

# Weighted Rectangle and Cuboid Packing

Rolf Harren

Original September 4, 2005  
Modified September 19, 2006

## Abstract

Given a set of rectangular items, all of them associated with a profit, and a single bigger rectangular bin, we can ask to find a non-rotational, non-overlapping packing of a selection of these items into the bin to maximize the profit. A detailed description of the  $(2 + \epsilon)$ -approximation algorithm of Jansen and Zhang [8] for the two-dimensional case is given. Furthermore we derive a  $(16 + \epsilon)$ -approximation algorithm for the three-dimensional case (which we call cuboid packing) and improve this algorithm in a second step to an approximation ratio of  $(9 + \epsilon)$ . Finally we prove that cuboid packing does not admit an asymptotic PTAS.

It turned out, that there is a mistake in Lemma 4.2 that affects the correctness of the  $(9 + \epsilon)$  algorithm. Instead of correcting the mistake here, we refer to [4]. Based on the ideas developed in this work, we derived a  $(9 + \epsilon)$  and a  $(8 + \epsilon)$  algorithm in [4].

## 1 Introduction

Different packing problems have been in the focus of study for a long time. This paper concentrates on weighted two- and three-dimensional packing problems with the objective to pack a selection of the rectangular items without rotations into a single bin. In contrast to this, bin packing problems require to pack all items into a minimal number of bins. Strip packing is another well-studied packing problem where items have to be packed into a strip of bounded basis and unlimited height. The aim is to minimize the height that is needed to pack all items. For all these problems different variants are examined. Rotation can be allowed or forbidden. In three-dimensional packing the z-orientated packing allows rotations only along the z-axis. Furthermore, for rectangle or cuboid packing (which we use to refer to packing into one single bin) we can maximize the number of items or maximize the covered space. These are special cases of the weighted packing where the profits are equal to 1 and equal to the space, respectively. Although these problems have some similarities, they differ considerably and the results cannot be adopted directly.

To show the diversity of these different problems we present some of the known results. For strip-packing Jansen and Stee [7] and Kenyon and Rémila [10] gave an asymptotic FPTAS for the rotational and the non-rotational case, respectively. The best-known result with absolute approximation ratio is given by Steinberg [15] with an approximation ratio of 2. Bansal and Sviridenko [1]

proved that two-dimensional bin packing does not admit an asymptotic PTAS. The best-known positive result is an asymptotic approximation ratio of 1,69... given by Caprara [2]. For the special case of packing squares into a minimum number of bins an asymptotic PTAS is given by Bansal and Sviridenko [1]. Finally, Jansen and Stee presented a  $(2 + \epsilon)$ -approximation for rotational bin packing. Packing rectangles into a single bin has been slightly neglected in the past. Therefore only a limited number of results are presently available. The best result is the  $(2 + \epsilon)$ -algorithm given by Jansen and Zhang in [8] that is subject of this paper. For weighted packing of a large number of items into a bin (which means, that the size of the items is much smaller than the size of the bin) Fishkin, Gerber and Jansen [5] gave a PTAS. The special case of maximizing the number of packed squares into a rectangular bin admits an asymptotic FPTAS and therefore a PTAS (Jansen and Zhang [9]).

There are much less results for cuboid packing and all corresponding papers focus on strip- or bin packing rather than on packing into one single bin or give heuristics without an analysis of their approximation ratio. Miyazawa and Wakabayashi gave asymptotic 2,67-approximation algorithms for strip packing [16] and z-oriented packing [17].

Formally the rectangle packing problem is stated as follows. We are given a list  $I$  of  $n$  rectangles  $r_1, \dots, r_n$  where each rectangle has a width  $a_i < a$ , a height  $b_i < b$  and a profit  $p_i$ . The objective is to find a non-overlapping packing of a selection of these rectangles into a bigger rectangle  $R = (a, b)$  such that the total profit of the packed rectangles is maximized. The cuboid packing problem is defined similarly to maximize the profit of a packing of a selection of items from a given list  $I$  with cuboid items  $r_1, \dots, r_n$  of sizes  $(a_i, b_i, c_i)$  and profit  $p_i$  into a cuboid bin of size  $(a, b, c)$  (we continue to use the notation  $r_i$  even for cuboid items in order to avoid confusion with the notation of the third dimension).

Obviously two- and three-dimensional packing finds many applications. Especially three-dimensional packing is very important for the transportation and distribution of goods. Rectangle packing also finds an application in the advertisement placement which is to maximize the profit for rectangular placards that can be stuck to a limited billboard. Not so obviously we can find analogies to non-preemptive scheduling problems. For rectangle packing the analogous scheduling problem is to maximize the profit of accepted parallel jobs that have to be fulfilled before a certain deadline and may require several consecutive processors. Cuboid packing (especially in the z-orientated setting) is related to job scheduling in partitionable mesh connected systems.

The paper is organized as follows. In the preliminaries in Section 2 we make some preparations as well as introducing an important result from 2-dimensional strip packing, which will be employed by our algorithms. In Section 3 we give a detailed description of the  $(2 + \epsilon)$ -approximation algorithm for rectangle packing that is presented by Jansen and Zhang in [8]. In Section 4 we develop a  $(16 + \epsilon)$ -approximation algorithm for cuboid packing, based on the ideas of a  $(3 + \epsilon)$ -algorithm for rectangle packing in [8]. Later we improve this algorithm to a  $(9 + \epsilon)$ -algorithm. In Section 5 we show that cuboid packing does not admit an asymptotic polynomial time approximation scheme if  $NP \neq P$ . To our knowledge, the results presented in Sections 4 and 5 are currently the best known results - in fact, there has not been any paper on this issue so far. Finally, we give some remarks on future work in Section 6.

## 2 Preliminaries

At first we introduce some notations for rectangle packing and describe the measurement of approximation algorithms. For our convenience, we will refer to both rectangles and cuboids as *items* and to the bigger rectangle and cuboid as *bin*. The dimension of the items will be obvious from the content. We distinguish rectangles as *wide*, *high*, *big* and *small*. An item  $r_i = (a_i, b_i)$  is called wide if  $a_i > a/2$  and  $b_i \leq b/2$ , high if  $a_i \leq a/2$  and  $b_i > b/2$ , big if  $a_i > a/2$  and  $b_i > b/2$  and finally small if  $a_i \leq a/2$  and  $b_i \leq b/2$ . See Figure 1 for an illustration.

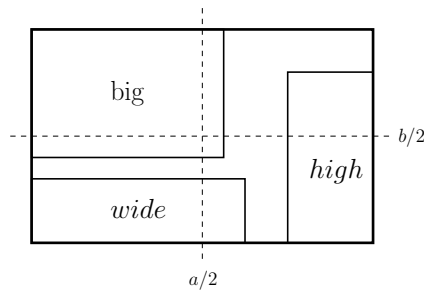


Figure 1: Wide, high and big items

In 1990, Leung et al. [13] proved the NP-completeness in the strong sense for the problem of determining whether a set of squares can be packed into a bigger square. As this is a very special case of weighted rectangle packing (and also cuboid packing) we directly obtain the NP-hardness in the strong sense for our problems. Therefore we concentrate on approximation algorithms. “To evaluate approximation algorithms, we use the standard measure *absolute performance ratio*, which is defined for maximization problems as  $R_A = \sup_I OPT(I)/A(I)$  where  $OPT(I)$  and  $A(I)$  are the optimal value and the objective value given by an approximation algorithm  $A$  for any instance  $I$ , respectively. An algorithm  $A$  is called a  $\delta$ -approximation algorithm if  $OPT(I)/A(I) \leq \delta$  for any instance  $I$ .”

Although our task is to find a concrete packing of a selection of items, we will, especially in the two-dimensional case, not really describe the packing, but rather show that a certain selection of items can be packed. We can show this by using a result from 2-dim strip packing, known as the Steinberg Theorem [15]. We give a variant of this theorem, which is proved in [8].

**Theorem 2.1** (Variant of Steinberg’s Theorem). *If the total area of a set  $T$  of items is at most  $\frac{ab}{2}$  and there are no high items (or wide items), then the items in  $T$  can be packed into a bin of size  $(a, b)$ .*

Notice that this variant includes the case that there are wide items and one big item in  $T$ , which is not included in the original theorem.

## 3 A $(2 + \epsilon)$ - approximation algorithm for rectangle packing with profits

### 3.1 Introduction

At first we present some basic methods used to design algorithms for theoretical purposes. These methods focus on polynomial running time which may unfortunately be of very high order. We present the *enumeration*, the *constant packing* and the *linear programming* method. Especially enumeration and constant packing are not really of practical use.

**Enumeration (guessing):** Assume a polynomial number of cases, each of which can be solved in polynomial time. Obviously we can compute all cases and therefore find an optimal solution in polynomial time. As we might want to enumerate the cases to do this, the method is called *enumeration*. Alternatively, we also use the *guessing* perspective. Hereby we guess an optimal solution from a polynomial number of cases. Actually this includes the exhaustive search of the enumeration. The advantage of this perspective is the possibility to use an optimal solution in the description of our algorithm.

**Constant packing:** “For a constant number of items (bounded by  $N$ ) the test of the existence of a feasible packing takes constant time (exponentially in  $N$ ). To see this, consider the following simple packing algorithm *Bottom-Up (BU)* for a list of items: When an item is packed, it is placed, left-justified at the lowest possible level in the current packing. It is not difficult to observe that any optimal packing can be transferred into a packing generated by *BU* on a specified order of items. For  $N$  items, we can apply the algorithm *BU* to  $N!$  different item lists (different orders). A feasible packing exists if and only if *BU* gives one.”

**Linear programming:** As it is NP-hard to solve integer programs, we use linear programming as a relaxation to find a selection of items with almost optimal profit. Fractional solutions may occur and will be handled appropriately. In 1979 Khachiyan introduced the ellipsoid method as the first polynomial time algorithm for linear programming [11]. Later Karmarkar introduced the more efficient inner-point method [12].

We describe the  $(2 + \epsilon)$ -algorithm in several steps starting with a very basic algorithm which will reveal the main ideas. Afterwards we will subsequently add more details.

### 3.2 First step - guessing the most profitable items, basic algorithm

Let  $k$  be a constant. We suppose the existence of an optimal packing with more than  $k$  items. The other case will be handled later.

Firstly, we guess the  $k$  most profitable items of an optimal packing, calling them the *candidate set*  $S$ . As  $k$  is a constant, the number  $\binom{n}{k}$  of possible candidate sets is polynomial and the guessing can be done in polynomial time. We ensure that the items of  $S$  can be packed into a bin by using the constant packing method. All additionally packed items have profits bounded by  $\min_{r \in S} p_r$ .

Later we will have more sophisticated estimations for this profit. Notice that it is crucial that we guess the most profitable items of an optimal packing and not just an arbitrary selection of packed items. Secondly, we construct a linear program to find a set  $X$  of additional items with small profits such that the overall profit is almost optimal. Thirdly, we pack  $X \cup S$  into two bins of size  $(a, b)$  by using our variant of Steinberg's Theorem 2.1.

Our basic algorithm can be outlined as follows

1. guess the  $k$  most profitable items  $S$  of an optimal solution,
2. find a set  $X$  of additional items with almost optimal profit by solving a linear program,
3. pack  $X \cup S$  into two bins of size  $(a, b)$  and choose the bin with the higher profit.

We will become more specific soon. It should be plausible now, that we get a  $(2 + \epsilon)$ -approximation because the profit of  $X \cup S$  is almost optimal (which means  $p(X \cup S) \geq (1 - \epsilon)OPT(I)$ ) and therefore the higher profit is at least near to the half of the optimum.

We might now ask why the first step is mandatory, if we can find the set  $X$  with a linear program. This is due to the discarding of some of the items of  $X$  in later steps. Therefore we need to estimate the total loss of discarded items.

### 3.3 Second step - guessing the gap structure

Our aim is to construct constraints for the linear program that ensure, that we can pack all the selected items into two bins. We derive constraints for the wide and for the high items separately but we do not consider the small items at the moment. For the sake of simplicity we consider the big items as high items. Exemplarily we describe the constraints for the wide items.

Although we obviously do not know an optimal packing it makes sense to examine the structure of an optimal packing to derive properties we can use for the construction of constraints. Suppose we know a packing  $P$  of an optimal solution  $L_{opt}$  with the candidate set  $S$  - see Figure 2. We examine the gap structure in the bin that is created by the items of the candidate set.

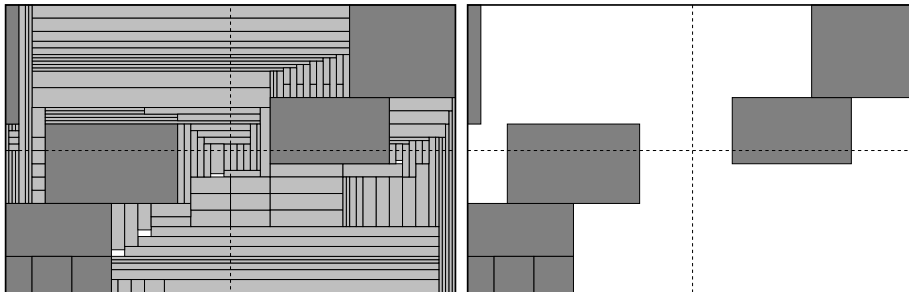


Figure 2: Packing  $P$  of an optimal solution  $L_{opt}$  with the candidate set  $S$

“To find the gaps in our packing  $P$ , use the following sweeping procedure. Remove all items that do not belong to the candidate set and start to sweep with a horizontal line beginning with the lower boundary of  $(a, b)$ . By sweeping find the lower and upper boundary of each candidate rectangle  $r$  in  $S$ . Insert a horizontal line to the left and right until it intersects another rectangle of  $S$  or ends at the boundary of rectangle  $(a, b)$ . At most  $2|S| = 2k$  lines are inserted. This partitions the big rectangle  $(a, b)$  into rectangles of  $S$  and a set of empty rectangles. Empty rectangles of width  $> a/2$  are called wide gaps. Clearly, there is at most one gap of width  $> a/2$  intersecting with a horizontal line. Therefore the number of wide gaps is bounded by  $2|S| + 1 = 2k + 1$  - see Figure 3 for an illustration.”

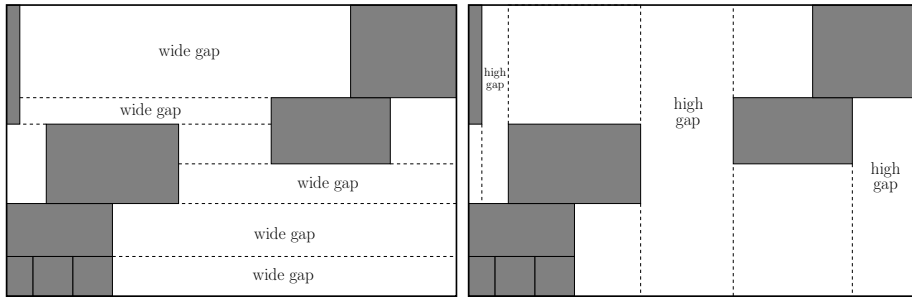


Figure 3: Gap structures of the wide and high gaps

It is clear, that additional wide items can only be contained in wide gaps - but they might overlap two or more wide gaps partially. We want to use the wide gaps, which we are going to guess later, to make up constraints for the linear program. Therefore we have to ensure on the one hand, that every wide item is interior to exactly one wide gap and on the other hand, that there is only a polynomial number of cases to choose for the gaps, such that they can be guessed.

As there are at most  $2k + 1$  wide gaps, there can be at most  $2k$  additional wide items in the optimal solution that overlap two or more gaps. We remove these items from the optimal solution  $L_{opt}$  - which makes  $L_{opt}$  suboptimal. Later we show, that  $L_{opt}$  is still almost optimal which is sufficient for us.

Guessing the gaps is a bit more complicated. We reduce the gap sizes to certain threshold values. Consider one wide gap together with the contained wide items (we do not consider small items that may be also contained in the gap). Suppose the wide items are piled up left-aligned in a non-increasing order. Let  $h$  be the height of the gap and  $h_{max}$  the height of the highest wide items in the gap. Select the greatest  $n \in \{1, \dots, |I \setminus S|\}$  such that  $n \cdot h_{max} \leq h$  and set the new height of the gap to  $n \cdot h_{max}$  (it is possible to bound  $n$  because the maximal height of  $|I \setminus S| \cdot h_{max}$  is obviously sufficient to pack all but the candidate set into the gap). If the height decreases under the total height of the contained wide items we remove the highest item. Next we reduce the width of the gap to the width of the widest item that is still in the gap - see Figures 4 and 5. In total we remove at most  $2k + 1$  items - which is one item per gap.

The overall loss of profit of the removed (at most)  $8k + 2$  items (for wide and high items each at most  $2k + 1$  for the resizing of the gaps and at most  $2k$  for the overlapping items) can be estimated by  $(\epsilon/2)OPT(I)$  if we choose  $k$

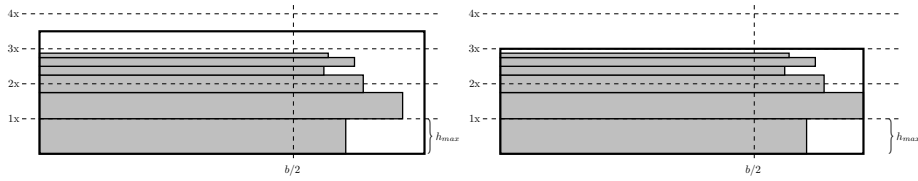


Figure 4: The size of the gap is reduced to three times  $h_{max}$  and  $w_{max}$ , all items can still be packed

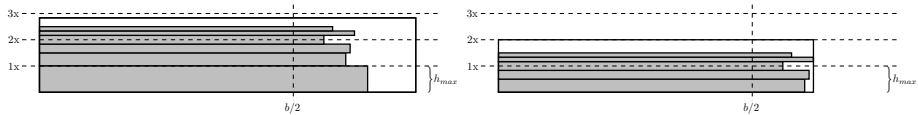


Figure 5: The size of the gap is reduced to double  $h_{max}$ . As not all items fit into the resized gap, the highest item is removed. Finally, the new width is set to  $w_{max}$ .

appropriately. We show this in the final step of the description.

The sizes of the gaps are now defined by three parameters: 1) the height of one item (not necessarily still included in the gap), 2) the multiple of the height and 3) the width of the widest item. These three parameters can take at most  $n$  different values each. Thus there are at most  $n^3$  possibilities to parametrize each gap and we can guess the complete gap structure.

Now we redesign the first step of our algorithm, such that we guess up to  $2k + 1$  wide and  $2k + 1$  high gaps additionally to the  $k$  most profitable items. Guessing the gaps is done by fixing the three parameters. At the same time we ensure that the wide gaps can be packed together with the candidate set as well as that the high gaps can be packed together with the candidate set. This can be done with the constant packing method we described earlier. Altogether we get

1. guess the  $k$  most profitable items  $S$  and the gap structure of an optimal solution,
2. find a set  $X$  of additional items with almost optimal profit by solving a linear program,
3. pack  $X \cup S$  into two bins of size  $(a, b)$  and choose the bin with the higher profit.

It might be surprising that we consider the wide and the high items separately and only ensure that the wide and the high gaps can be packed separately together with the candidate set. This is due to the aim of constructing a  $(2 + \epsilon)$ -approximation. We will pack the high and the wide items separately with the candidate set. Our examination of the gap structure ensures, that the selection of the linear program can be packed and has sufficiently high profit. Although we did not consider the small items so far, it will be possible to add them in the following steps.

### 3.4 Third step - integer program

Where did our considerations lead us so far? For every optimal solution there is a restricted solution with almost the same profit ( $OPT_{res}(I) \geq (1 - \epsilon/2)OPT(I)$ ) and the property, that all wide items are interior to the wide gaps and can additionally be packed into insignificantly downsized gaps. We can guess these downsized gaps and use them in the constraints of the linear program. Naturally the same applies to the high items and high gaps.

Suppose we guessed a candidate set  $S$  and additionally  $W \leq 2k + 1$  wide gaps  $(a_i^w, b_i^w)$  and  $H \leq 2k + 1$  high gaps  $(a_i^h, b_i^h)$ . Separate the remaining items into the sets  $R^w \subset I \setminus S$  of wide items,  $R^h \subset I \setminus S$  of high items and  $R^s \subset I \setminus S$  of small items. Each item in  $R^w$ ,  $R^h$  and  $R^s$  has profit at most  $\min_{r \in S} p_r$ . For each wide item  $r$  we define a set  $G_r^w$  of potential gaps  $g$  with  $a_r \leq a_g^w$ . Similarly, we define  $G_r^h$  for high items  $r$  with gaps  $g$  such that  $b_r \leq b_g^h$ .

For each wide (high) item  $r$  and each wide (high) gap  $g \in G_r^w$  ( $g \in G_r^h$ ) we introduce a variable  $x_{r,g} \in \{0, 1\}$  where

$$x_{r,g} = \begin{cases} 1, & \text{if } r \text{ is placed in gap } g \\ 0, & \text{otherwise.} \end{cases}$$

For each small item  $r$  we use a variable  $x_r \in \{0, 1\}$  where

$$x_r = \begin{cases} 1, & \text{if } r \text{ is selected} \\ 0, & \text{otherwise.} \end{cases}$$

Our objective is to maximize the profit, which is the sum of the profits from the candidate set  $S$  and the selected items from the sets  $R^w$ ,  $R^h$  and  $R^s$ . Formally

$$\max \underbrace{\sum_{r \in S} p_r}_{\text{candidate set } S} + \underbrace{\sum_{r \in R^w} p_r \sum_{g \in G_r^w} x_{r,g}}_{\text{wide items } R^w} + \underbrace{\sum_{r \in R^h} p_r \sum_{g \in G_r^h} x_{r,g}}_{\text{high items } R^h} + \underbrace{\sum_{r \in R^s} p_r \cdot x_r}_{\text{small items } R^s}$$

Due to the guessing of the gap structure we get two constraints for both wide and high items. First, every wide item can be packed at most in one wide gap and second, the total height of wide items in a wide gap is bounded by the height of that gap. For the wide items and wide gaps we get the following constraints

$$\begin{aligned} \sum_{g \in G_r^w} x_{r,g} &\leq 1 & r \in R^w \\ \sum_{r \in R^w, a_r \leq a_g^w} b_r x_{r,g} &\leq b_g^w & 1 \leq g \leq W \end{aligned}$$

and similar for high gaps and high items

$$\begin{aligned} \sum_{g \in G_r^h} x_{r,g} &\leq 1 & r \in R^h \\ \sum_{r \in R^h, b_r \leq b_g^h} a_r x_{r,g} &\leq a_g^h & 1 \leq g \leq H \end{aligned}$$

Finally we add a constraint on the space to bound the selection of small items. The total area of the selected items may not exceed the bin size.

$$\underbrace{\sum_{r \in S} a_r b_r}_{\text{candidate set } S} + \underbrace{\sum_{r \in R^w} a_r b_r \sum_{g \in G_r^w} x_{r,g}}_{\text{wide items } R^w} + \underbrace{\sum_{r \in R^h} a_r b_r \sum_{g \in G_r^h} x_{r,g}}_{\text{high items } R^h} + \underbrace{\sum_{r \in R^s} a_r b_r \cdot x_r}_{\text{small items } R^s} \leq ab$$

In fact the area of the candidate set is constant, as we guessed the set beforehand. Thus this constant can be computed with the right hand side of the inequality.

Adding the integer constraint we get the following integer program

$$\max \quad \sum_{r \in S} p_r + \sum_{r \in R^w} p_r \sum_{g \in G_r^w} x_{r,g} + \sum_{r \in R^h} p_r \sum_{g \in G_r^h} x_{r,g} + \sum_{r \in R^s} p_r \cdot x_r$$

such that

$$\begin{aligned} \sum_{g \in G_r^w} x_{r,g} &\leq 1 & r \in R^w \\ \sum_{g \in G_r^h} x_{r,g} &\leq 1 & r \in R^h \\ \sum_{r \in R^w, a_r \leq a_g^w} b_r x_{r,g} &\leq b_g^w & 1 \leq g \leq W \\ \sum_{r \in R^h, b_r \leq b_g^h} a_r x_{r,g} &\leq a_g^h & 1 \leq g \leq H \\ \sum_{r \in S} a_r b_r + \sum_{r \in R^w} a_r b_r \sum_{g \in G_r^w} x_{r,g} + \sum_{r \in R^h} a_r b_r \sum_{g \in G_r^h} x_{r,g} + \sum_{r \in R^s} a_r b_r \cdot x_r &\leq ab \\ x_{r,g} &\in \{0, 1\} & r \in R^w, g \in G_r^w \\ x_{r,g} &\in \{0, 1\} & r \in R^h, g \in G_r^h \\ x_r &\in \{0, 1\} & r \in R^s \end{aligned}$$

Relaxing the integer constraints to  $x_{r,g} \in [0, 1]$  and  $x_r \in [0, 1]$  we derive a linear program. Of course we have to adapt the interpretation of the variables to

$$\begin{aligned} x_{r,g} &= \text{fraction of } r \text{ that is placed in gap } g \\ x_r &= \text{fraction of } r \text{ that is selected.} \end{aligned}$$

Recall that an optimal solution  $(x_{r,g}^*, x_r^*)$  of the linear program can be found in polynomial time and notice that the profit  $P^*$  of this solution is at least as big as the profit of the restricted solution  $OPT_{res}(I)$  and therefore  $P^* \geq (1 - \epsilon/2)OPT(I)$ . Now we have to transform this, possibly fractional, solution into an integer solution without losing too much profit. This is done by a reduction to job scheduling on unrelated machines with costs which we present for the wide items:

“Suppose we have an optimal LP solution  $(x_{r,g}^*, x_r^*)$  of profit  $P^* \geq (1-\epsilon/2)OPT(I)$ . Consider the set  $R^w$  of wide items and compute the fractional filling height  $b_g^* = \sum_{r \in R^w, a_r \leq a_g^w} b_r x_{r,g}^*$  for all gaps  $1 \leq g \leq W$  and the fractional area  $A_w^* = \sum_{r \in R^w} a_r b_r \sum_{g \in G_r^w} x_{r,g}^*$ . The fractional profit of the wide items is  $P_w^* = \sum_{r \in R^w} p_r \sum_{g \in G_r^w} x_{r,g}^*$ . Obviously the vector  $(x_{r,g}^*)_{r \in R^w}$  satisfies the following system of inequalities

$$\begin{aligned} \sum_{r \in R^w} p_r \sum_{g \in G_r^w} x_{r,g} &\geq P_w^* \\ \sum_{r \in R^w} a_r b_r \sum_{g \in G_r^w} x_{r,g} &\leq A_w^* \\ \sum_{r \in R^w, a_r \leq a_g^w} b_r x_{r,g} &\leq b_g^* && \text{for all gaps } 1 \leq g \leq W \\ \sum_{g \in G_r^w} x_{r,g} &\leq 1 && \text{for all } r \in R^w \\ x_{r,g} &\in [0, 1] && \text{for all } r \in R^w \text{ and } g \in G_r^w \end{aligned}$$

The next step is to round the solution such that only few variables have a fractional assignment. To do this, we use a new variable  $x_{r,0} = 1 - \sum_{g \in G_r^w} x_{r,g}$  and replace the first inequality above by

$$\sum_{r \in R^w} p_r x_{r,0} \leq \sum_{r \in R^w} p_r - P_w^*$$

If  $\sum_{r \in R^w} p_r = P_w^*$ , then all wide items are selected and  $\sum_{i \in G_r^w} x_{r,i}^* = 1$  for each  $r \in R^w$ . In this case we remove the first inequality and consider the remaining system of inequalities. If  $A_w^* = 0$  then all variables  $x_{r,g} = 0$  and a rounding is not necessary. In this case no wide item is selected. If  $b_i^* = 0$  then all variables  $x_{r,g}$  corresponding to gap  $g$  and item  $r \in R^w$  with  $a_r \leq a_g^w$  in the summation above must be zero and can be removed. In addition we can remove the inequality corresponding to gap  $g$ .

Suppose that  $A_w^* > 0$ ,  $\sum_{r \in R^w} p_r - P_w^* > 0$  and  $b_i^* > 0$  for all gaps  $1 \leq i \leq W$ . Next we divide the inequalities by the right hand side (this is possible since the right hand sides are positive). This gives a system of inequalities of the following form:

$$\begin{aligned} \sum_{r \in R^w} d_r x_{r,0} &\leq 1 \\ \sum_{r \in R^w} e_r \sum_{g \in G_r^w} x_{r,g} &\leq 1 \\ \sum_{r \in R^w, a_r \leq a_i^w} f_r x_{r,g} &\leq 1 && \text{for all gaps } 1 \leq g \leq W \\ x_{r,0} + \sum_{g \in G_r^w} x_{r,g} &= 1 && \text{for all } r \in R^w \\ x_{r,g} &\geq 0 && \text{for all } r \in R^w \text{ and } g \in G_r^w \end{aligned}$$

where  $d_r = p_r / (\sum_{r \in R^w} p_r - P_w^*)$ ,  $e_r = a_r b_r / A_w^*$  and  $f_r = b_r / b_i^*$ . This system is related to a scheduling problem of jobs on  $W+1$  unrelated machines with costs.

The first inequality is related to the cost of the solution and minimizing the cost gives a feasible solution. The next  $W + 1$  inequalities are related to the machines  $1, \dots, W + 1$ . One can round a fractional solution of this system in polynomial time to another solution with optimum cost and at most  $W$  fractional jobs [14], [6]. Removing these fractional items after the rounding and using the same idea for the high and small items gives an integral solution where we have removed at most  $W + H + 1 \leq 4k + 3$  items. Notice, that we have no inequalities for gaps for the small items. Therefore we remove at most 1 small item.”

As in the last section we removed a number of items in this step. The total profit of these items can also be bound by  $(\epsilon/2)OPT$  as we show later. The current state of our algorithm is as follows

1. guess the  $k$  most profitable items  $S$  and the gap structure of an optimal solution,
2. find a fractional solution of the linear program with almost optimal profit,
3. round the fractional solution to a selection of items  $X$ ,
4. pack  $X \cup S$  into two bins of size  $(a, b)$  and choose the bin with the higher profit.

### 3.5 Fourth step - packing

In this chapter the packing of the items corresponding to the integer solution into two bins is described. From the result of our integer solution we know how to pack either the wide items or the high items together with the candidate set into a bin. We only need to pile up the wide (high) items into the corresponding gaps and pack the constant number of candidate items and gaps into the bin (we verified that this is possible in the second step).

Let  $X$  be the set of additional items from the integer solution. We can split  $X$  into three set  $X^w$ ,  $X^h$  and  $X^s$  of wide, high and small items respectively (recall that a probably included big item is considered as high one). We distribute these items to the sets  $T^w$  and  $T^h$  which represent the two bins in which we want to pack. Let  $\bar{P}$  be the total profit of the items  $S \cup X$ .

“First, we submit  $S$  to both sets  $T^w$  and  $T^h$ . Second, we add  $X^w$  to  $T^w$  and  $X^h$  to  $T^h$ . If either  $p(T^w) \geq \bar{P}/2$  or  $p(T^h) \geq \bar{P}/2$  we pack the set with the higher profit into a bin. This is possible due to the remark above. Otherwise  $p(T^w) < \bar{P}/2$  and  $p(T^h) < \bar{P}/2$ . In this case we have to add the small items in  $X^s$  to our sets. Therefore we can not use the gap structure to pack the bins any longer. Instead we use the variant of Steinberg’s Theorem (Theorem 2.1). This is possible because we separated the wide and high items into different sets.

If the area  $s(T^w) > ab/2$  then the area of the other items is  $s(T^h \cup X^s) \leq ab/2$ . Therefore we can use our variant of Steinberg’s Theorem to pack the items of  $X^s$  and  $T^h$  together into one bin. The profit of these items is  $p(T^h \cup X^s) > \bar{P}/2$ . The other case ( $s(T^h) > ab/2$ ) is packed similarly.

Suppose now that  $s(T^w) \leq ab/2$  and  $s(T^h) \leq ab/2$ . We distribute the small items of  $X^s$  into three sets  $X_{T^w}^s$ ,  $X_{T^h}^s$  and  $X^*$  such that  $s(T^w \cup X_{T^w}^s) \leq ab/2$ ,  $s(T^h \cup X_{T^h}^s) \leq ab/2$  and  $|X^*| = 1$ . This can be done easily in a greedy manner.

Let  $X^* = \{\bar{r}\}$ . From the sets  $T^w \cup X_{T^w}^s$  and  $T^h \cup X_{T^h}^s$  we choose the one with the higher profit and pack it, using the variant of Steinberg's Theorem, into a bin. In the last step we will show that the profit is  $\geq \bar{P}/2 - p_{\bar{r}} \geq (1 - \epsilon)OPT(I)/2$ ."

### 3.6 Final steps

In this chapter we present the missing bits of the algorithms which are the choice of the constant  $k$  and handling the cases with an optimal number of items smaller than  $k$ .

**Choice of the constant  $k$ :** The constant  $k$  determines the number of items we guess as a candidate set  $S$ . It is crucial that the constant is chosen carefully because the cardinality of  $S$  directly affects the upper bound for the number of gaps and therefore the number of removed items. Our aim is to choose  $k$ , such that the total profit of all removed items is at most  $\epsilon OPT(I)$ . In total we remove up to  $8k + 2$  items in step two (creating the restricted optimal solution), up to  $4k + 3$  (which is  $\leq 8k + 1$  for  $k \geq 1$ ) in step three (rounding the fractional solution) and at most 1 item in step four (packing). We will show, that the profit of up to  $8k + 2$  items that do not belong to  $S$  is bounded by  $(\epsilon/2)OPT(I)$  for optimal solutions where the number of items is high enough.

At first we scale the input instance such that  $OPT(I) \in [1/4, 1]$ . Assume an optimal solution  $L_{opt} \subset I$  of profit  $OPT(I)$ . Jansen and Zhang present an  $(3 + \epsilon)$ -approximation algorithm for rectangle packing in [8]. We use this algorithm to find a packable set of items  $L_H$  with profit  $OPT(I) \geq p(L_H) \geq (1/4)OPT(I)$ . By dividing the profit of every item by  $4/p(L_H)$  we can assume  $OPT(I) \geq 1/4 \geq (1/4)OPT(I)$  and therefore  $OPT(I) \in [1/4, 1]$ .

Second we use the following lemma to proof the existence of a fitting  $k$  for optimal solutions with sufficiently high number of items. Suppose w.l.o.g. that  $L_{opt} = \{r_1, r_2, \dots, r_{\bar{n}}\}$  where  $p_1 \geq p_2 \geq \dots \geq p_{\bar{n}}$ . In our case  $\sum_{i=1}^{\bar{n}} p_i = OPT(I) \in [1/4, 1]$ . In [6] Jansen and Porkolab prove

**Lemma 3.1.** *Let  $L = (p_1, \dots, p_n)$  be a sequence of integers such that  $\sum_{i=1}^n p_i \leq 1$  and let  $c, d$  be constant integers and  $\alpha > 0$ . If  $n$  is large enough (i.e.  $n > (\lceil 1/\alpha \rceil d + 1)(c + 1)^{\lceil 1/\alpha \rceil}$ ) then there is a constant  $k \geq 1$  such that*

$$p_{k+1} + \dots + p_{k+ck+d} \leq \alpha$$

and

$$k \leq \lceil (c + d)(c + 1)^{\lceil 1/\alpha \rceil - 1} - d \rceil / c.$$

We set  $n = \bar{n}$ ,  $c = 8$ ,  $d = 2$  and  $\alpha = \epsilon/8$  and derive, that if  $n > (\lceil 8/\epsilon \rceil 2 + 1)9^{\lceil 8/\epsilon \rceil}$  (which is constant for a given  $\epsilon > 0$ ) we have  $p_{k+1} + \dots + p_{k+8k+2} \leq (\epsilon/8) \leq (\epsilon/2)OPT(I)$  and  $k \leq 9^{\lceil 8/\epsilon \rceil}$ . This implies that for each optimal solution with more than  $(\lceil 8/\epsilon \rceil 2 + 1)9^{\lceil 8/\epsilon \rceil}$  items there is always a constant  $k$  such that the sum of  $8k + 2$  smaller profit values is at most  $(\epsilon/2)OPT(I)$ .

**Final adaptations of the algorithm:** As we do not know beforehand whether the optimal solution consists of a large number of items or not, we have to consider the case that the number of items is small (which means  $\leq (\lceil 8/\epsilon \rceil 2 + 1)9^{\lceil 8/\epsilon \rceil}$ ). To cover this, we enumerate all subsets of items with a cardinality  $\leq (\lceil 8/\epsilon \rceil 2 + 1)9^{\lceil 8/\epsilon \rceil}$  and test whether they can be packed together into one bin.

Obviously we get an optimal solution in this case. Additionally we guess the constant  $k$  and the  $9k + 2$  items with largest profit. If  $p_{k+1} + \dots + p_{8k+2} > \epsilon/8$  then we discard the choice. Otherwise we apply the procedure above. Lemma 3.1 shows, that such a constant  $k$  and selection of  $9k + 2$  items exist for an optimal solution with enough items.

“For this  $k$  there is a candidate set and a sequence of discrete gaps (with the structure described in step two) such that the LP gives a profit of at least  $(1 - \epsilon/2)OPT(I)$ . By the rounding procedure and the transformation into a feasible packing we get a solution with profit at least  $(1 - \epsilon)OPT(I)/2 \geq OPT(I)/(2 + 3\epsilon)$  for  $\epsilon \leq 1/3$ . By dividing  $\epsilon$  by 3, this gives the following result:

**Theorem 3.1** (Rectangle packing). *There is a polynomial time approximation algorithm with performance ratio of at most  $2 + \epsilon$  for rectangle packing, for any  $\epsilon > 0$ . ”*

**Final Remark:** We do not give a concrete bound for the running time rather than stating that it is polynomial. This is due to the fact, that even for quite big choices of  $\epsilon$  the bound for the number of items is very big. The enumeration for all cases is indeed polynomial (in fact it is constant for a given  $\epsilon$ ) but incredibly big.

## 4 Cuboid packing

In the remainder of this paper we consider the three-dimensional packing problem. There are two natural approaches to cover. First, we derive a constant ratio approximation algorithm (which we improve afterwards) and second, we show that cuboid packing does not admit even an asymptotic PTAS.

### 4.1 A constant ratio approximation algorithm for cuboid packing

At first we show a relatively simple  $(16 + \epsilon)$ -approximation algorithm which we improve with some minor tweaks to a  $(9 + \epsilon)$ -approximation. The main idea for our algorithm is to use a PTAS for an  $m$ -dimensional knapsack problem to get a selection of items with almost optimal profit and pack this selection in up to 16 bins.

The  $m$ -dimensional knapsack problem is to find for given matrix  $A \in N^{m \cdot n}$  and vectors  $b \in N^m$  and  $p \in N^n$  a solution  $x \in N^n$  which maximizes the profit  $\sum_{i=1}^n p_i x_i$  under the side condition  $Ax \leq b$ . All numbers are nonnegative integers. The side conditions can be interpreted as several bounds for space or weight of the items like in the 1-dim knapsack problem. Chandra et al. [3] proved the existence of a PTAS for the  $m$ -dimensional knapsack problem.

For every direction  $a$ ,  $b$  and  $c$  we create a side condition which bounds the sum of the side planes of the long items. An item is called *long in one direction* if the length in this direction is more than half of the bins size. The conditions are:

$$\sum_{i \in I, a_i > a/2} b_i c_i \cdot x_i \leq bc$$

$$\sum_{i \in I, b_i > b/2} a_i c_i \cdot x_i \leq ac$$

$$\sum_{i \in I, c_i > c/2} a_i b_i \cdot x_i \leq ab$$

A fourth and fifth condition is added in order to limit the total space used by all items and the number of big items, respectively.

$$\sum_{i \in I} a_i b_i c_i \cdot x_i \leq abc$$

$$\sum_{i \in I, a_i > a/2, b_i > b/2, c_i > c/2} x_i \leq 1$$

The objective remains untouched:

$$\sum_{i \in I} p_i x_i$$

**Remark:** In fact this resembles the construction of the integer program we derived in the previous sections. But as we only use upper bounds and a constant number of constraints, we can regard this as a  $m$ -dim knapsack problem and therefore solve it directly instead of having to round the fractional solution of a linear relaxation. A similar idea is used in [8] to derive a  $(3 + \epsilon)$ -approximation for the rectangle packing.

**Lemma 4.1.** *Every packable selection  $I'$  of items can be canonically interpreted as a vector that satisfies the  $m$ -dimensional knapsack instance.*

*Proof.* We show that all the constraints are satisfied. Obviously the *total space*- and the *big item*- constraint are satisfied by every feasible packing. Now observe that two items that are long in direction  $a$  can not be packed behind each other in this direction (they would exceed the boundary of the bin) - see left part of Figure 6. Therefore the sum of the projected area in  $bc$ -direction of items that are long in direction  $a$  is bounded by  $bc$  - see right part of Figure 6. Similarly we can derive the side conditions for the other directions.  $\square$

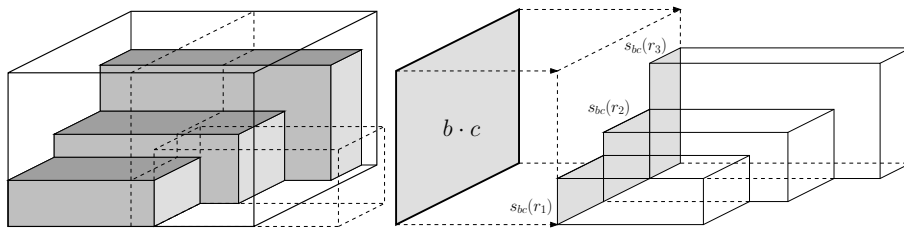


Figure 6: Observation for the side condition for long items in direction  $a$

We conclude that the optimal solution of the knapsack instance has at least the same profit as the optimal solution of the cuboid packing instance. Now we use the PTAS for  $m$ -dimensional knapsack to find a list  $I_k$  of items with almost optimal profit  $p(I_k) \geq (1 - \epsilon)OPT_{knapsack}(I) \geq (1 - \epsilon)OPT(I)$ .

Suppose we have a set  $I_k$  of items with almost optimal profit  $p(I_k)$  from the knapsack problem. Now we separate  $I_k$  into three sets  $S_a, S_b, S_c$  of long items,

a set  $S_{big}$  of at most one big item and a set  $S_{small}$  of small items such that (see Figure 7)

$$\begin{aligned}
S_a &= \{i \in I_k \mid a_i > a/2, c_i \leq c/2\} \\
S_b &= \{i \in I_k \mid b_i > b/2, a_i \leq a/2\} \\
S_c &= \{i \in I_k \mid c_i > c/2, b_i \leq b/2\} \\
S_{big} &= \{i \in I_k \mid a_i > a/2, b_i > b/2, c_i > c/2\} \\
S_{small} &= \{i \in I_k \mid a_i \leq a/2, b_i \leq b/2, c_i \leq c/2\}
\end{aligned}$$

We pack each set (except of  $S_{big}$  which we include to one of the others later)

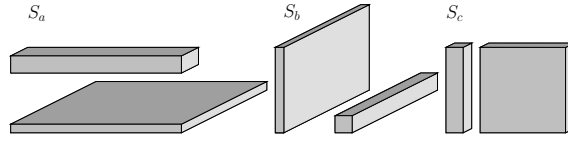


Figure 7: Illustration of shapes of items in  $S_a$ ,  $S_b$  and  $S_c$

separately into up to 3 (as for  $S_a$ ,  $S_b$  and  $S_c$ ) and up to 7 bins (as for  $S_{small}$ ). In total we get along with 16 bins and declare the bin with highest profit  $P^*$  as the solution, achieving an approximation ratio of  $(16 + \epsilon)$ .

**Packing of the long items:** Exemplarily we describe the packing of  $S_a$ . As we saw before the projections on the  $(b, c)$ -side plane of the bin do not overlap in a regular packing. We use this observation to reduce the problem to rectangle packing. The algorithm we use equals the  $(3 + \epsilon)$ -approximation algorithm in Jansen and Zhang [8]. Pack the  $bc$ -projections of the items into a rectangle of size  $(2b, c)$  using the variant of the Steinberg Theorem 2.1. This is possible since the total area of the projections is  $\leq bc$  and there are no wide items (if we regard the  $b$  direction as the horizontal direction). Draw a vertical line to divide the  $(2b, c)$  rectangle into two rectangles of size  $(b, c)$  and pack the items, that are cut by this line, into a third rectangle of the same size. To do this, retain the order of the items and pack them left aligned into the rectangle - see Figure 8 for an illustration.

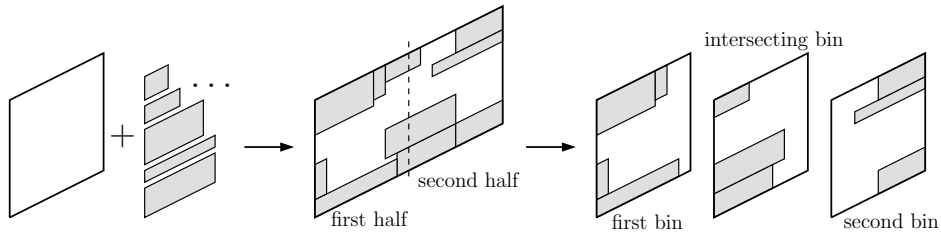


Figure 8: Packing the projections of  $S_a$  into three bins

Finally we fit these three  $(b, c)$ -rectangles into three  $(a, b, c)$ -bins and erect the associated items in  $a$  direction on their projections - see Figure 9. The same algorithm can be applied to  $S_b$  and  $S_c$ , packing each in 3 bins of size  $(a, b, c)$ .

As  $S_{big}$  contains at most one item it can be packed together with any of the sets  $S_a$ ,  $S_b$  or  $S_c$  - see the note on the variant of the Steinberg's Theorem 2.1.

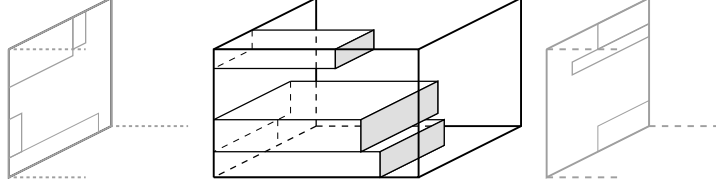


Figure 9: Erecting the items of  $S_a$  on the pattern of their projections

**Packing of the small items:** To pack the set of small items  $S_{small}$  we use a combination of rectangle packing with Steinberg and a layer packing into an unlimited 3-dim strip similar to the simple 2-dim algorithm *Next-Fit-Decreasing-Height (NFDH)* which orders the items in non-increasing height and fills the strip in layers. First, we order the items in non-increasing height  $c_1 \geq c_2 \geq \dots \geq c_k$ . Second, we group the items such that the sum of the  $ab$ -size of each group is as close to  $ab/2$  as possible. Formally, we define variables  $g_1, \dots, g_l$  such that  $g_1 = 1$  and

$$\begin{aligned}
 S_i &= \sum_{j=g_i}^{g_{i+1}-1} a_j b_j \leq ab/2 \quad \text{for all } 1 \leq i \leq l-1 \quad \text{and} \\
 &\quad \sum_{j=g_i}^{g_{i+1}} a_j b_j > ab/2 \quad \text{for all } 1 \leq i \leq l-1 \quad \text{and} \\
 S_i &= \sum_{j=g_l}^k a_j b_j \leq ab/2
 \end{aligned}$$

As all the items are small, the  $ab$ -area of each item is  $\leq ab/4$ . Therefore  $S_i \geq ab/4$  for all  $1 \leq i \leq l-1$ . Furthermore the  $ab$ -projection of each group  $G_i = (g_i, g_i + 1, \dots, g_{i+1} - 1)$  of items can be packed into a rectangle of size  $(a, b)$  with Steinberg's algorithm, as their total area is bounded by  $ab/2$  and there are neither wide nor high items. As for the large items we can erect the small items on their projection and therefore pack each group  $G_i$  into one layer of size  $(a, b, c_{g_i})$  with the highest height in that group. Insert the layers in non-increasing order into a strip with the basis  $(a, b)$  and unlimited height - see Figure 10.

We analyse the height of this strip packing. Suppose the total volume of small items is bounded by  $V = \sum_{i \in S_{small}} a_i b_i c_i \leq \alpha abc$ . In every layer  $1 \leq i \leq l-1$  all items have a  $c$ -length of at least  $c_{g_{i+1}}$ . Hence, the filling of every layer can be estimated by the height of the succeeding layer. Therefore the total volume can be estimated by

$$V \geq \sum_{i=1}^{l-1} c_{g_{i+1}} S_i \geq \sum_{i=1}^{l-1} c_{g_{i+1}} ab/4$$

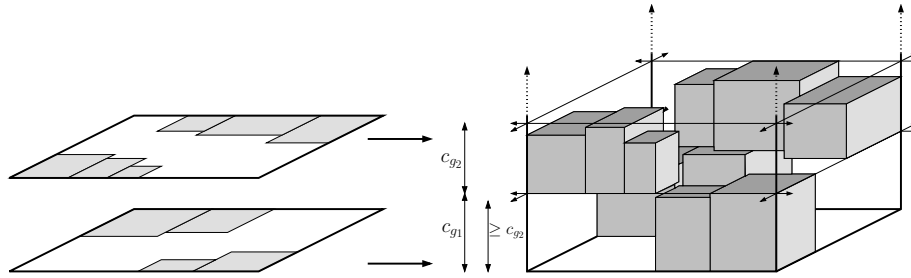


Figure 10: Layer packing of  $S_{small}$

omitting the thinnest layer. Thus we get

$$\sum_{i=1}^{l-1} c_{g_{i+1}} \leq 4\alpha c.$$

With this estimation we can bound the total height of the strip to:

$$H = \sum_{i=1}^l c_{g_i} = c_{g_1} + \sum_{i=2}^l c_{g_i} \leq \left(\frac{1}{2} + 4\alpha\right) c$$

In this basic analysis we use  $\alpha = 1$  as an upper bound (we need a more precise volume estimation for the improvement in the next step) and derive an absolute height limit of  $\frac{9}{2}c$ . Using the same idea as above - but this time the 3-dim equivalent - we cut the strip with vertical planes into 4 bins of size  $(a, b, c)$  and one bin of size  $(a, b, c/2)$  - see Figure 11. The items which are cut by the 4 planes

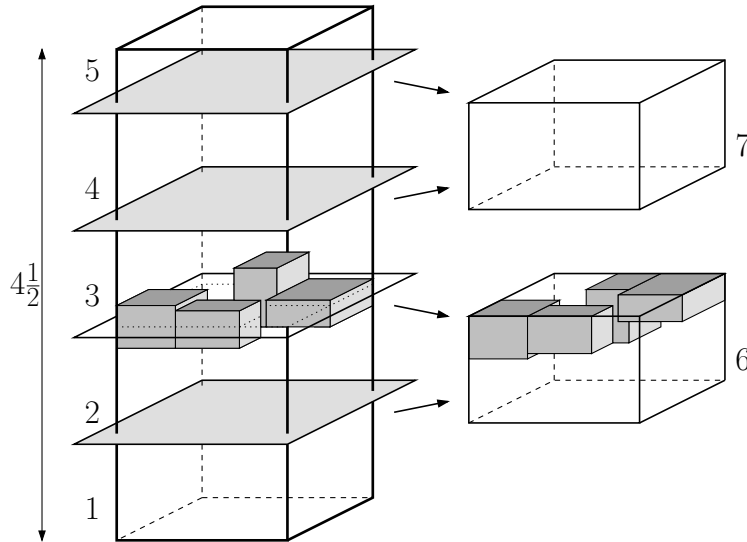


Figure 11: Layer packing of  $S_{small}$

can be put together into two more bins of size  $(a, b, c)$ . To do this, retain the

order of the intersections on two of the cutting planes and insert them bottom aligned and top aligned respectively into one bin. As the  $c$ -length is bounded by  $c/2$  the items do not overlap. Thus we can pack  $S_{small}$  into at most 7 bins which proves

**Theorem 4.1** (Cuboid packing). *There is a polynomial time approximation algorithm with performance ratio of at most  $16 + \epsilon$  for cuboid packing, for any  $\epsilon > 0$ .*

## 4.2 Improvement

As already mentioned in the modified abstract, Lemma 4.2, which is basic for the improvement of the algorithm, is not correct. Nevertheless the idea of separating the items into  $S_a, S_b, S_c, S_{big}$  and  $S_{small}$  and the Observation 1 are still correct and are used in [4] to derive a  $(9 + \epsilon)$ - and a  $(8 + \epsilon)$ -approximation. Furthermore the idea of using placeholder items might still be useful for improving on these results.

The basic  $(16 + \epsilon)$ -approximation algorithm finds a selection of items with the  $m$ -dimensional knapsack and packs this selection into up to 16 bins. This method is adopted from the  $(3 + \epsilon)$ -algorithm for rectangle packing in [8]. It is not surprising, that the algorithm can be improved by using the  $(2 + \epsilon)$ -algorithm of the previous part of this paper. To do this we have to change our perspective. We can not longer pack all items into a certain number of bins, but separate the selection into several sets of items and apply ‘as good as possible’-approximation algorithms on them. Therefore we need the following lemma

**Lemma 4.2.** *Given a set of items  $S$  which fulfills the  $m$ -dimensional knapsack instance of the previous section, an algorithm  $P$  that produces a partition  $S_1 \cup S_2 \cup \dots \cup S_l = S$  and a list of approximation algorithms  $A_1, A_2, \dots, A_l$  which have an approximation ratio of  $\delta_1, \delta_2, \dots, \delta_l$  on  $S_1, S_2, \dots, S_l$ , respectively, there is an approximation algorithm  $A$  for cuboid packing with approximation ratio at most  $\delta_1 + \delta_2 + \dots + \delta_l + \epsilon$  for every  $\epsilon > 0$ .*

*Proof.* Let  $A$  be the algorithm which first finds a selection of items  $S$  with the  $m$ -dimensional knapsack of the previous section, using  $\epsilon' = \epsilon / (\delta_1 + \delta_2 + \dots + \delta_l)$ . Then it applies  $P$  on  $S$  and  $A_i$  on  $S_i$  for  $1 \leq i \leq l$ . Let  $\bar{S}_i$  be the output of the algorithm  $A_i$ .  $A$  outputs  $\max(p(\bar{S}_1), p(\bar{S}_2), \dots, p(\bar{S}_l))$  together with the corresponding packing. We prove that  $OPT(I) \leq (\delta_1 + \delta_2 + \dots + \delta_l + \epsilon)A(I)$ .

$$\begin{aligned}
OPT(I) &\leq (1 + \epsilon')p(S) \\
&= (1 + \epsilon')(p(S_1) + p(S_2) + \dots + p(S_l)) \\
&\leq (1 + \epsilon')(\delta_1 p(\bar{S}_1) + \delta_2 p(\bar{S}_2) + \dots + \delta_l p(\bar{S}_l)) \\
&\leq (1 + \epsilon')(\delta_1 + \delta_2 + \dots + \delta_l) \max(p(\bar{S}_1), p(\bar{S}_2), \dots, p(\bar{S}_l)) \\
&= (\delta_1 + \delta_2 + \dots + \delta_l + \epsilon)A(I)
\end{aligned}$$

□

The mistake in this prove is that  $p(S_i) \leq \delta_i p(\bar{S}_i)$  doesn't hold. Instead we have  $OPT(S_i) \leq \delta_i p(\bar{S}_i)$ . The former equation would only be

true, if algorithm  $A_i$  packed all items  $S_i$  into the bins, which is not true for the  $(2 + \epsilon)$  algorithm.

Notice, that the final  $\epsilon$  can also absorb other arbitrarily small number in the approximation ratios of  $A_1, \dots, A_l$ . The algorithm  $P$  is only needed to ensure later, that the sets  $S_i$  have certain properties from which the algorithm  $A_i$  can benefit.

The main idea of the improved algorithm is to consider the volumes of the different sets. Obviously the strip height of  $S_{small}$  decreases if the volume of these items is smaller. We make two observations before we give the algorithm.

**Observation 1** (Packing small volume of  $S_a$ ).  $S_a$  can be packed into one bin if  $Vol(S_a) \leq \frac{1}{4} abc$ .

*Proof:* If  $Vol(S_a) \leq \frac{1}{4} abc$  then  $S_{bc}(S_a) \leq \frac{1}{2} bc$  where  $S_{bc}(S_a)$  is the sum of the  $bc$ -side planes of the items in  $S_a$ . As there are no high item in  $S_a$  (which means long in direction  $c$ ), the  $bc$ -projections of the items can be packed into one bin of size  $(b, c)$  with Steinbergs algorithm. Finally the items can be erected in  $a$ -direction as usual. Obviously the same applies to the sets  $S_b$  and  $S_c$  and we can include  $S_{big}$  to any of these sets.

**Observation 2** (Packing small volume of  $S_{small}$ ).  $S_{small}$  can be packed into  $f(\alpha)$  bins if  $Vol(S_{small}) \leq \alpha abc$ . Where  $f(\alpha)$  is defined as

$$f(\alpha) = \begin{cases} 7 & \text{if } 7/8 < \alpha \leq 1 \\ 6 & \text{if } 6/8 < \alpha \leq 7/8 \\ 5 & \text{if } 5/8 < \alpha \leq 6/8 \\ 4 & \text{if } 3/8 < \alpha \leq 5/8 \\ 3 & \text{if } 2/8 < \alpha \leq 3/8 \\ 2 & \text{if } 1/8 < \alpha \leq 2/8 \\ 1 & \text{if } 0 \leq \alpha \leq 1/8 \end{cases}$$

Unfortunately the funktion  $f(\alpha)$  can not be defined directly because of an odd gap between the  $\alpha$ -values of  $3/8$  and  $5/8$ . Nevertheless the proof is quite straightforward.

*Proof.* Recall, that we analyzed the height of the strip packing for the small items already bearing in mind their total volume. Therefore we can use the resulting strip height of  $\leq (\frac{1}{2} + 4\alpha)c$ . Now it is easy to see, that for certain strip heights a certain number of bins is sufficient. Exemplarily, a strip height of up to  $3c$  can be packed into 4 bins (we have two cutting planes whose intersection can be packed together into one bin) - and this height is sufficient for  $\alpha \leq 5/8$ . Notice, that for odd numbers of cutting planes (for example the height of  $3\frac{1}{2}c$  for  $\alpha \leq 6/8$  has 3 cutting planes) the remaining last regular bin might be only half-filled and can therefore hold the surplus intersection.  $\square$

Now we give a first improved algorithm with approximation ratio  $(10 + \epsilon)$  which we improve with a last tweak to  $(9 + \epsilon)$  afterwards.

1. Find a selection  $S$  of items with the  $m$ -dimensional knapsack problem as in the basic algorithm,

2. separate the items in  $S$  into the sets  $S_a, S_b, S_c$  and  $S_{small}$  as defined above (but consider a potential big item belonging to  $S_a$ ),
3. if  $Vol(S_a) \leq 1/4 abc$  than pack  $S_a$  into one bin, otherwise use the  $(2 + \epsilon)$ -algorithm for rectangle packing for the  $bc$ -projection and erect the items on the pattern (deal similarly with  $S_b$  and  $S_c$ ),
4. pack  $S_{small}$  into a strip and transfer the strip into a minimal number of bins according to  $f(\alpha)$ ,
5. choose the bin with the highest profit.

*Analysis:* We consider four different cases:

1.  $Vol(S_a), Vol(S_b), Vol(S_c) \leq 1/4 abc$   
In this case we need 3 bins for  $S_a, S_b$  and  $S_c$  and (like in the basis algorithm) 7 bins for  $S_{small}$  - summing up to 10 bins in total.
2.  $Vol(S_a) > 1/4 abc$  and  $Vol(S_b), Vol(S_c) \leq 1/4 abc$   
Using Lemma 4.2 we get  $\delta_a = 2 + \epsilon$  for packing  $S_a$  and  $\delta_b = \delta_c = 1$  for packing  $S_b$  and  $S_c$  (because they are packed completely in one bin each). Finally  $\alpha \leq 6/8$  and therefore  $\delta_{small} = 5$  (as  $S_{small}$  can be packed into 5 bins). In total we get  $\delta_a + \delta_b + \delta_c + \delta_{small} = 9 + \epsilon$ .
3.  $Vol(S_a), Vol(S_b) > 1/4 abc$  and  $Vol(S_c) \leq 1/4 abc$   
Analogue we get  $\delta_a = \delta_b = 2 + \epsilon, \delta_c = 1$  and  $\delta_{small} = 4$  (as  $\alpha \leq 4/8$ ) - summing up to  $\delta_a + \delta_b + \delta_c + \delta_{small} = 9 + 2\epsilon$ .
4.  $Vol(S_a), Vol(S_b), Vol(S_c) > 1/4 abc$   
Analogue we get  $\delta_a = \delta_b = \delta_c = 2 + \epsilon$  and  $\delta_{small} = 2$  (as  $\alpha \leq 2/8$ ) - summing up to  $\delta_a + \delta_b + \delta_c + \delta_{small} = 8 + 3\epsilon$ .

As we can interchange  $S_a, S_b$  and  $S_c$  we proved the approximation ratio of  $(10 + \epsilon)$  for our improved algorithm. Obviously only the first case causes difficulties for the improvement to  $(9 + \epsilon)$ . In this case  $S_a, S_b$  and  $S_c$  contain only very limited volume - this leads to the idea to pack some of the small items together with the long items.

If the total volume of items packed together with  $S_a, S_b$  and  $S_c$  into 3 bins is  $\geq \frac{1}{8} abc$  then the remaining small items can be packed into 6 bins (see Observation 2). Hence, assume that  $Vol(S_a \cup S_b \cup S_c) < \frac{1}{8} abc$ . W.l.o.g  $Vol(S_a) \leq Vol(S_b) \leq Vol(S_c)$  and therefore  $Vol(S_a), Vol(S_b) < \frac{1}{16} abc$ . Obviously the sums of the side planes of the items in  $S_a$  and  $S_b$  are therefore bounded by  $S_{bc}(S_a) < \frac{1}{8} bc$  and  $S_{ac}(S_b) < \frac{1}{8} ac$ . We can add a placeholder item  $P_a$  of size  $(a, b/2, c/2)$  in  $S_a$  and a placeholder item  $P_b$  of size  $(a/2, b, c/2)$  in  $S_b$  - and pack  $S_a$  and  $S_b$  together with their placeholder items into one bin each. Now we add some items of  $S_{small}$  into the space of the placeholder item. Therefore we define

$$S_a^* = \{r \in S_{small} \mid b_r c_r \geq \frac{1}{8} bc\}$$

$$S_b^* = \{r \in S_{small} \mid a_r c_r \geq \frac{1}{8} ac\}$$

and pile up items of  $S_a^*$  in  $a$ -direction into placeholder item  $P_a$  until we either exceed a height of  $a/2$  or run out of items. Similarly we pile up items of  $S_b^*$  in

$b$ -direction into  $P_b$ . As items can be included in both of the sets  $S_a^*$  and  $S_b^*$  we pay attention that each item is only included in one placeholder. If the item supply from  $S_a^*$  and  $S_b^*$  is sufficient,  $P_a$  and  $P_b$  have a volume of  $\geq \frac{1}{16}abc$  each (ground  $\geq \frac{1}{8}$  and height  $\geq \frac{1}{2}$  in the fitting dimensions), summing up to  $\geq \frac{1}{8}abc$ . This permits to pack the remaining small items in  $S_{small}$  into 6 bins.

Now assume w.l.o.g a lack of items in  $S_a^*$ . Then it is possible to pack all items of  $S_a^*$  into one of the two placeholders (items that are also contained in  $S_b^*$  can be put into  $P_b$  too) and the  $bc$ -side plane of each remaining small item in  $S_{small}$  is  $< \frac{1}{8}bc$ . Change the direction of the layer packing of the remaining items into direction  $a$ . Considering the analysis of the layer packing we can improve the layer filling from  $S_i \geq \frac{1}{4}bc$  to  $S_i \geq \frac{3}{8}bc$ . Taking this into account we derive a strip height of  $H \leq (\frac{1}{2} + \frac{8}{3}\alpha)a$  which is  $\leq 3\frac{1}{2}a$  for  $\alpha = 1$ . Therefore we can pack the remaining small items into 5 bins. We proved

**Theorem 4.2** (Improved cuboid packing). *There is a polynomial time approximation algorithm with performance ratio of at most  $9 + \epsilon$  for cuboid packing, for any  $\epsilon > 0$ .*

**Remark:** The running time of this improved algorithm is dominated by several applications of the  $(2 + \epsilon)$ -algorithm for rectangle packing and therefore of extremely high order. It is also possible to derive a  $(10 + \epsilon)$ -algorithm without using the  $(2 + \epsilon)$ -algorithm for rectangle packing and thus getting a much better running time.

## 5 Cuboid packing does not admit an asymptotic PTAS

We will regard the objective of maximizing the number of packed items in this section. Apparently, this is a special case of weighted cuboid packing with a uniform profit of 1.

Bansal and Sviridenko [1] showed the inapproximability of 2-dim bin packing by giving a reduction from the bounded 3-dimensional matching problem to a special subproblem of 2-dim bin packing. We adopt their result and give a reduction from the subproblem to cuboid packing. First, we define the subproblem  $BP^*$  - which is not formally done in [1]. Second, we give a reduction from  $BP^*$  to cuboid packing and thereby show the inapproximability of cuboid packing.

Let  $OPT_{bin}(r_1, \dots, r_n)$  be the minimal number of unit size bins in which the rectangles  $r_1, \dots, r_n$  can be packed. The subproblem  $BP^*$  of 2-dim bin packing is defined as follows. Given an integer  $T$  and a list  $I$  of  $n$  rectangles  $r_i = (a_i, b_i)$  such that  $n \in [\frac{14}{3}T, 8T]$  and  $OPT_{bin}(r_1, \dots, r_n) \geq T$ , find a packing for the rectangles in  $I$  into a minimum number of bins. Bansal and Sviridenko [1] showed that it is  $NP$ -hard to distinguish between instances of  $BP^*$  where  $OPT_{bin}(I) = T$  and instances where  $OPT_{bin}(I) \geq (1 + \epsilon)T$  for some constant  $\epsilon > 0$ .

We construct an input instance  $I'$  for cuboid packing from an instance  $I$  for  $BP^*$ . For each rectangle  $r_i = (a_i, b_i)$  we define a cuboid item  $r'_i = (a_i, b_i, 1/T)$ . The cuboid bin is of unit size  $(1, 1, 1)$ .

**Definition 1** (Layer packing). *A layer packing is a packing where the lower sides of all items are aligned to a horizontal plane in the height of  $k/T$ , where  $k \in \{0, \dots, T-1\}$ .*

**Lemma 5.1.** *Every packing of a selection of items of  $I'$  can be rearranged to a layer packing.*

*Proof.* Sweep with a horizontal plane through the bin and lower all intersected items to the previous layer. Obviously no items overlap after the sweeping procedure.  $\square$

**Observation 3** (Equivalence of packing in bins and layers). *If the rectangles  $(r_1, r_2, \dots, r_k)$  can be packed into one bin, the corresponding cuboid items  $(r'_1, r'_2, \dots, r'_k)$  can be packed into one layer of size  $(1, 1, 1/T)$  by arranging the items on the bottom of the layer in the same pattern.*

*Conversely, if the cuboid items  $(r'_1, r'_2, \dots, r'_k)$  can be packed into one layer of size  $(1, 1, 1/T)$ , the corresponding rectangles  $(r_1, r_2, \dots, r_k)$  can be packed into one bin.*

**Lemma 5.2.** *If the items  $(r'_1, r'_2, \dots, r'_k)$  can be packed into the cuboid bin, the corresponding items  $(r_1, r_2, \dots, r_k)$  can be packed into at most  $T$  bins.*

*Proof.* Regard a layer packing of  $(r'_1, r'_2, \dots, r'_k)$  and pack each layer into one bin.  $\square$

If  $OPT_{bin}(I) = T$ , we can pack the cuboid items corresponding to the rectangles of each bin into one layer and pile up all  $T$  layers in the cuboid bin. Thus all cuboid items can be packed and  $OPT_{cuboid}(I') = n$ . Now we assume, that every packing needs at least  $(1 + \epsilon)T$  bins. As all bins contain at least one item, the  $\epsilon T$  bins with the lowest number of items contain at least  $\epsilon T \geq \frac{\epsilon}{8}n$  items. We prove by contradiction, that  $OPT_{cuboid}(I') \leq (1 - \frac{\epsilon}{8})n$ . Therefore, assume that  $OPT_{cuboid}(I') > (1 - \frac{\epsilon}{8})n$ . This means, that more than  $(1 - \frac{\epsilon}{8})n$  items can be packed into the cuboid bin, which implies that all the corresponding items can be packed into  $T$  bins. The rectangles corresponding to the unpacked, less than  $\frac{\epsilon}{8}n \leq \epsilon T$  cuboid items, can be packed into one bin each, summing up to  $< \epsilon T$  bins, which makes up the contradiction.

As we could therefore distinguish between instances for  $BP^*$  where  $OPT_{bin}(I) = T$  and instances where  $OPT_{bin}(I) \geq (1 + \epsilon)T$  if cuboid packing would admit a PTAS, we proved that there is no PTAS if  $NP \neq P$ . Furthermore, as the gap holds for all instances (even for those with an arbitrary large number of required bins) we can expand the result to the following theorem:

**Theorem 5.1** (Inapproximability of Cuboid Packing). *If  $NP \neq P$ , there is no asymptotic polynomial time approximation scheme (APTAS) for cuboid packing.*

## 6 Future work

As the gap between the given approximation algorithm of  $(9 + \epsilon)$  and the inapproximability result is enormous, there is a lot to do in the future. In this last section we give a short overview of our ideas for the forthcoming approaches.

- We were able to improve the current result even more to a  $(8 + \epsilon)$ -approximation. We did not present the result in this paper because the proof requires 18 different cases to be handled. On the other hand the algorithm does contain one new idea which changes the grouping a little bit such that the items in the intersections of the strip packing can be packed more efficiently. We try to find a more convincing proof for this algorithm in order to present it.
- The main difficulty for cuboid packing is the oddness of the shapes of the items. In the worst case items will always be flat in one direction and long in the other directions - thus using almost no space but making it difficult to pack together with flat items in other directions. We think that the current approach of using the  $m$ -dimensional knapsack to make a selection of items which are then separated and packed into several bins is nearly exhausted. Therefore we need to think about better ways of preselecting the items.
- For three-dimensional strip packing Miyazawa and Wakabayashi [16] gave a algorithm with asymptotic approximation ratio of 2.67. Assuming a sufficient supply of fitting items a volume usage of 2.67 is therefore possible. Maybe some of the ideas of Miyazawa and Wakabayashi can also be applied to cuboid packing.
- Obviously a  $\delta$ -approximation algorithm for cuboid packing gives a  $\delta$ -approximation algorithm for rectangle packing. It would be interesting to examine whether it is possible to derive a  $\gamma \cdot \delta$ -approximation for rectangle packing with a  $\delta$ -approximation for cuboid packing for some  $\gamma < 1$ .
- Finally a further generalisation to multidimensional packing can be considered. Of course we already showed the inapproximability of all dimensions higher than 2 with our proof for dimension 3. Nevertheless we assume, that multidimensional packing for fixed dimensions is in APX as we can reduce every dimension to the lower ones.

## References

- [1] Bansal, N. and Sviridenko, M. (2004). New approximability and inapproximability results for 2-dimensional bin packing. Proc. 15th ACM-SIAM Symposium on Discrete Algorithms, 196-203.
- [2] Caprara, A. (2002). Packing 2-dimensional bins in harmony. Proc. 43rd Foundations of Computer Science, 490-499.
- [3] Chandra, A.K., Hirschberg D. S. and Wong, C. K. (1976). Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science* 3, 293-304.
- [4] Diedrich, F., Harren, R., Jansen, K., Thöle, R. and Thomas, H. (to appear). Approximation algorithms for 3D orthogonal knapsack.
- [5] Fishkin, A., Gerber, O. and Jansen, K. (2004). On weighted rectangle packing with large resources. Proc. 3rd IFIP International Conference on Theoretical Computer Science, 237-250.

- [6] Jansen, K. and Porkolab, L. (2001). Improved approximation schemes for scheduling unrelated parallel machines. *Mathematics of Operations Research* 26, 324-338.
- [7] Jansen, K. and Stee, R. (2005). On strip packing with rotations. Proc. 37th ACM Symposium on Theory of Computing, 755-761.
- [8] Jansen, K. and Zhang, G. (2004). On rectangle packing: maximizing benefits. Proc. 15th ACM-SIAM Symposium on Discrete Algorithms, 204-213.
- [9] Jansen, K. and Zhang, G. (2004). Maximizing the number of packed rectangles. Proc. 9th Scandinavian Workshop on Algorithm Theory, 362-371.
- [10] Kenyon, C. and Rémila, E. (2000). A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research* 25, 645-656
- [11] Khachiyan, L. (1979). A polynomial algorithm in linear programming. Soviet Mathematics Doklady 20, 191-194.
- [12] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 373-395.
- [13] Leung, J.Y.-T, Tam, T.W., Wong, C.S., Young, G.H. and Chin, F.Y.L. (1990). Packing squares into a square. *Parallel and Distributed Computation* 10, 271-275
- [14] Shmoys, D. and Tardos, E. (1993). An approximation algorithm for the generalized assignment problem. *Mathematical Programming* 62, 461-474.
- [15] Steinberg, A. (1997). A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing* 26, 401-409.
- [16] Miyazawa, F.K. and Wakabayashi, Y. (1997). An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica* 18, 122-144
- [17] Miyazawa, F.K. and Wakabayashi, Y. (1999). Approximation algorithms for the orthogonal z-oriented three-dimensional packing problem. *SIAM Journal on Computing*, 1008-1029