Two dimensional packing: the power of rotation *

Leah Epstein †

Abstract

Recently there is a rise in the study of two-dimensional packing problems. In such problems the input items are rectangles which need to be assigned into unit squares. However, most of the previous work concentrated on fixed items. Fixed items have a fixed direction and must be assigned so that their bottom is parallel to the bottom of the bin. In this paper we study two-dimensional bin packing of rotatable items. Those are rectangles which can be rotated by ninety degrees. We give almost tight bounds for bounded space bin packing of rotatable items, and introduce a new unbounded space algorithm. This improves the results of Fujita and Hada.

1 Introduction

Consider a situation where large sheets of paper need to be cut into smaller pages. A smaller page is cut off the large sheet so that its side is parallel to a side of the large paper. However, we are not restricted in the direction. We can rotate some of the requests by $90°$ if this makes the assignment of small pages into the large sheet easier. The difference between a page of height h and width w and a page of height w and width h is insignificant. Modeling this situation we get the familiar two-dimensional packing problem of packing rectangles into unit squares. However our problem is slightly different as the requests may be rotated. This problem is called "rotatable items packing", introduced and studied by Fujita and Hada [9]. The same problem is also known as "packing of non-oriented items" [6]. We use bins which are unit squares. The items are rectangles of sides bounded by 1. The items arrive one by one, each rectangle must be assigned to a bin before the next rectangle is introduced. The algorithm has to decide on a position for the rectangle in a previously opened bin, or in a new bin. This decision involves also the decision whether an item is rotated or not. The cost of the algorithm is the number of bins that were used. We study on-line algorithms which are measured

* A preliminary version of this paper appeared in Proc. of 28th Mathematical Foundations of Computer Science (MFCS'2003)

† School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-Mail: lea@idc.ac.il.
Research supported in part by the Israel Science Foundation (grant no. 250/01).

by the competitive ratio. This is the asymptotic worst case ratio between the cost of the on-line algorithm and the cost of an optimal off-line algorithm which sees the input stream as a set of items given in advance. Bounded space algorithms are algorithms which may have only a constant number of active bins. Active bins are bins that can be used to pack new items. Other previously used bins are "closed", which means that they cannot accommodate arriving items.

Our results: We design a bounded space algorithm of competitive ratio at most $2935/1152 + \delta \approx 2.54775$. We show that this algorithm is very close to have the optimal competitive ratio of bounded space algorithms by showing a lower bound of $60377/23814 \approx 2.53536$ on the competitive ratio of every such algorithm. We also design an algorithm of competitive ratio slightly below 2.45 which uses unbounded space. The first algorithm uses many ideas from the paper [9]. To improve the results we use a more advanced analysis, a better partition to types, and a technique which allows us to have a constant number of open bins. This technique is a special case of the technique developed in [7], where it allows to construct algorithms of optimal competitive ratio for non-rotatable rectangles, boxes and hyperboxes. The second algorithm tries to combine together items which, roughly speaking, occupy a large amount of space in any bounded space algorithm.

Note that in our model, not only the assignment to a bin, but also the position of a rectangle inside the bin is decided upon arrival. Our lower bound holds also for the model where only the assignment to the bin has to be done right away, and the exact packing of each bin can be postponed till later.

Previous work: For many years now, there has been extensive work on one-dimensional bin packing [13, 14, 15, 16, 20, 18] and on two-dimensional bin packing of rectangles into unit squares [2, 3, 5, 4, 10, 11, 21, 1, 19]. The best on-line algorithm currently known for packing of rectangles into squares achieves the competitive ratio 2.66013 and was given by [19]. Fujita and Hada [9] introduced the rotatable problem and designed two algorithms of constant competitive ratio. None of the algorithms was defined to be bounded space in their work. The first algorithm does not combine very different items in one bin, and using our techniques can be converted into a bounded space algorithm without losing its properties and analysis. Unfortunately there are two errors in the analysis, one of them can be fixed, but the other is more crucial. Given the current definition of the algorithm and the bounds used to analyze it, the competitive ratio should be slightly higher, namely, instead of at most $47/18 \approx 2.61112$ claimed in the paper it is at most $95/36 \approx 2.63889$. It might be possible to get a better bound on the same algorithm using a more complicated analysis. The second algorithm cannot be converted into a bounded space algorithm. Unfortunately it is not well defined in the paper and its analysis builds on the wrong analysis of the first algorithm. Its competitive ratio is claimed to be at most $100/39 \approx 2.56411$.

As for lower bounds, Seiden and Van Stee [19] give bounds on multi-dimensional square (and cube) packing which clearly imply lower bounds on the packing of rotatable items. We focus on their results for $d = 2$. They show a lower bound of 2.28229 for bounded space algorithms and 1.62176 for unbounded space algorithms. The lower bound for bounded space square packing was recently improved [7] to 2.3552, and the lower bound for unbounded space square packing was improved to 1.64062 [8]

Throughout the paper, we denote a rectangle by (x, y). This means that it has height x and width y. We always assume that the input is rotated in a way such that $x \geq y$. The algorithms may assign a rectangle to a bin in this position or rotated to the other position. In some cases we assume that we can split the bin into parts and rotate some of them. In practice this means that the rectangles are rotated to be placed into this part of the bin.

The paper is organized as follows. We start with a proof of the lower bound. In that section we also prove some useful properties. Then we move to the bounded space algorithm, and in the following section we adapt it to combine different types of large items and show how this affects the competitive ratio.

2 A lower bound for bounded space algorithms

In order to construct a sequence which allows us to prove the lower bound, we define seven types of relatively large rectangles. For each type we check how many instances of this rectangle can simultaneously fit into one bin (with no rectangles of other types). We will use the three following geometrical claims.

Claim 2.1 *Let $\gamma > 0$ be a small constant. Given a packing of squares of width and height $1/k + \gamma$, where $k \geq 2$ is an integer, each bin may have at most $(k-1)^2$ squares packed in it.*

Proof. Any vertical or horizontal line through the bin can meet at most $k - 1$ squares. Take the projection of the squares and the bin on one axis. We get short intervals of length $1/k + \gamma$ (projections of squares) on an main interval of length 1 (the projection of the bin). Each point of the main interval can have the projection of at most $k - 1$ rectangles. Consider the short intervals as an interval graph. The size of the largest clique is at most $k - 1$. Therefore, as interval graphs are perfect [12], we can colour the short intervals using $k-1$ colours. Note that the number of intervals of each independent set is at most $k-1$ (due to length), and so the total number of intervals is at most $(k-1)^2$.
□

Claim 2.2 *Consider a packing of identical rectangles of width* $1/2 < w < 2/3$ *and height* $1/3 < h < 1/2$*, such that* $w + h > 1$*. Each bin may have one or two rectangles.*

Proof. Consider a packed bin. A vertical or horizontal line through the bin can meet at most two rectangles. The only option to meet two is when both are placed rotated to the same direction, the line then goes through an interval of length h in each rectangle. In all other options, the rectangles cannot fit one next to the other. Consider the projection on one axis. There are two types of intervals, long of length w and short of length h. Each point may have at most one short interval or at most two long ones. Since a long interval does not fit together with a short interval, and cannot overlap, we can either have short intervals or long ones. If there are long intervals, all of them must overlap (in the middle point of the main interval). Therefore in this case there are at most two items in the bin. If all intervals are short, they cannot overlap, hence we can have at most two in this case as well. □

Claim 2.3 *Consider a packing of identical rectangles of width* $1/2 < w < 4/7$ *and height* $1/7 < h < 1/6$*, such that* $w + 3h > 1$*. Each bin may have at most eight rectangles.*

Proof. Each rectangle may appear in one of two directions in the bin. The number of rectangles in one position is at most six. The reason is that seven such items cannot fit one next to the other. If there are at most two in the other direction we are done. Therefore it is left to show that if there are at least three rectangles in one direction, there can be at most 4 in the other direction. This will complete the proof due to the following. Let k be the number of rectangles in one direction and ℓ in the other. If $k \geq 3$ then $\ell \leq 4$. If $\ell \geq 3$ then $k \leq 4$. In total we get: If $\ell \leq 2$ then $k \leq 6$ and $\ell + k \leq 8$. If $3 \leq \ell \leq 6$ then $k \leq 4$. If $k \leq 2$ then $\ell + k \leq 8$. Otherwise $k \geq 3$ and so $\ell \leq 4$ and together $\ell + k \leq 8$.

Consider now the rectangles in a certain direction. Let ℓ be their number and assume that $\ell \geq 3$. We consider a line through the middle of the bin, perpendicular to the side of these rectangles which is of length w. Since $w > 1/2$, this line meets all rectangles in this direction. Therefore the line does not meet any rectangles in the other direction (due to size limitations, i.e. $3h + w > 1$). Now split the line into two, and move one line in each directions (left and right) by a distance of $(w - 1/2)$. Both lines still meet the same ℓ rectangles at each position during the movement, and also at the final position. To the right of the right line, and to the left of the left line, it might be still possible to fit at most 2 rectangles placed in the other direction. (The sum of their heights must be at most $1 - w < 3h$.) □

We turn to the specific definition of items. Let $\delta > 0$ be a small constant.

Type A: Rectangles of width and height $1/2+\delta$. By Claim 2.1, a bin can only contain a single such rectangle.

Type B: Rectangles of width and height $1/3+\delta$. By Claim 2.1, a bin can only contain at most four such rectangles.

Type C_1: Rectangles of width $2/3-\delta$ and height of $1/3+2\delta$. By Claim 2.2, a bin can only contain at most two such rectangles.

Type C_2: Rectangles of width $2/3-2\delta$ and height of $1/3+3\delta$. By Claim 2.2, a bin can only contain at most two such rectangles.

Type D_1: Rectangles of width $11/21-4\delta$ and height of $10/63+2\delta$. By Claim 2.3, a bin can only contain at most eight such rectangles.

Type D_2: Rectangles of width $32/63-4\delta$ and height of $31/189+2\delta$. By Claim 2.3, a bin can only contain at most eight such rectangles.

Type E: Rectangles of width and height $1/7+\delta$. By Claim 2.1, a bin can only contain at most 36 such rectangles.

We also use tiny squares of very small height and width. The choice of large rectangles was in a way that one bin (of the optimal off-line algorithm) can contain exactly one of each type. Moreover, there are small gaps left in such a bin. The size of tiny rectangles is picked so that they can fill up all the gaps in a bin containing one of each type of large rectangles.

We show how the large rectangles can fit in a bin containing one of each type. We cut off one horizontal strip from the top of the bin. The height of this strip is $1/3+2\delta$. In this strip we assign one rectangle of type B and one C_1 rectangle. From the part which is left, we cut off the rightmost part, which is a strip of height $2/3-2\delta$ and width $1/3+3\delta$, and assign a rotated C_2 item there. We are left with a bin of height $2/3-2\delta$ and width $2/3-3\delta$. From this we cut off a horizontal strip of height $10/63+2\delta$ (from the top). We use it to pack one E item and one D_1 item. In the remainder of the bin we pack the A item and a rotated D_2 item. Note that $31/189+1/2<2/3$. The sizes of D_1 and D_2 were optimized so that they can fit together and with the E item, and under these conditions they need to have total minimum area. This leaves an area of $V=361/47628-\Theta(\delta)\approx 0.007579575-\Theta(\delta)$. A sequence consists of n rectangles of each type, followed by the tiny squares, i.e. n rectangles of type A followed by n rectangles of type B, then C_1, C_2, D_1, D_2, and E (n items each), and finally tiny squares of total area Vn .

We can now compute the value of the lower bound.

Theorem 2.1 *The competitive ratio of any on-line bounded space algorithm for packing of rotatable items is at least* $60377/23814\approx 2.535356$.

Proof. A bounded space algorithm can keep at most some constant number of bins open. Therefore each type of items is packed separately (except for a small number of items). Items of type A occupy $n - O(1)$ bins. Type B occupies $n/4 - O(1)$ bins. Each of types C_1 and C_2 occupies $n/2 - O(1)$ bins. Each of types D_1 and D_2 occupies $n/8 - O(1)$ bins. Type E occupies $n/36 - O(1)$ bins. The tiny items occupy $An - O(1)$ bins. In total we get $(91/36 + A)n - O(1)$ bins. However an optimal offline algorithm packs all items in n bins. This gives the competitive ratio of at least $91/36 + A \approx 2.53536$. $\Box$

3 A bounded space algorithm

The algorithm classifies items by their size and packs each class separately. To do the classification, we use two parameters ε and M. $M \geq 10$ is a positive integer, and $\varepsilon > 0$ is a small constant. The value of ε is chosen so that $1/(\varepsilon i(i+1))$ is integer for all $i < M$. The role of ε is to control the additive constant of the algorithm. As ε becomes smaller, there are more open bins. On the other hand, we bound the amount of occupied space in each bin. In those calculations, an exact computation would derive values which depend on ε. For simplicity, we assume that ε is small enough and neglect it, this can change the competitive ratio by a very small constant (tending to zero as ε becomes smaller). The value of M also influences the number of classes. A safe choice for M is $M = 20$, and we use this value. We use the following terms in order to analyze the algorithm.

Occupation ratio: For a class (or a subclass) of items, this is the minimum total area of items in a closed bin used to pack items of this class.

Weight: We give each item (assigned to a bin) a weight which is the fraction of a bin which it occupies. This is not the area of the item, but the fraction of the bin that it actually uses (which can be larger than its area). E.g. an item that is a single item in a bin gets weight one. When the algorithm terminates, some bins are active and did not receive a full amount of items as did other bins with similar contents. We ignore these bins, there is only a constant number of them. Some bins will contain a fixed number q of items (that are defined to have similar properties, the number q depends on the items). In this case the weight of each item is $1/q$. For a bin B that contains a variety of items, if the occupation ratio is OR_B, we simply give an item of area r the weight r/OR_B.

Expansion: The expansion of an item is the ratio between its weight and its area. As a first step, we would like to identify all cases where the occupation ratio is low, i.e. items with high expansion.

This type of algorithm can be analyzed using the weighting method introduced already in [15] and further developed in [18]. We use the following theorem.

Theorem 3.1 *Given a bin packing algorithm, define a weight function on items (rectangles with $0 < h \leq w \leq 1$) such that for all bins packed by the algorithm (except a constant number of bins), the sum of weights of items in the bin is at least 1. Consider the (infinite) set of all possible (finite) sets of items such that there exists a feasible packing of them into a single bin. For each such set, define its total weight by the sum of weights of all items. Then the competitive ratio of an algorithm is bounded from above by the supremum of the weights of those sets.*

Next we describe the classification. Each rectangle is classified according to its height and width. For a rectangle (x, y), let i be an integer such that $1/(i+1) < x \leq 1/i$ and let j be an integer such that $1/(j+1) < y \leq 1/j$, clearly $i \leq j$ (as $x \geq y$). If $j < M$, the class of rectangles with given values i and j is called (large) class (i, j). If $j \geq M$ but $i < M$, the class is the medium class i and otherwise we find an integer $p \geq 0$ such that $1/(2M) < x2^p \leq 1/M$. Let i' be an integer $M \leq i' \leq 2M - 1$ so that $1/(i'+1) < x2^p \leq 1/i'$. Then (x, y) belongs to small class i'. We define the packing of each class of each size. For "medium" and "large" classes, we use either "simple packing", or "advanced packing". The specific decision depends on the properties of each class. As a rule, in order to get a low competitive ratio we need to be more careful with packing relatively large items, therefore some classes of large items will be split into sub-classes. The exact definition will be given later. We would like to make sure that the expansion ratio of all small and medium items is at most 1.5. For large items this cannot be true and therefore we need to use a deeper analysis for those items.

We start with defining the simple packing methods for large and medium items. As simple packing of large class items is easiest, and advanced packing of such items is the most complicated, we start and end with large items.

The simple packing of a large class bin is natural. There is always at most one open bin used to pack this class. The bin is partitioned into exactly ij identical rectangles (of height $1/i$ and width $1/j$). The partition is done when the rectangle is opened, by cutting it into i identical rows and j identical columns. Each slot can accommodate exactly one item. After assigning ij items, the bin is closed and a new bin is opened for this class. For a class (i, j), a closed bin has ij items of area at least $1/((i+1)(j+1))$. Therefore the weight of each item is $1/(ij)$, the occupation ratio is $ij/((i+1)(j+1))$, and the expansion of items is $(i+1)(j+1)/(ij)$.

The simple packing of a medium class is done as follows, we use it for $i \geq 3$. For each $3 \leq i < M$, we keep one open bin. This bin is initialized by cutting it into i horizontal strips of height $1/i$ and width 1. The items are packed into the strips in an any-fit fashion. Each strip is seen as a bin of one dimension (the width). Each item is packed immediately to the right of the previous item in the strip. All items in a medium class i

are assigned to such a bin and packed into one of the strips (using any-fit) ignoring their heights. When an item does not fit into any of the strips, the bin is closed and a new bin for medium class i is initialized. In a closed bin, each strip is full by at least $i/(i+1)$ to the height, and by at least $1-1/M$ to the width (an item which did not fit, has width of at most $1/M$). This gives an occupation ratio of at least $(3/4)\cdot(M-1)/M \geq 57/80 \geq 2/3$.

Advanced packing of medium classes is done for $i=1$ and $i=2$. We first consider medium class 1. In this case the height of a rectangle is classified further into $\Theta(1/\varepsilon)$ subclasses. Let α be an integer such that $1 \leq \alpha \leq 1/(2\varepsilon)$. The subclass α consists of rectangles (x,y) such that $x \in (1/2+(\alpha-1)\varepsilon, 1/2+\alpha\varepsilon]$. The bin is split into one part of width 1 and height $1/2+\alpha\varepsilon$, and another part of height $1/2-\alpha\varepsilon$ and width $1/2+\alpha\varepsilon$. (Some space remains unused). After rotation we have two strips (one wide and one narrow) of height $1/2+\alpha\varepsilon$. We use them one-dimensionally applying any-fit. When the bin is closed, each strip is full by at least its width minus $1/M$. This gives total occupied area of at least $(1/2+(\alpha-1)\varepsilon)(1-1/M+1/2-\alpha\varepsilon-1/M)$. The function $x(1-1/M+1-x-1/M)$ is at least $7/10 > 2/3$ (achieved for $x=0.5$) for $M=20$ and $1/2 < x \leq 1$.

For $i=2$, the height is also classified further into $\Theta(1/\varepsilon)$ subclasses. Let α be an integer such that $1 \leq \alpha \leq 1/(6\varepsilon)$. The subclass α consists of rectangles (x,y) such that $x \in (1/3+(\alpha-1)\varepsilon, 1/3+\alpha\varepsilon]$. Given a subclass α, we partition the bin into two parts of width 1 and height $1/3+\alpha\varepsilon$, and two parts of height $1/3-2\alpha\varepsilon$ and width $1/3+\alpha\varepsilon$. Again some space remains unused. After rotation we have two wide and two narrow strips of height $1/3+\alpha\varepsilon$. Those strips are used in an any-fit fashion, in one dimension, to pack items of medium class 2. Similarly to the previous case, we can get the function $x(2+2(1-2x)-4/M)$ for $1/3 < x \leq 1/2$. This gives at least occupied area of $37/45 > 2/3$ (the minimum is obtained for $x=1/3$).

The packing of small items is done as follows. For a small class i the initial partition is the same as for medium classes for $i \geq 3$, but the strips get further partitioned. Packing into strips is again done in one dimension. For each i such that $M \leq i \leq 2M-1$, there is a single open bin dedicated to it. When such a bin is initialized, it is split into i horizontal strips of height $1/i$ and width 1. The items which are assigned to this bin are in class i, i.e. for an item (x,y) there exists an integer $p \geq 0$ such that $1/(i+1) < x2^p \leq 1/i$. The strips will have heights which are of the form $1/(i2^k)$. On arrival of an item there are several cases. Given p, if there is an open strip of height $1/(i2^p)$, and the item fits there, it is simply packed in it. Otherwise (no such strip, or the item does not fit), an empty strip of smallest height $1/(i2^{p'})$ that is still larger than $1/(i2^p)$ (i.e. largest $p' < p$) is picked, and partitioned into horizontal strips of heights $1/(i2^{p'+1}), \ldots, 1/(i2^p)$. An additional strip of height $1/(i2^p)$ is created in this process, and it is used to pack the new item. Finally if this is impossible (no p' exists), the bin is closed and a new one is initialized.

For a bin for small classes, used for a small class i, $M \leq i \leq 2M-1$, we analyze the situation when the bin is closed. Note that an empty strip of some height $1/(i2^k)$, $k \geq 1$, is created if no such empty strip exists. When a strip of height $1/(i2^{p'})$ is partitioned in order to get a strip of height $1/(i2^p)$, there are no empty strips of heights $1/(i2^{p'+1}), \ldots, 1/(i2^p)$. The only height of which two identical strips are created is $1/(i2^p)$, but one of them is immediately used. Hence only one such empty strip exists at a time.

If the bin is closed, clearly no empty strips of height $1/i$ exist any more. Apart for empty strips, there may be some strips that are only slightly used. Using any-fit, an empty strip is used only if an item does not fit into an open strip of the same height. This means that except for one strip, all non-empty strips are occupied with items of total width at least $1-1/M$ (the width of each item is at most $1/M$), we call such strips "busy". If the bin is closed, there are no empty strips of height $1/i$. For each height there may be two empty or almost empty strips, and only one such strip of height $1/i$. This gives a total of at most $1/i + (2/i) \sum\limits_{k \geq 1} 1/2^k \leq (3/i)$. Therefore, a height of at least $1-(3/i)$ contains busy strips. A busy strip is occupied by at least a $i/(i+1)$ fraction of its height, and $1-(1/M)$ of its width. This gives an occupation ratio of at least $(i-3)(M-1)/(M(i+1)) \geq 323/420 > 2/3$.

As mentioned before, for some of the large classes we do not use the simple packing method, but advanced packing. We define and analyze it now.

The choice of those classes is done in a way that the occupation ratio becomes large enough for most items. Specifically, we use advanced packing for the following large classes (we picked $M = 20$ so all the classes listed below are large). 1. $i = 1$ and $2 \leq j \leq M-1 = 19$ 2. $i = 2$ and $3 \leq j \leq M-1 = 19$ 3. $i = 3$ and $4 \leq j \leq 7$.

For items in classes (i,i) $(i < M)$, rotation or further classification does not help, so we pack them using simple packing. Other classes that are not discussed here already have occupation ratio of at least $2/3$.

Each large class (i,j) $(i \leq j)$ that is packed by advanced packing is further partitioned into $\Theta(1/(\varepsilon^2))$ subclasses. Let α, β be integers such that $1 \leq \alpha \leq 1/(\varepsilon i(i+1))$ and $1 \leq \beta \leq 1/(\varepsilon j(j+1))$. The subclass (i,j,α,β) consists of rectangles (x,y) such that $x \in (1/(i+1)+(\alpha-1)\varepsilon, 1/(i+1)+\alpha\varepsilon]$ and $y \in (1/(j+1)+(\beta-1)\varepsilon, 1/(j+1)+\beta\varepsilon]$.

We describe two advanced methods: "side by side" packing and "round packing". The round packing is used only for $(i=1,\, j \leq 10)$, $(i=2,\, j=3,4)$ and $(i=3,\, j=4)$.

We start with the side by side packing. Consider a subclass (i,j,α,β), we describe a partition of a bin into parts which can contain one item each. The size of each part is at least (x_α, y_β) where $x_\alpha = 1/(i+1)+\alpha\varepsilon$ and $y_\beta = 1/(j+1)+\beta\varepsilon$.

We first cut the bin into two horizontal strips of heights ix_α and $1 - ix_\alpha$ and width 1. The first one is further cut vertically into j identical parts, and horizontally into i identical parts. In result we get ij identical parts of width $1/j$ and height x_α. The other part is cut into $\lfloor 1 - ix_\alpha / y_\beta \rfloor$ horizontal strips, each of height at least y_β, and further cut into parts of width $1/i$. This adds $i\lfloor 1 - ix_\alpha/y_\beta \rfloor$ parts which can keep one item each.

To compute the occupation ratio we see that the total area packed in a closed bin is at least $i(x_\alpha - \varepsilon)(y_\beta - \varepsilon)(j + (1 - ix_\alpha)/y_\beta - 1)$ which is close to $ix_\alpha y_\beta(j-1) + ix_\alpha(1 - ix_\alpha)$ for small ε. Given that $y_\beta > 1/(j+1)$ we get at least $ix_\alpha(2j/(j+1) - ix_\alpha)$. This is at least $\min\{(j-1)/(j+1), (i^2 j + 2ij - i^2)/((i+1)^2(j+1))\}$.

We get at least $(j-1)/(j+1)$ for $j \le 2i+1$. Hence for $(i = 2, j = 5)$ and $(i = 3, j = 5, 6, 7)$ we get occupation ratios of at least $2/3$. Otherwise we get at least $(i^2 j + 2ij - i^2)/((i+1)^2(j+1))$ and so for $(i = 1,\ j \ge 11)$ and $(i = 2, j \ge 6)$ we get occupation ratios of at least $2/3$.

Next we describe the "round packing" method. Given sizes s, t such that $s + t \le 1$, we fit two rectangles of width s and height t in the upper left and lower right corners, and two rectangles of width t and height s in the other corners. Since rotation is allowed, we can assume that we thus have four identical areas. Given i, j, α, β we can find positive integers k and r such that $kx_\alpha + ry_\beta \le 1$, we define $s = kx_\alpha$, $t = ry_\beta$ and get four areas such that each area can contain rk items. In total we can pack $4rk$ items of subclass (i, j, α, β). Note that using side by side packing we can pack $i(j+r)$ items when r is the largest integer satisfying $ix_\alpha + ry_\beta \le 1$.

For $i = 3, j = 4$ we can pack 12 or 15 items using side by side packing and 16 items using round packing. The case $2x_\alpha + 2y_\beta > 1$ implies $3x_\alpha + y_\beta > 1$ (as $x_\alpha > y_\beta$) so we need to consider only two options. We get 12 items if $2x_\alpha + 2y_\beta > 1$, and we get 16 items if $2x_\alpha + 2y_\beta \le 1$. In the first case the minimum for $12x_\alpha y_\beta$ (which is the occupation ratio) is $18/25 > 2/3$. If 16 items fit, the occupation ratio is at least $16 \cdot (1/5) \cdot (1/4) = 4/5$.

For $i = 2, j = 3$, six items are assigned by simple packing. If $x_\alpha + 2y_\beta \le 1$, round packing allows to pack eight items. The first option gives occupation ratio of at least $6/9 = 2/3$, the second gives at least $1/3 \cdot (1/4) \cdot 8 = 2/3$ as well. Simple packing may be used if the inequality does not hold, and round packing if it does.

For $i = 2, j = 4$, if $x_\alpha + 2y_\beta > 1$, simple packing assigns eight items. If $2x_\alpha + y_\beta \le 1$ and $x_\alpha + 3y_\beta > 1$, we can assign ten items using side by side packing. If $x_\alpha + 3y_\beta \le 1$ we can pack 12 items using round packing. We get occupation ratios of $16/25$, $20/27$ and $4/5$.

For $i = 1$ we use either a simple packing of j items (if $x_\alpha + y_\beta > 1$) or if $x_\alpha + ry_\beta \le 1$ but $x_\alpha + (r+1)y_\beta > 1$, we can choose between packing $j + r$ items side by side or $4r$ items using round packing. Therefore, if $j \ge 3r$ we prefer side by side packing and otherwise round packing.

For all simple packing cases we get $x_\alpha + y_\beta > 1$ and j items packed. This gives an occupation ratio of at least $j^2/((j+1)^2)$. If $x_\alpha + y_\beta \le 1$ (but $x_\alpha + 2y_\beta > 1$) we use round packing for $j = 2$ only, and for this case get occupation ratio of at least $2/3$. For $j = 3$ we pack four items using round or side by side packing and get occupation ratio of at least $1/2$. In all other cases where $x_\alpha + y_\beta \le 1$ and $x_\alpha + 2y_\beta > 1$ we get an occupation ratio of at least $(j-1)/(j+1)$.

The case $x_\alpha + 2y_\beta \le 1$ and $x_\alpha + 3y_\beta > 1$ is relevant for $j \ge 4$. If $4 \le j \le 5$ we can use round packing and pack eight items. This gives occupation ratio of at least $2/3$ ($x_\alpha > 1/2$ and $y_\beta > 1/6$). We get occupation ratio of at least $(j^2-4)/((j+1)^2)$ for $j \ge 6$. For $j = 6$ this gives $32/49$ and for $j \ge 7$, $(j^2-4)/((j+1)^2) \ge 2/3$

The case $x_\alpha + 3y_\beta \le 1$ and $x_\alpha + 4y_\beta > 1$ is relevant for $j \ge 6$. For $6 \le j \le 8$ we pack 12 items using round packing. The occupation ratio is at least $2/3$. For $9 \le j \le 10$ we pack $j+3$ items using side by side packing, and get occupation ratio of at least $(j^2-9)/((j+1)^2) > 2/3$ for both cases ($j = 9$ and $j = 10$).

The last option $x_\alpha + 4y_\beta \le 1$ is useful only in some cases where $8 \le j \le 10$. In this case, 16 items are packed using round packing. We get an occupation ratio of at least $8/11 \ge 2/3$.

In order to proceed, we summarize all cases where the occupation ratio is smaller than $32/49$. For any other item, the expansion is at most $49/32 = 1.53125$.

1. $i = j = 1, i = j = 2, i = j = 3, i = j = 4$. We got occupation ratios $1/4$, $4/9$, $9/16$ and $16/25$ respectively.
2. $i = 1$ and $j = 2$, we get $4/9$ if $x_\alpha + y_\beta > 1$.
3. $i = 1$ and $j = 3$, we get $9/16$ if $x_\alpha + y_\beta > 1$ and $1/2$ otherwise.
4. $i = 1$ and $j = 4$, we get $16/25$ if $x_\alpha + y_\beta > 1$ and $3/5$ otherwise.
5. $i = 2, j = 4$, the case $x_\alpha + 3y_\beta > 1$ and $2x_\alpha + y_\beta > 1$ gives $16/25$.

We analyze the competitive ratio using Theorem 3.1.

Consider a bin of an off-line algorithm. If it does not contain an item of class (1,1) then all expansion ratios are at most 2.25. Therefore we only need to check cases where there are no class (1,1) items.

We consider several special items.

a. Items of class (1,2) such that $x_\alpha + y_\beta > 1$
b. Items of class (2,2)
c. Items of class (1,3) such that $x_\alpha + y_\beta \le 1$
d. Items of class (3,3)
e. Items of class (1,3) such that $x_\alpha + y_\beta > 1$

The weights of such items are $W(a) = 1/2, W(b) = 1/4, W(c) = 1/4, W(d) = 1/9$

and $W(e) = 1/3$. The areas of such items are at least $2/9, 1/9, 1/8, 1/16$ and $3/16$ respectively. All other items have expansion of at most $5/3$.

The proof of Claim 2.1 implies that a bin can have at most one item of class (1,1), at most nine items of sides larger than $1/4$ and at most twenty-five items with sides larger than $1/6$.

One item of class (1,1) can be seen as four class (3,3) items or nine class (5,5) items. This is true since we can cut it into four identical rectangles by cutting once across each direction, and this may only improve the optimal offline packing. Similarly we can cut into nine identical items. Using similar arguments, an item of type (a) can be seen as two class (3,3) items or six class (5,5) items. A type (b) item is one class (3,3) item or four class (5,5) items. An item of types (c) or (e) is two class (3,3) or three class (5,5) items. Type (d) is a single item of (3,3) or of (5,5).

Therefore, given that the bin contains one item of class (1,1), we may have the following combinations of (a) and (b) items: 1. a, a, b 2. a, a 3. a, b, b 4. a, b 5. a 6. b, b, b, b 7. b, b, b 8. b, b 9. b 10. No (a) or (b) items.

Option 6 is impossible as we would get 5 items of sides larger than $1/3$ (see proof of Claim 2.1).

We analyze the other cases. In case 1, all other items have at least one side smaller than $1/6$. The largest expansion of such items is $49/32 = 1.53125$. The maximum weight of items which fit in a bin is therefore: $1+2W(a)+W(b)+(1-1/4-4/9-1/9)\cdot(49/32) = 2935/1152 \approx 2.547743$.

In case 2 we may have a single item of class (3,3) (or larger) and the other items are smaller (at least in one of the sides). Therefore, there may be an item of type (d) and otherwise all items have expansion of at most $5/3$. This gives weight of $1 + 2W(a) + W(d) + (1 - 1/4 - 4/9 - 1/16) \cdot 5/3 = 1087/432 \approx 2.5162$.

Moving from case 2 to case 3, an (a) item is replaced by two (b) items. There is no change in weight or area so we get the same bound as in case 2.

In case 4 we can have at most one (e) item or at most two (d) items, or one (c) item. Other items have expansion of at most $5/3$. One (e) item gives higher performance than two (d) items (it is larger and has the same expansion) so we have to check two cases.

An (e) item: $1 + W(a) + W(b) + W(e) + (1 - 1/4 - 29 - 1/9 - 3/16) \cdot 5/3 = 355/144 \approx 2.46528$.

A (c) item: $1 + W(a) + W(b) + W(c) + (1 - 1/4 - 2/9 - 1/9 - 1/8) \cdot (5/3) = 179/72 \approx 2.48611$.

In case 5 the best combination would be to add one (d) item and one (c) or (e) item. We replaced a (b) item by a (d) item (compared to case 4) so the weight could not grow.

In case 7, compared to case 4, we replaced one (a) item by two (b) items, so there is no change.

In case 8, we may add one (d) item, compared to case 7. Again, an item of type (b) is replaced by (d) and small items so the weight only decreases. (This is similar to case 5.)

In case 9, 10, the expansion of all items is at most 2 (except the class (1,1) item and possibly a (b) item). This gives $1 + W(b) + (1 - 1/4 - 1/9)2 = 91/36 \approx 2.527778$.

We conclude by the following theorem.

Theorem 3.2 *The competitive ratio of the above algorithm is at most* $2935/1152 + \delta$.

It is unclear whether an algorithm of this type can have the best performance among bounded space algorithms. It seems that a different algorithm could pack some items better. However, as we saw in the lower bound section, the performance cannot be improved by very much.

Note that it is difficult to analyze algorithms for rotatable items using a computer program. For the one-dimensional case, and also for the standard two-dimensional case, there is a fixed set of sizes which is critical. Those are sizes of the structure $1/k + \delta$ (for small $\delta > 0$). However, as we saw in the lower bound section, there are no such rules for rotatable items.

4 An unbounded space algorithm

In the previous section we saw that the drawback of bounded space algorithms is having to pack different types of items separately. As we saw earlier, the items with largest expansion are the relatively small items which still belong to the (1,1) class.

In this section we define an algorithm which combines large but different items together. The idea is to combine not all, but a certain percentage of relatively large items in the same bins. This is similar to the algorithms "Refined Harmonic" [16] and "Modified Harmonic" [17]. Our algorithm packs most items as in the previous algorithm. Some items in the largest three classes: (1,1), (1,2) and (2,2) are packed differently and combined. The analysis uses the following generalized form of Theorem 3.1 and gives a competitive ratio slightly below 2.45. The generalization is still a special case of [18].

Theorem 4.1 *Given a bin packing algorithm, define two weight functions on items (rectangles with* $0 < h \le w \le 1$*) such the following property holds for at least one of the weight functions. For all bins packed by the algorithm (except a constant number of bins that is defined by the algorithm), the sum of weights of items in the bin is at least* 1 *on average. Consider the (infinite) set of all possible (finite) sets of items such that there exists a feasible packing of them into a single bin. For each such set, define its total weight according to a given weight function by the sum of weights of all items, and it* *maximum weight to be the maximum of the two total weights calculated using the two weight functions. Then the competitive ratio of an algorithm is bounded from above by the supremum of the maximum weights of those sets.*

We use a parameter $D = 0.61615$. The goal is to reach a competitive ratio below 2.45. It is possible to decrease the ratio very slightly using more accurate parameters and a detailed case analysis on the exact sizes of items.

Items of class (1,1) which are larger than D in both dimensions are packed one per bin as before. Other items of this class are packed (if possible) together with one item of the class (1,2) or two items of the class (2,2). In this algorithm, there can be bins which contain one item of class (1,1) that are waiting for items of other classes to arrive and join the bin, or waiting bins that received an item of class (1,2) or two items of class (2,2), that still did not receive an item of class (1,1). In both of the last cases, a waiting bin of one of those classes ((1,2) or (2,2)), needs to contain an empty part of width 1 and height D, where the item of class (1,1) should fit. This gives further restrictions on which class (1,2) and (2,2) items may be packed in the waiting bins. The method of packing for an item of class (1,1) with $h \leq D$ is as follows. If there is a bin waiting for such an item, it is packed there. Otherwise, we open a new bin. We partition it into two horizontal strips of heights D and $1 - D$ (and width 1). The class (1,1) item is packed in the first and the bin will be waiting for other items.

For class (1,2), we define the new method of packing only for items that would have been packed in pairs, and such that $h \leq 1 - D$. (Other items are packed exactly as before.) Each such item gets an index i according to the order of arrival. Let $i' = i \bmod 23$ (the very first item gets the index $i = 0$). The packing of every subsequence of 23 items is similar.

The first 22 items are packed using advanced packing (i.e. they are further classified, as in the bounded space algorithm). The 23$^{\text{rd}}$ item is supposed to be combined with a class (1,1) item. If there is a waiting bin, the class (1,2) item is packed there into the strip of height $1 - D$. Otherwise a new bin is opened, partitioned into two parts as explained previously for class (1,1) and after the assignment of the 23$^{\text{rd}}$ item it is waiting for a class (1,1) item.

For class (2,2), we define a new method of packing only for items for which $h \leq 1-D$. Since for those items $w \leq 1/2$, we can fit two of them in a strip of height $1 - D$ and width 1. Other items of this class are packed four in each bin as before.

We use a similar indexing system for these items. We pack every subsequence of 36 items in the same way. Items 0 up to 27 are packed in seven bins. Items 28 up to 35 are packed in four bins - either waiting bins, or by creating bins which will wait for class (1,1) items.

Clearly, there are two options (neglecting at most a constant number of bins). The waiting bins either contain only items of class (1,1) or only items of classes (1,2) and (2,2).

If the algorithm terminates with waiting bins of class (1,1), then items of class (1,1) get weight 1, whereas other items need to pay for the bins they use, but not for the bins that also contain class (1,1) items. Consider sets of 23 items of class (1,2). Such a set occupies only eleven bins and each item gets a weight of $11/23$ (this concerns only items with $h \leq 1-D$ that should have been packed in pairs). Every 36 class (2,2) items occupy seven bins and get weight $7/36$ (again only items for which $h \leq 1-D$). Otherwise, items of class (1,1) with $h \leq D$ get weight zero. In this case the items of classes (1,2) and (2,2) which are combined with them need to pay for all the bins they occupy. Items with $h \leq 1-D$ of class (1,2) that should have been of weight $1/2$ (in the bounded space algorithm) now get weight $12/23$ (every 23 occupy 12 bins), and class (2,2) items with $h \leq 1-D$ get weight $11/36$ as every 36 items occupy eleven bins.

To do the calculation of maximum weight in a bin we separate the discussion into three cases. We call the option where there are class (1,1) waiting bins "small weights" and the other option "high weights".

Case 1: The bin has no items of class (1,1). In this case clearly the "high weights" option is worse. All items except classes (1,2) and (2,2) have expansion 2 or less. We compare each possible situation with the situation where all items are of weight 2. We may have at most four items of those classes. We get the weight $W + (1-A)2$, where W is the total weight of at most four items and A is their area. Let V denote a weight of a rectangle. There are five possible types of items.

1. Class (1,2) weight $1/4$. The area of such items is at least $1/6$. The contribution to $W - 2A$ is negative.

2. Class (1,2) weight $1/2$. In this case $h > 1-D$ so $V \geq D(1-D)$. The contribution to $W - 2A$ is at most 0.027.

3. Class (1,2) weight $12/23$. $V \geq 2/9$ so we get contribution of at most 0.0773.

4. Class (2,2) weight $1/4$. In this case $h > 1-D$ so $V \geq (1-D)^2$. The contribution to $W - 2A$ is negative.

5. Class (2,2) weight $11/36$. $V \geq 1/9$ so $W - 2A$ increases by at most $1/12 \approx 0.083333$.

Therefore $W + 2(1-A) \leq 2 + 4 \cdot 1/12 = 7/3 < 2.45$.

Case 2: "High weights" with one class (1,1) item. If the weight of the class (1,1) item is zero, we reduce to an easier situation than case 1. We need to consider the case where the class (1,1) item has area of at least D^2.

We need to consider cases similarly to the bounded space algorithm. Note that for classes (2,2) and (1,2) it always gives a worse situation if the items are relatively small and have a large weight, rather than larger items which were not combined and have their original weight. Therefore assume that an (a) item has weight 12/23 and a (b) item has weight 11/36.

We recompute the nine cases, note that the area of the class (1,1) item is at least $(0.61615)^2$.

1. $1 + 2W(a) + W(b) + (1 - (0.61615)^2 - 4/9 - 1/9) \cdot (49/32) = \leq 2.44827$.

2. $1 + 2W(a) + W(d) + (1 - (0.61615)^2 - 4/9 - 1/16) \cdot (5/3) \leq 2.34362$.

3. $1 + W(a) + 2W(b) + W(d) + (1 - (0.61615)^2 - 4/9 - 1/16) \cdot (5/3) \leq 2.433$.

4. $1 + W(a) + W(b) + W(e) + (1 - (0.61615)^2 - 2/9 - 1/9 - 3/16) \cdot (5/3) \leq 2.3266$.

   $1 + W(a) + W(b) + W(c) + (1 - (0.61615)^2 - 2/9 - 1/9 - 1/8) \cdot (5/3) \leq 2.3474$.

5. The same reasoning as before works here as well.

7. Comparing to case 4, the difference $2W(b) - W(a) \leq 0.089372$ hence the competitive ratio is at most 2.4367.

8. The same reasoning as before works here as well.

9, 10. $1 + W(b) + (1 - (0.61615)^2 - 1/9) \cdot 2 \leq 2.3241$.

Case 3: "Low weights" with one class (1,1) item of area at least 1/4.

In many of the formulas we get the terms $W(a) - \mu A(a)$, $\mu = 49/32, 25/16, 5/3, 16/9, 2$ where $A(a)$ is the minimum area of an "a" item, and similarly we get expressions involving "b" items.

Note that there are two types of (a) items: of weight 11/23 and area at least 2/9, and of weight 1/2 and area at least $D(1 - D)$. The value of all expressions is always larger for the first type of (a) items. There are also two types of (b) items: of weight

$7/36$ and area at least $1/9$, and of weight $1/4$ and area at least $(1-D)^2$.

The only expression that is very slightly larger for the second type is $W(b)-(49/32)A(b)$.

We again recompute the nine cases.

1. $1+2W(a)+W(b)+(1-1/4-2A(a)-A(b))\cdot(49/32) \leq$
   $1+2(W(a)-(49/32)A(a))+(W(b)-(49/32)A(b))+0.75\cdot(49/32) \leq 2.44879.$

2. In this case we may have up to four extra items with $h > 1/6$. To fit all in a unit bin, each extra item may be of the size of one, two or four items such that $h > 1/6$ (due to the shape of class (1,1) items which is a mesh of 3 columns and three rows of items with $h > 1/6$, and (1,2) items which are 3 rows and 2 columns or 2 rows and 3 columns).

There is room for only at most one class (3,3) item. Apart from that, there may occur class (4,4) items and class (2,4) items. All other items have expansion of at most 49/32. Together with possible class (4,4) or (2,4) items, all items have expansion of at most 1.5625. We need to take into account only an item of type "d".
$1+2W(a)+W(d)+1.5625(1-1/4-2A(a)-A(d)) \leq$
$2.171875+2(W(a)-1.5625A(a))+(W(d)-1.5625A(d)) \leq 2.44741$

3. In this case again there can be just one extra item with $h > 1/4$, hence all expansions are at most 16/9.
$1+W(a)+2W(b)+(16/9)(1-1/4-A(a)-2A(b)) \leq 2.4104$

4. In this case the only item with expansion larger than 16/9 may be a (c) item.
$1+W(a)+W(b)+W(c)+(16/9)(1-1/4-A(a)-A(b)-1/8) \leq 2.44123$

5. Again there can be one (c) item and all others have expansion of at most 16/9.
$1+W(a)+W(c)+(16/9)(1-1/4-A(a)-1/8) \leq 2.44432$

7, 8, 9, 10. The expansion of (b) items is at most 1.75, therefore the worst case is no (b) items, two (c) items, and all other area with expansion 16/9.
$1+2W(c)+(16/9)(1-1/4-2/8) \leq 43/18 \approx 2.38889.$

We can conclude with the following Theorem.

Theorem 4.2 *The above algorithm has competitive ratio strictly below 2.45.*

5 Conclusion

We showed an almost optimal algorithm (in terms of competitive ratio) for bounded space packing of rotatable items. It is interesting to find an algorithm which can be shown to be optimal. It could be the case that our algorithm, or a simple adaptation, could be used for this purpose. Another interesting issue is simplifying the analysis of unbounded space algorithms (i.e. reducing the problem to a given set of possible items as in [18]) in a way that a computer program will be able to perform the analysis. This will allow a deeper analysis which can take smaller items into account, and should result in smaller competitive ratios.

A natural extension of the problem is packing rotatable boxes in three dimensional cubes, and packing rotatable d-dimensional hyper-boxes in d-dimensional hyper-cubes. Most methods used in this paper can be applied for more dimensions as well, however there may be many cases to check and it seems that a good analysis can be done once a method to computerize such calculations is developed.

Acknowledgement

The author would like to thank the MFCS referee who gave many helpful comments.

References

[1] D. Blitz, A. van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.

[2] F. R. K. Chung, M. R. Garey, and D. S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic and Discrete Methods*, 3:66–76, 1982.

[3] D. Coppersmith and P. Raghavan. Multidimensional online bin packing: Algorithms and worst case analysis. *Operations Research Letters*, 8:17–20, 1989.

[4] J. Csirik and A. van Vliet. An on-line algorithm for multidimensional bin packing. *Operations Research Letters*, 13(3):149–158, Apr 1993.

[5] J. Csirik, J. B. G. Frenk, and M. Labbe. Two dimensional rectangle packing: on line methods and results. *Discrete Applied Mathematics*, 45:197–204, 1993.

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

[7] L. Epstein and R. van Stee. Optimal online bounded space multidimensional packing. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, 2004. to appear.

[8] L. Epstein and R. van Stee. 2003.

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

[10] G. Galambos. A 1.6 lower bound for the two-dimensional online rectangle bin packing. *Acta Cybernetica*, 10:21–24, 1991.

[11] G. Galambos and A. van Vliet. Lower bounds for 1-, 2-, and 3-dimensional online bin packing algorithms. *Computing*, 52:281–297, 1994.

[12] T. R. Jensen and B. Toft. *Graph coloring problems.* Wiley, 1995.

[13] D. S. Johnson. *Near-optimal bin packing algorithms.* PhD thesis, MIT, Cambridge, MA, 1973.

[14] D. S. Johnson. Fast algorithms for bin packing. *Journal Computer System Sciences*, 8:272–314, 1974.

[15] D. S. Johnson, A. Demers, J. D. Ullman, Michael R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.

[16] C. C. Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32:562–572, 1985.

[17] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. Online bin packing in linear time. *Journal of Algorithms*, 10:305–326, 1989.

[18] S. S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

[19] S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36(3):261–293, 2003. In a special issue on on-line algorithms. Also in Proc. SODA'2002, pp. 486–495.

[20] A. van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43:277–284, 1992.

[21] A. van Vliet. *Lower and upper bounds for online bin packing and scheduling heuristics.* PhD thesis, Erasmus University, Rotterdam, The Netherlands, 1995.