Software ENgineering

This Side Up!

L. Epstein, R. van Stee

Report SEN-E0412 September 2004

ISSN 1386-369X

This Side Up!

ABSTRACT
We consider two- and three-dimensional bin packing problems where 90º rotations are allowed. We improve all known asymptotic performance bounds for these problems. In particular, we show how to combine ideas from strip packing and two-dimensional bin packing to give a new algorithm for the three-dimensional strip packing problem where boxes can only be rotated sideways. We propose to call this problem `This side up'. Our algorithm has an asymptotic performance bound of 9/4.

2000 Mathematics Subject Classification: 68W25,68W40
1998 ACM Computing Classification System: F.2.2
Keywords and Phrases: bin packing;rotations;approximation algorithms

This Side Up!

Leah Epstein[*] Rob van Stee[†]

September 7, 2004

Abstract

We consider two- and three-dimensional bin packing problems where 90° rotations are allowed. We improve all known asymptotic performance bounds for these problems. In particular, we show how to combine ideas from strip packing and two-dimensional bin packing to give a new algorithm for the three-dimensional strip packing problem where boxes can only be rotated sideways. We propose to call this problem 'This side up'. Our algorithm has an asymptotic performance bound of $9/4$.

1 Introduction

The study of multi-dimensional packing problems gained an increasing interest in the last few years [1, 2, 4, 7]. A main trend was the study of offline and online packing algorithms for oriented items which are rectangles or boxes. Given a large supply of bins which are squares, or cubes, or a strip of infinite height, the goal is to pack items efficiently, without rotation. This problem clearly has applications; however, in many applications, there is no reason to exclude the option of changing the orientation of items before assignment. Some applications may allow rotation only in certain directions.

In this paper we study several rotatable packing problems. The same problem is also known as "packing of non-oriented items" [5]. All packing problems involve an input which is a set of items. In "strip packing" problems, the goal is to pack the items into a strip of unlimited height, so as to minimize the maximum height ever used. In "bin packing" problems, the goal is to use a minimum number of bins for the packing. The items always need to be packed without overlap. The exact structure of the strip, bins and items depends on the specific problem.

The two-dimensional bin packing problem with rotations is defined as follows. The bins are unit squares and the items are rectangles that may be rotated by 90°. In the strip packing version, the strip is two-dimensional, with a base of width one and infinite height. In the three-dimensional bin packing problem with rotations, the bins are three-dimensional cubes, and the items are non-oriented three-dimensional boxes, rotatable by 90° in all possible directions. In the strip packing version we pack these items into a three-dimensional strip with a base which is a unit square, and again, infinite height.

In three dimensions, we can also consider the case of items that may be rotated so that the width and length are interchanged, however the height is fixed. We call this problem "This Side up", as it has applications in packing of fragile objects, where a certain face of the box must be placed on top. This three-

[*]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. `lea@idc.ac.il`. Research supported by Israel Science Foundation (grant no. 250/01).

[†]Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands. `Rob.van.Stee@cwi.nl`. Research supported by the Netherlands Organization for Scientifi c Research (NWO), project number SION 612-061-000.

dimensional problem, as the rotatable problems has two versions: packing into a three-dimensional strip (also called "the z-oriented 3-D packing problem" [10]) and packing into three-dimensional bins.

The standard measure of algorithm quality for box packing is the *asymptotic performance ratio*, which we now define. For a given input sequence σ, let $\mathcal{A}(\sigma)$ be the number of bins used by algorithm $\mathcal{A}$ on σ. Let $\mathrm{OPT}(\sigma)$ be the minimum possible number of bins used to pack items in σ. The *asymptotic performance ratio* for an algorithm $\mathcal{A}$ is defined to be

$$\mathcal{R}_{\mathcal{A}}^{\infty} = \limsup_{n\to\infty} \sup_{\sigma} \left\{ \frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)} \middle| \mathrm{OPT}(\sigma) = n \right\}.$$

Previous Results: The oriented packing problems have been widely studied. The best result for two-dimensional packing into bins is 1.691, due to Caprara [2]. See references in [1, 2, 4, 7] for further results on oriented packing problems. The square and cube packing (into bins) problems are special cases of rotatable packing. An APTAS for these problems was given in [1] and independently in [4].

Although the possibility of allowing rotations was already mentioned in [3], there has been relatively little research into this subject from a worst-case perspective until recently. Fujita and Hada [8] considered the two-dimensional bin packing problem with rotations. They presented two online algorithms and claimed asymptotic performance ratios of at most 2.6112 and 2.56411. Epstein [6] showed that the first algorithm instead has an asymptotic performance ratio of at most 2.63889 and questioned the validity of the second algorithm. She also presented an online algorithm with asymptotic performance ratio slightly below 2.45.

Recently, Miyazawa and Wakabayashi [11] presented an *offline* approximation algorithm for two-dimensional bin packing with rotations with an asymptotic performance bound of 2.64. It is most likely that the extended abstract [6] as well as the earlier paper [8] were unknown to those authors.

The paper [11] also considers several other problems with rotatable items and gives an upper bound of 2.64 for the This Side Up problem which was also considered in [9, 10]. The paper [10] demonstrates a reduction from the general three-dimensional strip packing problem with rotations to the This Side Up problem in a strip, but this reduction does not hold for the case considered in this paper, where the three-dimensional strip always has a square base of side 1.

Our Results. We improve upon the best known results for all the above problems.

- An algorithm of asymptotic performance ratio $3/2 = 1.5$ for two-dimensional rotatable strip packing. This improves on the bound 1.613 in [11].
- An algorithm of asymptotic performance ratio $9/4 = 2.25$ for two-dimensional rotatable packing into bins. This improves on the on-line algorithm in [6] that has an asymptotic performance ratio of slightly less than 2.45. Although this algorithm basically consists of many (easy) cases, it has the advantage that it can easily be adapted to the more complex problems listed below.
- An algorithm which combines methods of the two above algorithms and has asymptotic performance ratio $9/4 = 2.25$ for the "This side up" problem in a strip. This is the main result of the paper. This improves the bound of [11] which is 2.64.
- An adaptation of the previous algorithm to packing of rotatable items in a three-dimensional strip, with the same asymptotic performance ratio $9/4 = 2.25$. This improves the bound of [11] which is 2.76.
- A simple adaptation of the two previous algorithms for the bin packing versions of these problems, with asymptotic performance ratio $9/2 = 4.5$. This improves the bound of [11] for three-dimensional bin packing of rotatable items, which is 4.89.

2 Two-dimensional strip packing

We assume that all items have height and width at most 1. The strip has width 1 and unbounded height. As a subroutine for our algorithm, we use the well-known algorithm First Fit Decreasing Height (FFDH). The following theorem was proved by Coffman et al. in 1980 [3].

Theorem 1 *Let L be any list of rectangles ordered by non-increasing height such that no rectangle in L has width exceeding $1/m$ for some $m \geq 2$. Then $FFDH(L) \leq (1 + 1/m)A(L) + 1$, where $A(L)$ is the total area of the items in L.*

Items that have width and height greater than $1/2$ are called *big*. These items are rotated so that their width is not smaller than their height. For items that have only one dimension greater than $1/2$, we rotate them so that this dimension is the height.

1. The big items are stacked at the bottom of the strip, in order of decreasing width and aligned with the left side of the strip. Denote the height needed for this packing by h_1.
2. Denote the height at which the first item of width at most $2/3$ is packed by h_1' ($0 \leq h_1' \leq h_1$). If $h_1' < h_1$, define a substrip of width $1/3$ that starts at height h_1', at the right side of the strip. Pack items that have widths in $(0, 1/6]$ inside this strip using FFDH, until all these items are packed or until the next item to be packed would be placed (partially) above height h_1.
3. If all items that have width in $(0, 1/6]$ have now been packed:
   (a) Stack items of widths in $(1/6, 1/2]$ at the right side of the bin, on top of the substrip from step 2. Place these items in order of *increasing width*. Each item is placed as low as possible, at the extreme right of the bin, under the constraint that it does not overlap with previously placed items. Do this until all such items are packed or the next item to be packed would be placed (partially) above height h_1.
   (b) Place the unpacked items of width in $(1/3, 1/2]$ in two stacks starting at height h_1 by each time adding an item to the shortest stack. Pack the unpacked items of width in $(1/6, 1/3]$ using FFDH.
4. Else, place all remaining items above height h_1 using the algorithm FFDH.

Theorem 2 *For this algorithm, we have $\text{ALG}_1(L) \leq \frac{3}{2}\text{OPT}(L) + 3$ for any input list L.*

Proof We start with the simple inequality. We have

$$\text{OPT}(L) \geq h_1 \tag{1}$$

because the packing in step 1 is optimal for the big items. If no items are packed in step 3(b) or 4 (all items are already packed) then it follows that the algorithm gives an optimal packing.

Denote the list of items packed in step i by L_i, and the height of that packing by h_i. By Theorem 1, for Step 4 we have $h_4 \leq \frac{3}{2}A(L_4) + 1$ and therefore

$$A(L_4) \geq \frac{2}{3}(h_4 - 1). \tag{2}$$

Denote the height of the substrip in step 2 by h_2 (the top of the substrip is at height h_1 or at the top of the highest item placed inside it), and the list of items packed in it by L_2. Note that all items placed inside this substrip have width at most half the width of the substrip.

Suppose we multiply all the widths of items in the substrip by 3, as well as the width of the substrip itself. Denote the new area of the list L_2 by $A'(L_2)$. Then Theorem 1 implies $h_2 \leq \frac{3}{2}A'(L_2) + 1$. Since $A'(L_2) = 3A(L_2)$, we can conclude

$$A(L_2) \geq \frac{2}{9}(h_2 - 1). \tag{3}$$

Case 1 Suppose no items are packed in Step 3.

The only interesting case is where some items are packed in Step 4. Consider Step 2. Below the substrip, a width of at least $2/3$ is covered everywhere by packed items. Next to the substrip, a width of at least $1/2$ is covered, and in the substrip, we have (3). On this part of the strip we have therefore packed items of total area at least $\frac{1}{2}h_2 + \frac{2}{9}(h_2 - 1) \geq \frac{2}{3}(h_2 - 1)$.

We can conclude that on the main strip, we have packed items of total area at least $\frac{2}{3}(h_1 - 2)$ below a height of h_1, since $h_2 \geq h_1 - h_1' - 1$ in the present case. In summary, we have that our algorithm packs items to a height of $h_1 + h_4$, and the area of these items is at least $\frac{2}{3}(h_1 - 2 + h_4 - 1)$ by (2). This implies that $\text{OPT}(L) \geq \frac{2}{3}(h_1 + h_4 - 3)$ and therefore $\text{ALG}_1(L) \leq \frac{3}{2}\text{OPT} + 3$.

Case 2 Some items are packed in Step 3. Denote the height of the packing in Step 3(b) by h_3 (i.e. the extra height that is packed above h_1). We have $\text{ALG}_1(L) = h_1 + h_3$.

In Step 3(a), some item(s) may not always be placed directly on top of the previous item. This happens if they would overlap with an item from Step 1. Suppose each item placed in Step 3(a) is placed directly on top of the previous item in that Step. Then a width of $1/2 + 1/6 = 2/3$ is everywhere covered by items above h_1' and below $h_1 - 1$. Below h_1', a width of $2/3$ is covered everywhere.

Above h_1, a width of at least $2/3$ is covered in the part of the strip that is used in Step 3(b), apart from at most two parts of total height at most 2. We find $\text{OPT}(L) \geq \frac{2}{3}(h_1 - 1 + h_3 - 2)$ and therefore $\text{ALG}_1(L) = h_1 + h_3 \leq \frac{3}{2}\text{OPT}(L) + 3$.

Now suppose there is an item placed in Step 3(a) that is not placed on top of its predecessor, or the first item in Step 3(b) would have been placed in Step 3(a) if its width had been smaller (i.e. it is placed in 3(b) because of its width, not because of its height). Denote the width of the *last* such item by w, and the height at which this item is placed by $h_3' \in [h_1', h_1]$. Then up to height $\min(h_3', h_1 - 1)$, a width of at least $1 - w$ is covered by items from Step 1. If $h_3' < h_1 - 1$, then above height h_3' and until height at least $h_1 - 1$ there are everywhere items from Step 3 and therefore a width of at least $1/2 + 1/6 = 2/3$ is covered by items.

Suppose $w > 1/3$. Then only items of width in $[w, 1/2] \subset (1/3, 1/2]$ remain to be packed in Step 3(b). If $h_3' < h_1 - 1$, we have

$$\begin{aligned} \text{OPT}(L) &\geq (1 - w)h_3' + \frac{2}{3}(h_1 - 1 - h_3') + 2w(h_3 - 1) \qquad (4) \\ &\geq (1 - w)(h_1 - 1) + 2w(h_3 - 1) \qquad (5) \end{aligned}$$

If $h_3' \in (h_1 - 1, h_1]$, then we have (5) immediately. By combining (5) with (1), it can be seen that $\text{ALG}_1(L) = h_1 + h_3 \leq \frac{3}{2}\text{OPT}(L) + 2$.

Suppose $w \leq 1/3$. Then a width of at least $2/3$ is covered in the part of the strip that is used in Step 3(b), apart from at most two parts of total height at most 2. We find $\text{OPT}(L) \geq \frac{2}{3}(h_1 - 1 + h_3 - 2)$ and therefore $\text{ALG}_1(L) = h_1 + h_3 \leq \frac{3}{2}\text{OPT}(L) + 3$. □

3 Two-dimensional bin packing

We apply a first partition of items to types in the following way. We rotate all items such that the length is at least as large as the width. We call this the *standard* orientation, and the other one the *reversed* orientation. The one-dimensional intervals we use in the initial partition are

- $(2/3, 1]$ (type 0)
- $(1/2, 2/3]$ (type 1)
- $(1/(i+1), 1/i]$ for $i = 2, \ldots, 8$ (type i)
- $(0, 1/9]$ (type 9)

A two-dimensional item is of type (i, j) if its width is of type i and its length is of type j. Clearly $i \geq j$ due to the orientation we defined. In some cases, we will use a finer classification for type 1. We let type $1a = (1/2, 11/20]$, type $1b = (11/20, 3/5]$ and type $1c = (3/5, 2/3]$.

There are four types which we will call *large*. We will begin by defining their packing. Each such type is packed independently of the other ones. We pack items of types $(1, 1)$, $(1, 0)$ and $(0, 0)$ one per bin, always in the left bottom corner of the bin and in standard orientation. The items of type $(2, 1)$ are further classified according to their length (largest dimension). Items of type $(2, 1a)$ are packed two per bin, both in reverse orientation, touching the same edge of the bin and each other, with one of them in the left bottom corner of the bin. The same holds for items of type $(2, 1b)$ and $(2, 1c)$ (but the items of these three subtypes are not packed together in any bin).

There are also four *medium* types. Items of type $(2, 2)$ are packed four per bin (the bin is first partitioned into four identical sub-bins). Items of type $(i, 0)$ are packed i per bin for $i = 2, 3, 4$.

After the packing of the large and medium types is completed, smaller items are added. They are first added into bins which contain large items. If some items remain unpacked after those bins are considered, they are packed into empty bins.

Bins containing items of the types $(0, 0)$, $(2, 2)$, $(2, 0)$, $(3, 0)$ and $(4, 0)$ do not receive smaller items. We note that all these bins are packed so that a fraction of at least $4/9$ of their area is occupied, except possibly the last bin for each of the last four types. This follows from the types and the amounts of items per bin.

The performance bound of $9/4$ follows from one of the two following reasons.

1. If no new bins are opened for smaller items, we use a weighting function for the analysis. Those functions are usually useful in analyzing on-line algorithms. Here we use it to analyze an offline algorithm.

2. If at least one bin was opened for smaller items, we use an area based analysis. We show that all bins except a constant number have items of total area of at least $4/9$. Then we get $OPT \geq W \geq (4/9)(ALG_2 - c)$ which implies the performance ratio.

Case 1 The weighting function is defined in the following way. Small items get weight 0.

Type	$(0,0)$	$(1,0)$	$(1,1)$	$(2,1)$	$(2,0)$	$(2,2)$	$(3,0)$	$(4,0)$
Weight	1	1	1	$1/2$	$1/2$	$1/4$	$1/3$	$1/4$

The following claim is immediate from the definitions.

Claim 1 *All bins packed by our algorithm with large items, except possibly the last one for each (sub)type, contain a weight of 1.*

Claim 2 *A bin can contain at most nine items of both width and length larger than* $1/4$*.*

Proof See [7]. $\square$ $\square$

It follows from the same result that a bin can contain at most twenty-five items of both width and length larger than $1/6$.

Claim 3 *A bin can contain at most 9/4 of weight.*

Proof Consider a bin with a certain amount of weight. We may assume there is no item of type $(0, 0)$ or $(1, 0)$, because the smaller type $(1, 1)$ also has weight 1, and also no item of type $(2, 0)$ because $(2, 1)$ gives the same weight.

We will use Claim 2 to determine the highest possible weight in a bin by expressing all items as multiples of items of width and length just larger than $1/4$ or $1/6$. For instance, by cutting a (1,1) item halfway both horizontally and vertically, it can be seen that other items of 'worth' at most 5 items of width and length just larger than $1/4$ can be placed with it in one bin (otherwise this cutting would create a packing with more than 9 such items, contradicting Claim 2).

An overview can be found in the following table. Here the heading 'items $> 1/4$' means 'number of items of length and width more than $1/4$ that items of this type contain', etc.

Type	items $> 1/4$	items $> 1/6$	weight	weight per item $> 1/6$
$(1, 1)$	4	9	1	$1/9$
$(2, 1)$	2	6	$1/2$	$1/12$
$(2, 2)$	1	4	$1/4$	$1/16$
$(3, 0)$	2	4	$1/3$	$1/12$
$(4, 0)$	0	4	$1/4$	$1/16$

If there is no item of type $(1, 1)$, then by the last column, the weight per 'virtual' item of width and length larger than $1/6$ is at most $1/12$ which gives total weight of at most $25/12 < 9/4$.

Otherwise, by the second column and Claim 2, at most 2 items of type $(2, 1)$ or $(3, 0)$ can be in the bin together with the item of type $(1, 1)$. To get maximum weight, we should maximize the number of virtual items that have weight $1/12$ per item. We can have at most 12 such virtual items because there can be at most 2 items that cover 6 of them. This leaves at most 4 virtual items with weight per item $1/16$, giving additional weight of $1/4$. The total weight therefore is at most 1 (from the largest item) $+1$ (from the $(2, 1)$ items) $+1/4 = 9/4$. $\square$ $\square$

Case 2 It is left to show how small items are packed to keep a $4/9$ fraction of each bin occupied (except for a constant number of bins). Each bin will contain items of a given small type or set of types. For each type or set of types, we need to show how they are packed in the following three cases.

A. A bin that already contains a $(1, 0)$ item, or two $(2, 1)$ items.

B. A bin that already contains a $(1, 1)$ item.

C. An empty bin.

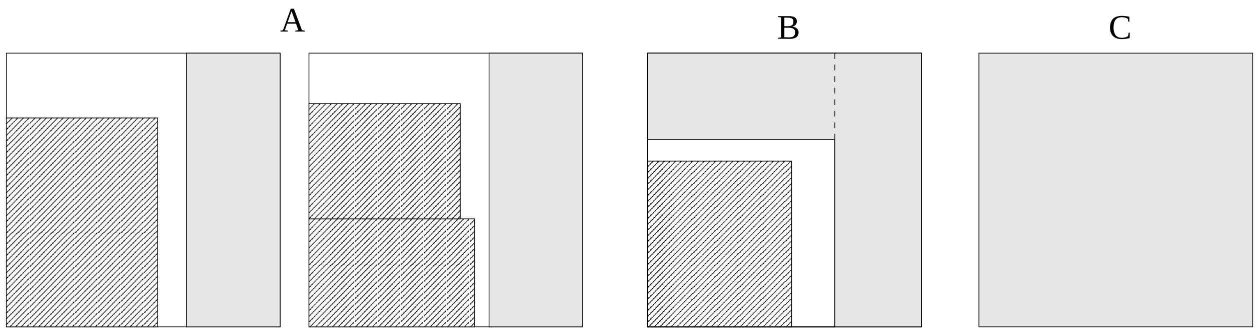

Figure 1: Unused areas in bins of types A, B and C. Small items are packed here

Consider the area left for further packing in the three cases. See Figure 1. For many small types, summarized in Table 1, the packing of the small items does not depend on the exact size of the large items that they are packed with.

In type A bins, there is a strip of width $1/3$ and length 1 that does not contain any items. Such a bin already contains an area of at least $1/3$.

In type B bins, the area outside of a square of $2/3$ by $2/3$ in the left bottom corner does not contain any items. We partition this L-shaped area in two rectangles, one of dimensions 1 and $1/3$ and the other of dimensions $2/3$ and $1/3$. The orientation is not important since rotations are allowed. We pack some number of small items in the larger rectangle and some number in the smaller rectangle; the numbers are written as a sum in the ‘items’ column for type B. These bins already contain an area of at least $1/4$.

In type C bins, we use the so-called side-by-side packing [6] to pack items. I.e., for type (i, j) items, we place ij of these items in an i by j grid at the bottom of the bin, and then (when possible) add some extra items in reverse orientation at the top of the bin.

Table 2 contains the items that are slightly more complicated to pack, at least in Type A and Type B bins. Here it is usually important what the exact width of the large items is. This is the reason that we classified type (2,1) items further: we can now be assured that if one of them has e.g. width $w \le 3/5$, the other one has this as well. (Note: to keep the analysis uniform, for these items we let the width be the *largest* size. The width of a pair can be taken arbitrarily as either the width of the first or the second item, due to our classification.)

In type A bins, we will now sometimes (where possible) use a strip of width $2/5$ or $9/20$ to pack small items, e.g. for type $(4, 1)$. In type B bins, the L-shaped free area will sometimes also be partitioned such that one strip of width $2/5$ or $9/20$ (and length 1) is created. For items of type $(6, 0)$, $(7, 0)$ and $(8, 0)$, the width of the largest strip depends on the width of the *small* items packed in there. The area that is already in a type A or B bin is of course different if we put restrictions on the width of the large items; it is given by the first term of the sum in the ‘area’ column.

Finally, there is one type that still remains to be packed. This type, $(9, 9)$, is described below.

Type $(9, 9)$ We show how to pack these items into square sub-bins of width and length $1/3$ so that inside each such sub-bin at least $4/9$ of its area is occupied. We begin by showing that this is a dense enough packing in all three cases.

A. We can create three sub-bins. We get a total area of $1/3 + 12/81 = 13/27 = 4/9$.

B. The item already packed in this bin has length and width no larger than $2/3$. Therefore we can create five sub-bins. The total occupied area would be $1/4 + 5 \cdot 4/81 = 161/324 > 4/9$.

Type	A		B		C	
	items	area $= 1/3+$	items	area $= 1/4+$	items	area
$(3,1)$	1	$1/8$	$1+1$	$1/4$	4	$1/2$
$(3,2)$	2	$1/6$	$2+1$	$1/4$	6	$1/2$
$(4,2)$	2	$2/15$	$2+1$	$1/5$	8	$8/15$
$(i,j)\quad 3 \le j \le i \le 4$	3	$3/25$	$3+2$	$1/5$	ij	$9/16$
$(5,j)\quad j=3,4,5$	5	$5/36$	$5+3$	$2/9$	$5j$	$5/8$
$(i,1)\quad i=6,7,8$	2	$1/9$	$2+2$	$2/9$	8	$4/9$
$(i,2)\quad i=6,7,8$	4	$4/27$	$4+2$	$2/9$	12	$12/27$
$(i,3)\quad i=6,7,8$	6	$1/6$	$6+4$	$5/18$	18	$1/2$
$(i,j)\quad i=6,7,8, j=4,5$	8	$4/27$	$8+4$	$2/9$	24	$4/9$
$(i,j)\quad 6 \le j \le i \le 8$	12	$12/81$	$12+8$	$20/81$	36	$4/9$

Type	shelves	area $= 1/3+$	shelves	area $= 1/4+$	shelves	area
$(9,i)\quad i=2,\ldots,8$	i	$2/3 \cdot 2/9$	$i+1$	$2/9$	i	$8/9 \cdot 2/3$

Table 1: All types that are combined on a single line of the table are packed together, except the types (i,j) for $3 \le j \le i \le 5$ in empty bins (type C bins) and the $(9,i)$ types.
For the $(9,i)$ types, shelves of length $1/i$ and width $1/3$ are created in type A and B bins. They are all filled to a length of $1/(i+1)$ and a width of $2/9$. In bins of type B, one extra shelf is created in the smaller part of the L-shape. In bins of type C, shelves of width 1 are created; they are filled to a width of $8/9$.

C. We create nine sub-bins and get a total area of $4/9$.

Next we explain the packing into sub-bins. We use Next Fit Decreasing Length (NFDL) to pack items into these sub-bins. All items are rotated such that their length is no smaller than their width, and then sorted by decreasing length. Then, the items are packed into levels using Next Fit, where the length of each level is the length of the first item placed in it. When the next item does not fit in the current level anymore, a new level is started, if necessary in a new sub-bin.

Since all items have width at most $1/9$, we find that each shelf is filled to a width of at least 2/9. Denote the length of shelf i by H_i. Let k be the number of shelves in the current bin. The first item that does not fit has length H_{k+1}. All items in shelf i have length at least H_{i+1}.

Then for each sub-bin except the last, the packed area is at least $2/9(H_2 + ... + H_{k+1}) > 2/9(1/3 - H_1) > 4/81$. This is a $4/9$ fraction of the area of the sub-bin which is $1/9$.

Theorem 3 *For any input list L, we have* $\text{ALG}_2(L) \le 9/4 \cdot \text{OPT}(L) + 41$.

Proof If no new bins are opened for small items, we have from Claim 1 that there are at most 7 bins with weight less than 1 (this cannot occur for the types $(0,0)$, $(1,0)$ and $(1,1)$). Combining this with Claim 3 gives $\text{ALG}_2(L) \le 9/4 \cdot \text{OPT}(L) + 7$.

If there are new bins opened for small items, then almost all bins contain an area of at least $4/9$. By the above analysis, this holds in this case for all bins that contain small items, except possibly the last such bin for each type that is packed separately. Note that it is also possible that we run out of a certain small type while we are packing a bin of type A, in this case this bin is not used further and has a bad area guarantee.

Type	Bins	Condition	items	area
$(4,1)$	A	subtype $(4,1)_a$	1	$1/3+9/80$
		$w > 11/20$	1	$11/30+1/10$
		$w \le 11/20$	2	$1/3+2/10$
	B		$1+1$	$1/4+1/5$
	C		5	$1/2$
$(5,0)$	A		1	$1/3+1/9$
	B	$w > 3/5$	$1+0$	$9/25+1/9$
		$w \le 3/5$	$2+0$	$1/4+2/9$
	C		5	$5/9$
$(5,1)$	A	$w > 3/5$	1	$2/5+1/12$
		$w \le 3/5$	2	$1/3+1/6$
	B	$w > 3/5$	$1+1$	$9/25+1/6$
		$w \le 3/5$	$2+1$	$1/4+1/4$
	C		6	$1/2$
$(5,2)$	A		2	$1/3+1/9$
	B	$w > 3/5$	$2+1$	$9/25+3/18$
		$w \le 3/5$	$4+1$	$1/4+5/18$
	C		10	$5/9$
$(i,0), i=6,7,8$	A		2	$1/3+4/(3i+3)$
	B	subtype $(i,0)_a$	$2+1$	$1/4+2/(i+1)$
		subtype $(i,0)_b$	$2+0$	$1/4+2\frac{i-1}{i}\frac{1}{i+1}$
	C		i	$2/3 \cdot i/(i+1)$

Type	Bins	Condition	shelves	area
$(9,i), i=0,1$	A		1	$1/3+2/9\cdot 1/2$
$(9,0)$	B	$w > 11/20$	1	$\frac{121}{400}+\frac{2}{3}\frac{2}{9}$
		$w \le 11/20$	1	$1/4+(\frac{9}{20}-\frac{1}{9})$
$(9,1)$	B		2	$1/4+4/9\cdot 1/2$
$(9,i), i=0,1$	C		1	$1/2\cdot 8/9$

Table 2: The variable w in the Condition column refers to the width of the big item(s) in the current bin of type A or B. Recall that the width is the *smallest* size of an item.
The type $(4,1)_a$ contains items of width in the interval $(9/40, 1/4]$. The type $(i,0)_a$ $(i = 6,7,8)$ contains items of width in $(2/3, \frac{i-1}{i}]$, the type $(i,0)_b$ contains items of width in $(\frac{i-1}{i}, 1]$. The types $(i,0)$ $(i = 4,\dots,8)$ are packed separately (in type B bins: both subtypes separately), the types $(9,0)$ and $(9,1)$ are packed together in type A and C bins. For the $(9,1)$ items in type B bins, we use two shelves of length 1 and $2/3$, both of width $1/3$. In both shelves, at least a width of $2/9$ and length of $1/2$ will be occupied.

Counting the number of types packed separately, there are 21 such types in Table 1 and 12 in Table 2. (Note that the type $(4,1)$ can only cause a single bin with a low area guarantee, because this cannot happen for subtype $(4,1)_a$ or for a bin of type A with $w > 11/20$.) Finally there is the type $(9,9)$.

Moreover, there can be at most 7 bins with large items that have no small items and area less than $4/9$: these are the bins that had weight less than 1 after packing the large items. Since the total area of the items is a lower bound on the optimal number of bins required to pack them, we find in this case $\text{ALG}_2(L) \le 9/4 \cdot \text{OPT}(L) + 41$. $\square$

4 This side up

We now show how to use the algorithm in the previous section to get a $9/4$-approximation algorithm for the This Side Up problem. Naively, one might think that one could simply group items of similar height and pack each group using the algorithm from the previous section (ignoring the height of the items). However, the problem with this is that some groups might contain only large items and other groups only small items. In this case, the groups with large items will have poor volume guarantees, and we will not get a good approximation ratio.

We therefore have to be more careful. Our algorithm works as follows. All items are classified into types as in the previous section, where the height of the items is (for now) ignored. We then begin by packing the large items. The $(0,0)$ items are stacked in some way, nothing is placed next to this stack.

For the $(1,0)$, $(1,1)$ and $(2,1)$ items, we classify them further along a dimension that has type 1, using the types $1a$, $1b$ and $1c$ that were defined at the start of section 2. Thus we have in total nine subtypes (the $(1,1)$ items are only classified further along their width (smallest size)). We sort the items by subtype, items of subtypes of $(2,1)$ are further sorted by decreasing height.

For each subtype, the items are stacked in the strip so that one corner of each item is directly above a designated corner of the base of the strip, and all items are oriented in the same way. Pairs of $(2,1)$ items are considered as a single $(1,0)$ item in this step, where the width is one of the lengths (largest sizes) of the pair. Thus we have six stacks of items on top of each other: three for the $(1,1)$ items and one each for items of types $(1x,0)$ and $(2,1x)$ for $x = a,b,c$. Here we rotate the $(2,1)$ items such that their length is oriented along the width of the $(1,0)$ items, and a free strip is left next to these items along one side of the main strip.

If we view any one of these stacks from above, it leaves either an L-shaped area or one strip. We can now start using this extra volume for the small items, using the six stacks one by one. The small items are also packed per type. Within each type, the items are sorted in order of decreasing height. Then the items are packed in levels, where each level is packed as in section 3 next to the current stack. The height of a level is the height of the first item packed in it.

Because this stack contains items of only one large subtype (by considering pairs of $(2,1)$ items as $(1,0)$ items), and the levels contain items from one small type, the packing uses the same unique method on all levels that are used for this small type. It is for this reason that we can ignore the single large items in the stack and only care about the height of the stack. If we did not use this distinction into subtypes for the large items, we might need to change the packing method many times, and we would leave much vertical space unused.

We continue creating levels until we run out of items for this small type or the next small item does not fit next to the current stack (its height would be higher than the height of the stack). In this last case, the remaining items of this type are packed next to the next stack of large items. I.e., the next level for this type is not created immediately above the previous level, but instead at the height where the next stack starts. Also, the packing method might be changed at this point.

Finally, if all six stacks are used in this way, or the small items are all packed, we pack the remaining large and small items according to the methods for packing items into empty bins. That is, for each (large or small) type, the items are sorted according to height and then packed in order of decreasing height using the methods from section 3, using as many levels as necessary.

Analysis We begin by making a general remark. Whenever items are packed into levels in order of decreasing height, some height in each level is lost because the first item on the level determines the height of the level, and the next items might have smaller height. However, if we denote the heights of the levels by $H_1, \ldots, H_k$, we have that all items on level i have height at least H_{i+1}. To see how much area is occupied, we can move all items from each level i to level $i+1$. Then level $i+1$ is completely covered by items for each i, and only level 1 is left empty. This implies that when we consider the height of the entire packing for these items, at most a height of H_1 does not contain any items.

We have two cases in our analysis. First of all, it can be seen that if all the small items fit next to the six created stacks, we can ignore the small items in the analysis because they do not add to the total height used. In this case, we can use the weighting technique from section 3.

The weight of an item is now defined as the vertical size divided by the number of times that the 'horizontal item' fits in a square. To give a bound on the performance ratio, we introduce a new concept which is the *weight density*. This is the weight of an item *per unit of vertical dimension*, i.e. it is the weight of the two-dimensional item that we get when we ignore the vertical dimension of an item. We will examine the weight densities at arbitrary horizontal planes through the packings of our algorithm and of the optimal packing.

We find that for each large (sub)type, if it is packed in levels between heights h_1 and h_2, the weight density is 1 at all heights $h \in [h_1, h_2]$ apart from a total height of at most 1. For $(0,0)$ and the subtypes of $(1,1)$ and $(1,0)$, there is even a weight density of 1 at the entire height of their stacks, because all these items are placed directly on top of each other. In the optimal solution, according to Claim 3, there can not be a weight density of more than $9/4$ at any height. Since we use seven types that do not have a weight of 1 everywhere (four medium types and the three subtypes of $(2,1)$), we find that $\text{ALG}_3(L) \leq 9/4 \cdot \text{OPT}(L) + 7$.

Now suppose that some small items need to be packed above the large items. Consider some large (sub)type (one stack) and a single small type t. Suppose all items from this type are placed next to this large subtype, between heights h_t and h'_t. Since the small items are sorted by decreasing height, and the large items are all stacked on top of each other, we have for each height $h_t \leq h \leq h'_t$ an area guarantee of $4/9$ using the proof from section 3, apart from a total height of at most 1.

A small type may also be split among two large stacks, or among one stack and levels of its own (not next to any stack). In this case, some height at the top of the first stack might not contain small items. We can assign this additional height loss to the large (sub)type of that stack. We then find that for each large and small (sub)type, there is a height of at most 1 at which we do not have an area guarantee of $4/9$. In total we have 10 large (sub)types in separate stacks and $21 + 13 + 1 = 35$ small (sub)types and we find $\text{ALG}_3(L) \leq 9/4 \cdot \text{OPT}(L) + 45$.

5 Further applications

5.1 Three-dimensional strip packing

To pack items in a three-dimensional strip, we place each item such that its weight, defined as in the previous section, is minimized. Note that this does not mean simply placing it with its smallest dimension vertical, because the number of times that the implied horizontal item fits in a square might depend on the orientation.

Let w, ℓ, and h be the smallest, second smallest and third smallest dimension of an item. We describe in the table below how the items which we will call large are placed.

Type	Condition	Vertical dimension	Weight	Type (2d)
$(0,0,0),(1,0,0)$		w	w	$(0,0)$
$(1,1,0)^*$		w	w	$(1,0)^*$
$(1,1,1)^*$		w	w	$(1,1)^*$
$(2,0,0)$	$\ell \leq 2w$	ℓ	$\ell/2$	$(2,0)$
	$\ell > 2w$	w	w	$(0,0)$
$(2,1,0)$		ℓ	$\ell/2$	$(2,0)$
$(2,1,1)^*$		ℓ	$\ell/2$	$(2,1)^*$
$(2,2,0)$	$h \leq 2w$	h	$h/4$	$(2,2)$
	$h > 2w$	w	$w/2$	$(2,0)$
$(2,2,1)$		h	$h/4$	$(2,2)$
$(2,2,2)$		w	$w/4$	$(2,2)$
$(3,0,0)$	$\ell \leq 3w$	ℓ	$\ell/3$	$(3,0)$
	$\ell > 3w$	w	w	$(0,0)$
$(4,0,0)$	$\ell \leq 4w$	ℓ	$\ell/4$	$(4,0)$
	$\ell > 4w$	w	w	$(0,0)$

The last column contains the type of the item when the vertical dimension is ignored. For each line of the table, all items are stacked in levels in order of decreasing height. Naturally items that map to the same two-dimensional type can be stacked together in a single stack (e.g. items in lines 1, 5, 13 and 15). All types not mentioned in this table are placed with their *largest* dimension, h, vertically, leaving a small two-dimensional type. Such items are combined with large items (marked with * in the table) exactly as in the This Side Up algorithm from the previous section. That is, for each a and b such that (a, b) is a small type, all types of the form (a, b, x) are combined into a single type. Items are sorted in order of decreasing height and packed into levels next to existing stacks, or at unused heights.

In this problem, the weight of an item is determined by how the algorithm packs an item. As an example, consider a $(0, 0, 0)$ item. If it is placed with h as its vertical dimension, its weight is $h \geq w$, and the weight density at a horizontal plane through this item is $h/w \leq 1$.

As before, we have two cases. Suppose first that all small items can be placed next to the large items. It can be seen that our algorithm has a weight density of 1 at all heights apart from a total height of at most 7, as before. For the optimal packing, we use an unusual definition of the weights and let the weight of any item be the vertical size of this item *as packed in the optimal packing*, divided by the number of times that the 'horizontal item' fits in a square. Denote the sum of the weights of input list L when packed by algorithm $\mathcal{A}$ as $W_{\mathcal{A}}(L)$. Since our algorithm places the items such that their weight is minimized, we have for the total weights that $W_{\text{ALG}_4}(L) \leq W_{\text{OPT}}(L)$.

Denote the average weight density of algorithm $\mathcal{A}$ by $d(\mathcal{A})$. In the optimal packing, the weight density of a single item is at most the weight of the corresponding two-dimensional item in a horizontal plane through the packing. Thus by Claim 3, the total weight density at any horizontal plane is at most $9/4$. Therefore $d(\text{OPT}) \leq 9/4$. We have $\text{ALG}_4(L) \leq W_{\text{ALG}_4}(L) + 7 \leq W_{\text{OPT}}(L) + 7 = \text{OPT}(L)d(\text{OPT}) + 7 \leq 9/4 \cdot \text{OPT}(L) + 7$.

Now suppose that some small items are placed above all large items. In this case, we find as in the analysis of the This Side Up problem that at all heights apart from a total height of at most 45, an area of at least $4/9$ is occupied by items. Since the optimal algorithm must pack the entire volume, the performance ratio of $9/4$ again follows: $\text{ALG}_4(L) \leq 9/4 \cdot \text{OPT}(L) + 45$.

5.2 Three-dimensional bin packing

Finally, we can turn the packing generated by ALG_4 into a packing for the fully rotatable three-dimensional bin packing problem. This is done by making horizontal cuts at all integer heights. Then, for each cut we do the following:

- all unpacked items below this cut are packed into an empty bin
- all items that are intersected by this cut are packed into an empty bin

Since all items have height at most 1, it is clear that this generates a valid packing. Moreover, if we place the bins on top of each other in a stack, it is clear that the height of the highest bin is at most twice the height achieved by ALG_4 plus 1, and the number of bins required to pack L can not be smaller than the optimal height to pack L in a strip.

We conclude
$$\text{ALG}_5(L) \leq 9/2 \cdot \text{OPT}(L) + 91.$$

Although almost all our algorithms are based on ALG_2, this is the only direct reduction where no further modifications to the algorithm are required. More formally, any algorithm for three-dimensional strip packing with rotations that has performance ratio $\mathcal{R}$ can be turned into an algorithm for three-dimensional bin packing with rotations that has performance ratio $2\mathcal{R}$.

The above algorithm and analysis can be also applied to the “This side up” problem in bins.

6 Conclusion

In this paper we design offline algorithms for six packing problems. Most of these problems were not studied in on-line environments, which can be interesting as well. It might be the case that some of the bounds in this paper are possible to improve. Specifically we are interested in improving the constant for packing in three dimensional bins (both for rotatable items and for the “This Side Up” problem). This can be done by designing algorithms for these problems directly instead of adaptation of algorithms for other problems.

References

[1] Nikhil Bansal and Maxim Sviridenko. New approximability and inapproximability results for 2-dimensional packing. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 189–196. ACM/SIAM, 2004.

[2] Alberto Caprara. Packing 2-dimensional bins in harmony. In *Proc. 43th IEEE Symp. on Found. of Comp. Science*, pages 490–499, 2002.

[3] Edward G. Coffman, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826, 1980.

[4] Jose Correa and Claire Kenyon. Approximation schemes for multidimensional packing. In *Proceedings of the 15th ACM/SIAM Symposium on Discrete Algorithms*, pages 179–188. ACM/SIAM, 2004.

[5] Mauro Dell'Amico, Silvano Martello, and Daniele Vigo. A lower bound for the non-oriented two-dimensional bin packing problem. *Discrete Applied Mathematics*, 118:13–24, 2002.

[6] Leah Epstein. Two dimensional packing: the power of rotation. In *Proc. of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'2003)*, pages 398–407, 2003.

[7] Leah Epstein and Rob van Stee. Optimal online bounded space multidimensional packing. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 207–216. ACM/SIAM, 2004.

[8] Satoshi Fujita and Takeshi Hada. Two-dimensional on-line bin packing problem with rotatable items. *Theoretical Computer Science*, 289(2):939–952, 2002.

[9] Keqin Li and Kam-Hoi Cheng. Static job scheduling in partitionable mesh connected systems. *Journal on Parallel and Distributed Computing*, 10:152–159, 1990.

[10] Flavio Keidi Miyazawa and Yoshiko Wakabayashi. Approximation algorithms for the orthogonal z-oriented 3-d packing problem. *SIAM Journal on Computing*, 29(3):1008–1029, 2000.

[11] Flavio Keidi Miyazawa and Yoshiko Wakabayashi. Packing problems with orthogonal rotations. In M. Farach-Colton, editor, *Theoretical Informatics, 6th Latin American Symposium*, number 2976 in Lecture Notes in Computer Science, pages 359–368, 2004.