at most δ^s and delete the newly cut rectangles with height at most δ^s.

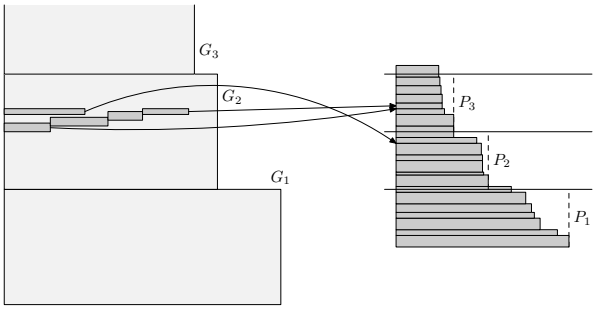

Figure 7: The stack of the wide rectangles

In addition we are able to guess the heights of the group-rectangles. To this end we round down the height of each group-rectangle $G_2, \ldots, G_{1/\delta^2}$ to a multiple of δ^s. In order not to lose some rectangles we increase the height of group-rectangle G_1 by at most $1/\delta^2 \cdot \delta^s$. Later we delete this additional area when we delete the group-rectangle G_1.
Now we change the point of view. Until now we considered a packing of the optimal solution OPT. Our algorithm has to select a subset of wide rectangles which we have to pack into the group-rectangles. For this we sort the wide rectangles by nonincreasing widths and the wide rectangles with the same width by nonincreasing heights. Then we guess the height h_{P_i} and the width w_{P_i} of each part P_i for $i \in \{1, \ldots, m'\}$. We fill each part greedily with the sorted wide rectangles. We have an area loss of at most δ^s in each part because otherwise we could pack a wide rectangle with height less than δ^s into this

part. Hence we select wide rectangles with an area of $A\left(L_W^*\right) - \delta^{s-2}$ using $m' \leq \frac{1}{\delta^2}$ parts. Now we round up the widths of the selected wide rectangles in the ith part according to the width of the optimal part w_{P_i}. Then we use a generalization of the fractional strip packing algorithm by Kenyon & Rémila [12] with $1/\delta^2$ target areas instead of one. This technique is already used in [7]. For each group-rectangle G_ℓ a *configuration* $C_j^{(\ell)}$ is a set of wide rectangles with total width less or equal than the width of G_ℓ. We denote by $q^{(\ell)}$ the number of possible configurations for this group-rectangle. For each configuration $C_j^{(\ell)}$ let $a(i,C_j^{(\ell)})$ be the number of wide rectangles of width w_{P_i}. We solve the feasibility problem of the following Linear Program. The variable $x_j^{(\ell)}$ describes the height of configuration $C_j^{(\ell)}$ in group-rectangle G_ℓ.

$$\sum_{j=1}^{q^{(\ell)}} x_j^{(\ell)} \leq h_{G_\ell} \qquad \forall \ell \in \{1,\ldots,1/\delta^2\}$$

$$\sum_{\ell=1}^{\frac{1}{\delta^2}} \sum_{j=1}^{q^{(\ell)}} a\left(i,C_j^{(\ell)}\right) \cdot x_j^{(\ell)} \geq h_{P_i} \qquad \forall i \in \{2,\ldots,m'\}$$

$$x_j^{(\ell)} \geq 0 \qquad \forall \ell \in \{1,\ldots,1/\delta^2\} \text{ and } \forall j \in \{2,\ldots,m'\}$$

The first constraint asserts that the total height of the configurations in group-rectangle G_ℓ do not exceed the height of G_ℓ. So all configurations fit into the group-rectangles. The second constraint asserts that the total height of all wide rectangles with width w_{P_i} is larger or equal than the height h_{P_i}. Hence all wide rectangles are packed fractionally into the group-rectangles.
A basic solution has at most $(m'-1) + 1/\delta^2 \leq 2/\delta^2$ non-zero variables. We pack $a(i,C_j^{(\ell)})$ rectangles with width w_{P_i} and height $x_j^{(\ell)}$ into the group-rectangle G_ℓ greedily. Since we might not reach exactly this amount we lose for each packed configuration the topmost wide rectangles. Hence we have an area loss of at most $2/\delta^2 \cdot \delta^s$. By bounding the total area loss in this section by 3δ we obtain the following lemma.

Lemma 10. *Our algorithm selects wide and long rectangles and packs them into gap-rectangles. The selected rectangles have an area of at least* $A\left(L_L^*\right) + A\left(L_W^*\right) - 3\delta$.

5 Small Rectangles

We select a subset of small rectangles with an FPTAS for Subset Sum [13] with precision ε' (see Section 2). The instance consists of a set of items whose sizes are equal to the areas of the small rectangles. The capacity is equal to the remaining space in the bin with the generated solution which contains the big, wide and long rectangles. This remaining space is an upper bound for the optimal value of the small rectangles $A\left(L_S^*\right)$, because the selected rectangles have smaller or equal total area than the optimal selected rectangles. We divide the selected small rectangles into three parts: one subset will be packed into the remaining space in the wide gap-rectangles, one into the remaining space of the long

gap-rectangles and one subset will be packed into the remaining space of the bin. We pack the small rectangles with Next Fit Decreasing Height [2]. Since the number of gaps is constant and the rectangles are small, each side has length of at most δ^s, we lose only a small area. For the details we refer to Appendix A. We get the following result.

Lemma 11. *Our algorithm selects and packs small rectangles with an area of at least* $(1-\varepsilon')A(L_S^*) - \delta$.

6 Algorithm

First we describe the entire algorithm for our packing problem. Then we prove that the area of the selected rectangles is close to the optimal value.

1. Guess the values δ and δ^s and partition the rectangles of L in big, wide, long, small and medium rectangles. Then discard the medium rectangles (see Section 2).
2. Guess the big rectangles of the optimal solution OPT (see Section 2).
3. Guess the set of the gap-rectangles with their heights and widths (see Section 3).
4. Guess the packing of the gap-rectangles and the big rectangles (see Section 3).
5. Guess the heights and widths of the parts in the stacks with the wide and long rectangles. Furthermore guess the heights of the group-rectangles (see Section 4).
6. Select wide and long rectangles greedily as described in Section 4.
7. Solve the feasibility version of a Linear Program to pack the wide and long rectangles into the gap-rectangles (see Section 4).
8. Use an FPTAS for Subset Sum for selecting small rectangles and pack them with NFDH in the gaps (see Section 5).

Since all guessing steps are polynomial in n our algorithm runs in polynomial time in the length of the instance. The total area loss is given in the next theorem.

Theorem 3. *Let L be an instance with $Opt(L) \geq \varepsilon^2$. Our algorithm selects a subset of rectangles in L and computes a solution with area of at least $(1-\varepsilon)Opt(L)$.*

Proof. In Section 2 we bound the area of the discarded rectangles $A(L_M^*)$. By creating the gap-rectangles we lose an area of at most 3δ and by selecting and packing wide and long rectangles into them we obtain an area of at least $A(L_W^*) + A(L_L^*) - 3\delta$. We select all big rectangles by guessing them and obtain $A(L_B^*)$. Using Lemma 11 we obtain a solution of small rectangles with an area of at least $(1-\varepsilon') \cdot A(L_S^*) - \delta$. We have a solution with area at least

$$A(L_L^*) + A(L_W^*) + A(L_B^*) + A(L_S^*) \cdot (1-\varepsilon') - A(L_M^*) - 7\delta$$
$$\geq (1-\varepsilon')A(L^*) - 2A(L_M^*) - 7\delta.$$

Using Lemma 1 we obtain

$$(1-\varepsilon')A(L^*) - 2A\left(L^*_M\right) - 7\delta \geq (1-\varepsilon')A(L^*) - 2\cdot\frac{\varepsilon'}{2}A(L^*) - 7\delta$$
$$= (1-2\varepsilon')A(L^*) - 7\delta \geq \left(1-\frac{\varepsilon}{2}\right)A(L^*) - 7\delta.$$

Using the definition of δ in Section 2, $\varepsilon' \leq 1/2$ and $A(L^*) \geq \varepsilon^2$ we obtain

$$7\delta < 7\left(\frac{\varepsilon}{4}\right)^{12/\varepsilon'+8} \leq 7\left(\frac{\varepsilon}{4}\right)^{32} < \frac{1}{2}\cdot\varepsilon^{32} = \frac{\varepsilon}{2}\cdot\varepsilon^{31} \leq \frac{\varepsilon}{2}\cdot A(L^*).$$

This implies

$$\left(1-\frac{\varepsilon}{2}\right)\cdot A(L^*) - 7\delta \geq \left(1-\frac{\varepsilon}{2}\right)\cdot A(L^*) - \frac{\varepsilon}{2}\cdot A(L^*) = (1-\varepsilon)\cdot A(L^*) = (1-\varepsilon)\cdot Opt(L).$$

□

If the total area of the selected rectangles is very small then we use an additional algorithm (see Appendix B).

Theorem 4. *Let L be an instance with $Opt(L) < \varepsilon^2$. Our algorithm selects a subset of rectangles in L and computes a solution with area of at least $(1-\varepsilon)\cdot Opt(L)$.*

References

[1] B. S. Baker, A. R. Calderbank, E. G. Coffman, Jr., and J. C. Lagarias. Approximation algorithms for maximizing the number of squares packed into a rectangle. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):383–397, 1983.

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

[3] A. V. Fishkin, O. Gerber, K. Jansen, and R. Solis-Oba. Packing weighted rectangles into a square. In *Mathematical Foundations of Computer Science*, pages 352–363, 2005.

[4] R. Harren. Approximating the orthogonal knapsack problem for hypercubes. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4051 of *LNCS*, pages 238–249. Springer, 2006.

[5] K. Jansen and R. Solis-Oba. New approximability results for 2-dimensional packing problems. In *Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2007.

[6] K. Jansen and R. Solis-Oba. A polynomial time approximation scheme for the square packing problem. In *Integer Programming and Combinatorical Optimization*, pages 184–198, 2008.

[7] K. Jansen and R. Thöle. Approximation algorithms for scheduling parallel jobs: Breaking the approximation ratio of 2. In *International Colloquium on Automata, Languages and Programming*, pages 234–245, 2008.

[8] K. Jansen and G. Zhang. Maximizing the number of packed rectangles. In *Scandinavian Workshop on Algorithm Theory*, pages 362–371, 2004.

[9] K. Jansen and G. Zhang. On rectangle packing: maximizing benefits. In *Algorithmica*, volume 4, pages 323–342, 2007.

[10] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Tatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[11] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems.* Springer, 2004.

[12] C. Kenyon and E. Rémila. A near optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25:645–656, 2000.

[13] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operation Research*, 4(4):339–356, 1979.

[14] J.Y.-T. Leung, T.W. Tam, C.S. Wong, G.H. Young, and F.Y.L. Chin. Packing squares into a square. *Journal of Parallel Distributed Computation*, 10(3):271–275, 1990.

[15] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, 1990.

[16] I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the Second Annual European Symposium on Algorithms*, pages 290–299. Springer-Verlag, 1994.

[17] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal of Computing*, 26(2):401–409, 1997.

A Small Rectangles

Lemma 11. *Our algorithm selects and packs small rectangles with an area of at least* $(1-\varepsilon')A\left(L_S^*\right)-\delta$.

For proving this lemma we need the following results.

Lemma 12. *We select small rectangles with an area of at least* $(1-\varepsilon')\cdot A\left(L_S^*\right)-3\delta^{2s}$.

Proof. We select a subset of small rectangles with an FPTAS for Subset Sum [13] with precision ε' (see Section 2). The instance consists of a set of items whose sizes are equal to the areas of the small rectangles. The capacity is equal to the remaining space in the bin with the generated solution which consists of big, wide and long rectangles. This remaining space is an upper bound for the optimal value of the small rectangles $A\left(L_S^*\right)$, because the selected rectangles have smaller or equal total area than the optimal selected rectangles. We divide the selected small rectangles into three parts: One subset will be packed into the remaining space in the wide gap-rectangles, one into the remaining space of the long gap-rectangles and one subset will be packed into the remaining space of the bin. Since we are not able to divide this set exactly so that the rectangles in each part fit, we delete three rectangles and hence an area of at most $3\delta^{2s}$. □

We use a Next Fit Decreasing Height algorithm for packing the rectangles, whereas we pack the rectangles in more than one bin. This extension is straightforward. We show a general result for packing small rectangles into some specified areas.

Lemma 13. *Let* $A_1,\ldots,A_\ell$ *be a set of areas with heights* H_i *and widths* W_i *for all* $i\in\{1,\ldots,\ell\}$. *The area loss for packing small rectangles with Next Fit Decreasing Height into these areas is less than* $\delta^s\sum_{i=1}^{\ell}\left(H_i+2W_i\right)$

Proof. We assume that there are small rectangles which cannot be packed into one of these areas. Otherwise we are able to pack all small rectangles and therefore we have for this part an optimal solution for the selected small rectangles.
In the following steps we compute the remaining space in the areas after packing. This is a bound for the area loss.
At the right side of each area, we have a free space of at most δ^s. Otherwise we could pack there a small rectangle, because each small rectangle has a width less or equal than δ^s. There we have a free space of at most $\delta^s\cdot H_i$ for every area A_i.
In the first step of the algorithm we sort the rectangles by height. Hence the height of one level, this means the height of the first rectangle, is at most the height of the smallest rectangle in the previous level. Hence each level covers the area of the following level, except some space at the right side of at most δ^s, which we already estimated above. In other words, the area of packed rectangles in each level is larger than the total area of the following level. So we cover in each area every level but the first one and thus we lose an area of $\delta^s W_i$ for every area A_i. At the top of each area is also a free space of at most δ^s. Altogether we have a free space of at most

$$\sum_{i=1}^{\ell}\left(\delta^s\cdot H_i+2\delta^s\cdot W_i\right)=\delta^s\sum_{i=1}^{\ell}\left(H_i+2W_i\right).$$

□

Lemma 14. *While packing small rectangles into the gap-rectangles we have an area loss of at most* $14 \cdot \delta^{s-2}$.

Proof. The total height of the areas is bounded by $\frac{1}{\delta}$, because the heights of the stacks with the gap-rectangles has at most this value (see Section 3). We have filled the $\frac{1}{\delta^2}$ group-rectangles with at most $\frac{2}{\delta^2}$ configurations (see Section 4). Thus we have $\frac{3}{\delta^2}$ areas with free space. Since the width of each area is less than 1 the total width of all areas is less than $\frac{3}{\delta^2}$.
By Lemma 13 we obtain an area loss bounded by $\delta^s \left(1/\delta + 2 \cdot 3/\delta^2\right) \leq \delta^s \cdot 7/\delta^2$. The same holds for the space in the long gap-rectangles by interchanging widths and heights. We obtain a total area loss of at most $2\delta^s \cdot 7/\delta^2 = 14\delta^{s-2}$. □

Now we pack the rectangles into the remaining space beyond the gap-rectangles in the bin. We bound the total number of areas with free space.

Lemma 15. *The number of areas with free space in the bin is less than* $\frac{8}{\delta^{s/2-2}}$.

Proof. Consider a bin with k rectangles. Extend the top edge and the bottom edge of each rectangle by a horizontal line until it hits another rectangle. The bin is now separated in rectangular areas. We have at most $2k$ horizontal lines. Each line, which was drawn at the bottom of one rectangle is the top edge of one rectangular area. Furthermore the line is the bottom edge of two rectangular areas, at the left and at the right side. Each line at the top edge of a rectangle is the bottom edge of one rectangular area. Furthermore the top edge of two rectangular areas. The bottom of the bin is also a bottom edge of a rectangular area and the top wall a top edge of a rectangular area. So we have $3k + 1$ bottom and top edges of rectangles and so we have at most $3k + 1 \leq 4k$ rectangular free areas.
There are at most $\frac{1}{\delta^2}$ big rectangles in the solution and $\frac{1}{\delta^{s/2-2}}$ gap-rectangles (see Theorem 2). With this we have at most $4 \cdot \left(\frac{1}{\delta^2} + \frac{1}{\delta^{s/2-2}}\right) \leq \frac{8}{\delta^{s/2-2}}$ rectangular areas with free space in the bin. □

We estimate the width and height of each area with 1. With this we obtain the following lemma.

Lemma 16. *We have an area loss of at most* $24 \cdot \delta^{\frac{4+s}{2}}$ *by packing small rectangles into the remaining gaps of the bin.*

Proof. We have at most $\frac{8}{\delta^{s/2-2}}$ rectangular areas in the bin with free space left (see Lemma 15). We know that the width and the height of each area is bounded by 1. So we have by Lemma 13 an area loss of at most

$$\delta^s \cdot 8 \cdot \delta^{\frac{4-s}{2}} \cdot 3 = 24 \cdot \delta^{\frac{4-s}{2}} \cdot \delta^s = 24\delta^{\frac{4+s}{2}}.$$

□

The total sum of the area loss is less than δ and with this Lemma 11 follows.

B Small Optimal Value

In this section we will show the following theorem:

Theorem 4. *Let L be an instance with $Opt(L) < \varepsilon^2$. Our algorithm selects a subset of rectangles in L and computes a solution with area of at least $(1-\varepsilon) \cdot Opt(L)$.*

Here we assume that ε is sufficiently small, i.e. $\varepsilon \leq \frac{1}{24}$. Otherwise we could run the algorithms with precision of $1/24$ and obtain a better result. We consider also an optimal solution OPT. We divide the instance into three parts, the set

- $L_S := \{r_i \in L | w_i \leq 1-3\varepsilon \wedge h_i \leq 1-3\varepsilon\}$ called *small rectangles*,
- $L_W := \{r_i \in L | w_i > 1-3\varepsilon\}$ called *wide rectangles* and
- $L_L := \{r_i \in L | h_i > 1-3\varepsilon\}$ called *long rectangles*.

This is a partition of L, because there exists no rectangle in $L_W \cap L_L$, since it would have an area larger than $(1-3\varepsilon) \cdot (1-3\varepsilon) \geq 1 - \frac{6}{24} + \varepsilon^2 \geq \varepsilon^2$.
We take the small rectangles out of the optimal solution OPT. Hence we have only long and wide rectangles in the solution. We are able to shift the long and wide rectangles to the border of the bin, so that a free space of $1-2\varepsilon \times 1-2\varepsilon$ is in the middle of the bin. In this free space we will pack the small rectangles later. Furthermore we are able to exchange rectangles, so that the wider and longer rectangles are positioned at the border of the bin (see Figure 8).
Let m be the smallest integer such that $m \geq \frac{4}{(1-3\varepsilon)\varepsilon}$. Furthermore let $(L_B)^*$ be the set of $\frac{32m}{\varepsilon}$ rectangles from $(L_L)^* \cup (L_W)^*$, the optimal subset of $L_L \cup L_W$, with largest area. Our algorithm guesses these rectangles by selecting at most $\frac{32m}{\varepsilon}$ rectangles out of n. With this we bound the area of the remaining rectangles.

Lemma 17. *Each rectangle of $((L_L)^* \cup (L_W)^*) \setminus (L_B)^*$ has an area of at most $\frac{\varepsilon}{32m} Opt(L)$.*

Proof. Assume a rectangle $r_j \in ((L_L)^* \cup (L_W)^*) \setminus (L_B)^*$ has area larger than $\frac{\varepsilon}{32m} Opt(L)$. It follows that $A(\{r_i\}) \geq \frac{\varepsilon}{12m} Opt(L)$ holds for all $r_i \in (L_B)^*$ and hence

$$A\left((L_B)^*\right) + A(\{r_j\}) \geq \left(\frac{32m}{\varepsilon} + 1\right) \cdot \frac{\varepsilon}{32m} Opt(L) > Opt(L).$$

is a contradiction.
Thus the remaining rectangles have area less than $\frac{\varepsilon}{32m} Opt(L)$. □

In total we obtain the following result.

Theorem 5. *We are able to modify the optimal solution OPT so that all wide and long rectangles are positioned at the border of the bin. The wide rectangles have non-increasing widths and the long rectangles non-increasing heights towards the middle of the bin. We round the widths and heights of these rectangles, but $\frac{32m}{\varepsilon}$ rectangles of largest area, to m widths and heights with an area loss of at most $7/8 \cdot \varepsilon \cdot Opt(L)$.*

Proof. We modify the structure of the optimal packing OPT with long and wide rectangles, so that all rectangles are positioned at the border of the bin. The heights of the long rectangles and the widths of the wide rectangles are decreasing towards the middle of the bin.
We repeat the rounding and repacking step from the gap-rectangles in Section 3.
Hence we put the wide rectangles which are not in $(L_B)^*$ on a stack of height H. We group the rectangles on the stack into $m+1$ groups of same height. Here we discard the splitted rectangles by losing an area of at most $m\frac{\varepsilon}{32m}Opt(L)$. We round up the width of each rectangle to the width of the widest rectangle in each group. Then we discard the group with the largest width from the solution, so the remaining rounded rectangles still fit into the bin. We lose an area of at most

$$\frac{H}{m+1} < \frac{H}{m} \leq \frac{H}{\frac{4}{(1-3\varepsilon)\varepsilon}} = \frac{H(1-3\varepsilon)\varepsilon}{4} \leq \frac{Opt(L)\cdot\varepsilon}{4},$$

because the optimal value is larger than the height of the stack times $1-3\varepsilon$.
Then we repack the rounded rectangles into the space of the rectangles in the next wider group. Consider that in the optimal solution the wide rectangles in the same group are positioned on top of each other at the bottom or at the top of the bin, because we ordered them by width. Hence only rectangles of $(L_B)^*$ are between wide rectangles of the same group at one side of the bin. Since we are able to resort the rounded wide rectangles by width, again we can assume that the wide rectangles of the same group are on top of each other at the bottom and at the top of the bin. When we repack the rounded wide rectangles there is at most one rectangle splitted, because one part of the group is positioned at the bottom and the remaining part is positioned at the top of the bin. We remove these rectangle hence we lose again an area of at most $m\frac{\varepsilon}{32m}Opt(L)$.
We merge all wide rectangles of the same group at the bottom and at the top of the bin so there remain at most $2m$ wide rectangles. We round the heights of these merged wide rectangles down to a multiple of $\frac{\varepsilon}{32m}\cdot\varepsilon^2$. In this step we lose the area of the rounding plus one newly splitted rectangle. Hence we lose for all $2m$ merged rectangles

$$2m\cdot\left(\frac{\varepsilon}{32m}\cdot\varepsilon^2+\frac{\varepsilon}{32m}Opt(L)\right) = \frac{2m\varepsilon}{32m}\cdot\left(\varepsilon^2+Opt(L)\right) < \frac{\varepsilon}{16}\cdot 2Opt(L) = \frac{\varepsilon\cdot Opt(L)}{8}.$$

We do the same steps with the long rectangles. Hence we have a total area loss of at most

$$2\left(\frac{\varepsilon\cdot Opt(L)}{32}+\frac{\varepsilon\cdot Opt(L)}{4}+\frac{\varepsilon\cdot Opt(L)}{32}+\frac{\varepsilon\cdot Opt(L)}{8}\right) = \frac{7}{8}\cdot\varepsilon\cdot Opt(L).$$

□

Now we change the point of view. Our algorithm is able to guess the widths of the groups by selecting m rectangles out of n. Furthermore we are able to guess the heights of the merged rectangles. Hence we have $2m$ merged wide and long rectangles and the guessed $\frac{32m}{\varepsilon}$ rectangles of $(L_B)^*$. We have a solution with a constant number of rectangles which can be placed by guessing all different possibilities. The selection of wide and long rectangles to pack into these merged rectangles is left. We sort the wide rectangles by

width and the wide rectangles with the same width by height. The long rectangles will be ordered by height and these with same height by width. Then we fill them into the $4m$ merged rectangles greedily similar to the selection of the wide and long rectangles in Section 4. We lose for each of the $4m$ groups at most one rectangle. Hence we lose $4m \cdot \frac{\varepsilon}{32m}$. We obtain the following lemma:

Lemma 18. *We select rectangles with an area of at least* $A\left((L_L)^* \cup (L_W)^*\right) - \frac{\varepsilon}{8} \cdot Opt(L)$.

The small rectangles will be packed into the middle of the bin.

Lemma 19. *We pack all small rectangles.*

Proof. We divide the rectangles of L_S into three parts. The first part is the set of the rectangles, with height $h_i \geq \varepsilon$. The rectangles in the second part have heights $h_i < \varepsilon$, but widths $w_i \geq \varepsilon$ and the third part contains the remaining rectangles.
The rectangles in the first part will be packed into an area of $\varepsilon \times (1 - 3\varepsilon)$. We are able to pack the long rectangles side by side into this area, because if they have a total width larger than ε they fill a total area of more than ε^2.
The rectangles of the second part will be packed into an area of $(1 - 3\varepsilon) \times \varepsilon$. Since the rectangles have a width between $\varepsilon \leq w_i \leq (1 - 3\varepsilon)$ we stack them on top of another. Symmetrically as in case of the first part the total height is less than ε.
The remaining rectangles have widths $w_i < \varepsilon$ and heights $h_i < \varepsilon$. We pack these rectangles into an area of $(1 - 3\varepsilon) \times 20\varepsilon$ with NFDH by using Lemma 13 eith ε instead of δ^s. Since $\varepsilon \leq \frac{1}{24}$ we have $1 - 3\varepsilon > 20\varepsilon$ and hence a sufficient large area. By Lemma 13 we have at the right side and on top of the area a free space of at most ε which we lose. Since each level covers the area of the following level the space is sufficient. Otherwise the packed area is at least

$$(1 - 4\varepsilon) \cdot 18\varepsilon > 18\varepsilon - 16\varepsilon^2 > 2\varepsilon > Opt(L)$$

which is a contradiction.
These areas will be packed into the middle of the bin, by packing the first area at the left side, the second area right orientated at the bottom and the third area above the second area (see Figure 9). □

Now we have selected and packed all rectangles in each part. The only thing left is to sum up the area loss.

of Theorem 4. By Theorem 5 we have an area loss of at most $7/8 \cdot \varepsilon \cdot Opt(L)$. The selected long and wide rectangles have an area of at least $A((L_L)^* \cup (L_W)^*) - 1/8 \cdot \varepsilon \cdot Opt(L)$. We pack all rectangles of L_S by Lemma 19 and lose no area in this step. Altogether we obtain an area of at least

$$A((L_S)^* \cup (L_L)^* \cup (L_W)^*) - \varepsilon \cdot Opt(L) = (1 - \varepsilon)Opt(L).$$

□

We finish this section with a description of the entire algorithm.

1. Partition the instance of L into wide, long and small rectangles.

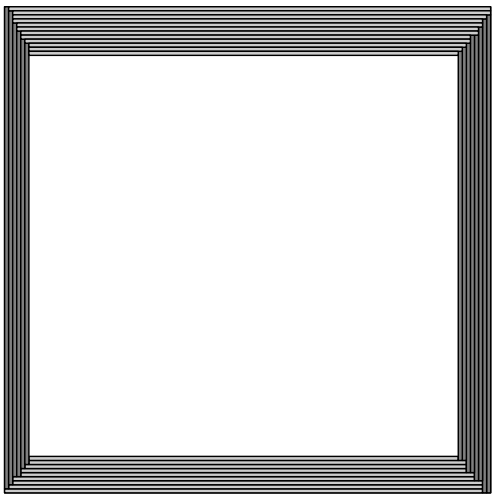

Figure 8: Long and wide rectangles at the outsides

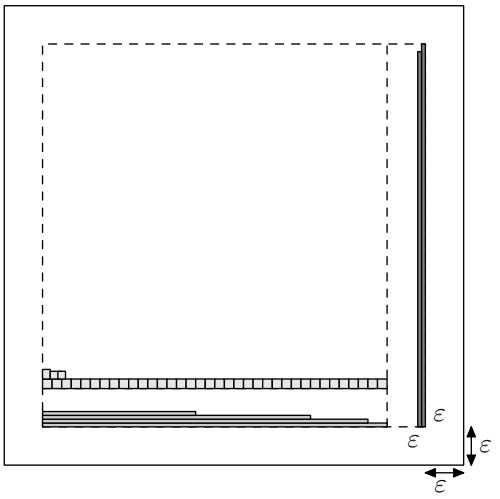

Figure 9: All rectangles have width and height less than $1 - 3\varepsilon$

2. Guess the wide and long rectangles in the set $(L_B)^*$.
3. Guess the heights and the widths of the $2m$ wide and long merged rectangles.
4. Guess the packing with the rectangles of $(L_B)^*$ and the merged rectangles.
5. Pack into the merged rectangles wide and long rectangles greedily.
6. Pack the small rectangles as described in Lemma 19.

Since all guessing steps are polynomial in n our algorithm runs in polynomial time in the length of the instance.