Journal of Discrete Algorithms 8 (2010) 356–372

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Dynamic multi-dimensional bin packing

Leah Epstein [a,*], Meital Levy [b]

[a] Department of Mathematics, University of Haifa, 31905 Haifa, Israel
[b] School of Computer Science, Tel-Aviv University, Israel

ARTICLE INFO

Article history:
Received 7 February 2009
Accepted 24 June 2010
Available online 17 July 2010

Keywords:
Bin packing
Online algorithms
Dynamic algorithms
Multi-dimensional packing

ABSTRACT

A natural generalization of the classical online bin packing problem is the *dynamic bin packing problem* introduced by Coffman et al. (1983) [7]. In this formulation, items arrive and depart and the objective is to minimize the maximal number of bins ever used over all times. We study the oriented multi-dimensional dynamic bin packing problem for two dimensions, three dimensions and multiple dimensions. Specifically, we consider dynamic packing of squares and rectangles into unit squares and dynamic packing of three-dimensional cubes and boxes into unit cubes. We also study dynamic d-dimensional hypercube and hyperbox packing. For dynamic d-dimensional box packing we define and analyze the algorithm NFDH for the offline problem and present a dynamic version. This algorithm was studied before for rectangle packing and for square packing and was generalized only for multi-dimensional cubes. We present upper and lower bounds for each of these cases.

1. Introduction

Bin packing is one of the oldest and most thoroughly studied problems in computer science. In the classical one dimension bin packing problem, the study of which dates back to the works of Johnson and Ullman in the early 1970's [15,23] (also see [16]), there is a sequence of items, each with size in the range $(0, 1]$. The goal is to pack all the requested items into a minimum number of unit bins.

A natural generalization of the classical problem is the *dynamic bin packing problem* introduced in [7], where items arrive and depart in an online fashion and the goal is to minimize the maximal number of bins ever used over all times. As mentioned in [7], in certain potential applications, such as storage allocation, the simple model disregards departures of items. Therefore, a more realistic setting is needed. The dynamic formulation also seems much more suitable for maintaining a vast physical storage of bins. Items of the shape of rectangles or squares may be purchased (arrive) and sold (depart).

The problem we address through our model is that of how to distribute items among storage units so that at all times each unit will have sufficient space to hold all the items assigned to it. The one-dimensional version [7] assumes that at a departure of an item, the remaining items of its bin are shifted towards the bottom, to make room for future items. Similarly to that, we assume that re-arranging the bin is allowed at any time and thus allowed upon arrival of items. Hence we do not consider the related problem of how one manages space within a storage unit. Our approach focuses on the allocation of items to units. These units may be geographically far from each other and therefore migration between units cannot be allowed. However, local changes in the exact packing of items do not affect other units and therefore are not forbidden. The related problem, where positions inside bins are fixed, seems to be much more difficult. To stress this, we give the following example. Assume a very large number of tiny items arrive and are packed into two-dimensional boxes. If the boxes are fully

* Corresponding author.
E-mail addresses: lea@math.haifa.ac.il (L. Epstein), levymeit@gmail.com (M. Levy).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2010.07.002

or almost fully packed, it is possible to let most of the items depart and yet to block these bins completely if future items are even only slightly larger. However, if bins are packed very sparsely, already the initial packing of the small items is not successful.

We consider the dynamic bin packing problem where the goal is to pack items which are multi-dimensional cubes or boxes (rectangles or squares, in the case of two dimensions) into bins which are unit cubes (or squares, for two dimensions). The items are oriented and cannot be rotated (in the case of rectangles or boxes), and need to be assigned in a non-overlapping way into the bins. This problem is an online problem, since future input is unknown. Unlike the classical online bin packing problem, an event here can be either an arrival or a departure of an item. We consider the problem where items are assigned to bins irrevocably, but their location in the bins may be modified. Note that in the classical multi-dimensional online problem once an item placed insides a bin, it's position can no longer be altered.

All the known lower bounds for the classical non-dynamic online problem (in one or more dimensions) do not assume that the position of items inside the bins must not change and therefore they are valid for our model too.

We study the oriented online dynamic d-dimensional bin packing problem in terms of competitive analysis. The standard measure of algorithm quality for box packing is the *asymptotic competitive ratio*, which we now define. For a given input sequence σ, let $\mathcal{A}(\sigma)$ (or $\mathcal{A}$, if the sequence is clear from the context) be the number of bins used by algorithm $\mathcal{A}$ on σ, that is, the maximal number of bins ever used at the same time. Let $OPT(\sigma)$ (or OPT) be the maximal number of bins used throughout the sequence σ by an optimal offline algorithm OPT which knows the sequence of arrivals and departures in advance. The *asymptotic competitive ratio* is defined to be $\limsup_{n\to\infty} \sup_{\sigma}\{\frac{\mathcal{A}(\sigma)}{OPT(\sigma)} \mid OPT(\sigma) = n\}$, for an algorithm $\mathcal{A}$. We say that an algorithm is $\mathcal{R}$-competitive, if its asymptotic competitive ratio is at most $\mathcal{R}$.

1.1. Previous work

Online one-dimensional dynamic bin packing

This problem was introduced by Coffman et al. [7]. It was shown in their paper that the online algorithm First Fit has an asymptotic competitive ratio between 2.75 and 2.897. They also showed that no online algorithm can achieve an asymptotic competitive ratio smaller than 2.5 if the optimal offline algorithm may re-pack the items and 2.388 if re-packing is not allowed. A stronger results, which is a lower bound of 2.5 for the case where the optimal offline algorithm cannot re-pack the items, was recently shown in [6].

Ivkovic and Lloyd [20] studied a more general problem called *the fully dynamic bin packing problem*, where migration of items is permitted, and gave a 1.25-competitive online algorithm for this version.

In [5] the dynamic formulation was considered for a restricted class of unit fraction items. Unit fraction items are items with sizes $\frac{1}{k}$ for some integer $k \geqslant 1$. They studied the family of any fit algorithms including First Fit for which they proved an asymptotic competitive ratio in the range of 2.45 and 2.4985 for unit fraction items. They also provided a general lower bound of 2.428 on the asymptotic competitive ratio.

Multi-dimensional bin packing

Oriented packing problems have been widely studied. The best result for two-dimensional offline packing into bins is an algorithm of an asymptotic approximation ratio 1.525, due to Bansal, Caprara and Sviridenko [1]. Caprara [4] designed algorithms for any $d \geqslant 3$ with an asymptotic approximation ratio of at most 1.691^{d-1}. An APTAS for square and cube packing (of any dimension) was given in [2]. See papers [2,4,11] and references therein for further results on offline oriented packing problems.

The online problem of non-dynamic multi-dimensional packing was studied in several papers, see e.g. [9,22,21]. The current best results for two and three dimensions are as follows. For square packing, an upper bound of 2.1439, and for three-dimensional cube packing, an upper bound of 2.6852, both by Han et al. [14]. In [12], lower bounds for unbounded space hypercube packing were given. The known lower bounds are as follows. For $d = 2$, it is 1.6406, while for $d = 3$ it is 1.6680. In fact, this paper provides lower bounds for the dimensions $4, \ldots, 10$, where the lower bounds are between 1.67 and 1.7. The best online result for rectangle packing is by Seiden and van Stee [22] which is an upper bound of 2.66013. The best lower bound known for the problem is 1.907, by [3]. The general problem of online packing of d-dimensional boxes (for $d \geqslant 3$) was studied by Csirik and van Vliet [10] (see also [19] for $d = 3$). Their algorithm has an asymptotic competitive ratio of 1.691^d. This is also the asymptotic competitive ratio of the bounded space algorithm of [13], for rectangles and boxes. Bounded space algorithms are such that can only keep a constant number of bins "open", i.e., ready to receive items. A bin which is declared "closed", i.e., cannot receive new items, is never opened again. Such algorithms cannot be used for dynamic bin packing.

1.2. Our results

We first consider the dynamic bin packing problem in two dimensions. Namely, we consider dynamic packing of squares and of rectangles. We give a 4.2154-competitive algorithm for dynamic packing of squares, for which we provide a lower bound of 2.2307 on the asymptotic competitive ratio. We give a 8.5754-competitive algorithm for rectangles and a lower bound slightly higher than 3.7. Next, we also study the three-dimensional case which can be found in Appendix B. For

three-dimensional cubes we present an algorithm which is 5.37037-competitive, and a lower bound of 2.117. For three-dimensional boxes, we supply a 35.346-competitive algorithm and a lower bound of 4.85383.

We further consider multi-dimensional packing. We define and analyze the algorithm NFDH for the offline box packing problem. This algorithm was studied before for rectangle packing (two-dimensional only) [8], and for square and cube packing for any dimension [20,17], but not for box packing. One of the main results is the generalization of NFDH for packing multi-dimensional boxes into unit multi-dimensional cubes for the offline version as well as the dynamic version. This algorithm requires a more clever approach than the version given for cube packing. Sorting cubes by decreasing order according to one dimension implies a decreasing order of every dimension. However, for general boxes, a decreasing order of one dimension is not necessarily a decreasing order in other dimensions. Therefore the simple recursive scheme used for cubes does not work.

For d-dimensional boxes we provide an algorithm which implies an upper bound of $2 \cdot 3.5^d$ and a lower bound of $d+1$. Note that the best bound known for the regular offline multi-dimensional box packing problem is exponential as well. For d-dimensional cubes we provide an upper bound of $O(\frac{d}{\ln d})$ which is below the lower bound of the d-dimensional box packing. We prove a lower bound of 2 for this case.

2. Preliminaries

The model is a natural extension of the classical static two-dimensional bin packing model. The items to be packed will be described by a finite sequence $L = (p_1, p_2, \ldots, p_n)$. Each item $p_i \in L$ corresponds to a vector (a_i, d_i, x_i, y_i), where a_i is the arrival time for p_i, d_i is its departure time, $x_i \leqslant 1$ is its width and $y_i \leqslant 1$ is its height. The volume (or area) of this item is $x_i \cdot y_i$, and for squares we have $x_i = y_i$. For d dimensions there is a vector of $d+2$ dimensions $(a_i, d_i, x_{i_1}, \ldots, x_{i_n})$. Both a_i, d_i, play the same role as before and x_{i_j} denotes the j-dimensional value of item p_i. The volume of the item is $\prod_{j=1}^{d} x_{i_j}$. The item p_i resides in the bin for the time interval $[a_i, d_i)$ and L is ordered so that $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_n$. Note that at the time of arrival a_i, the value d_i is unknown to the online algorithm.

We consider the case where migration of an item between bins is not allowed. However upon arrival of an item, the position of items inside the bins can be altered.

In the classical bin packing problem, partitioning the input sequence into sub-classes and packing each one separately can be useful (see e.g. [18]). In this type of analysis, there is a loss of at most one bin for each class if this bin is not filled by its planned number of items, but the analysis for bins of different classes that are packed successfully can be usually done together. Since in our model items may depart, the asymptotic competitive ratio of an algorithm that partitions the input into classes and packs each class separately, is the sum of the asymptotic competitive ratios of all the classes. There may be a point in time where for every bin of a certain calls, a very small portion of the bin is utilized due to departure of items. The unused part of the bin is not used again if future items are of different class. Nevertheless, the asymptotic competitive ratio is no larger than the sum of the asymptotic competitive ratios of the composed sub-algorithms as we can see below.

Suppose that we use a constant number of classes, k, and for each class we apply an online sub-algorithm A_i with an asymptotic competitive ratio of at most R_i for every $1 \leqslant i \leqslant k$. The input sequence L is split into i disjoint sub-sequences, where sequence i consists of items that belong to class L_i. Since $A_i(L_i) \leqslant R_i \cdot OPT(L_i) + o(OPT(L_i))$ (by the definition of the asymptotic competitive ratio) and $OPT(L_i) \leqslant OPT(L)$ (since L_i is a sub-sequence of L) we get the following: $A(L) \leqslant \sum_{i=1}^{k} A_i(L_i) \leqslant \sum_{i=1}^{k} (R_i \cdot OPT(L_i) + o(OPT(L_i))) \leqslant \sum_{i=1}^{k} R_i \cdot OPT(L) + o(OPT(L))$. Getting an asymptotic competitive ratio of at most $\sum_{i=1}^{k} R_i$. Therefore one should find the tradeoff between the advantage of packing similar items as an independent class and the asymptotic competitive ratio of that class that is added to the total asymptotic competitive ratio.

In our algorithms we use a variant of a level algorithm called NFDH as a sub-routine. This algorithm has been used in an offline environment. We make adjustments to the algorithm to suit the dynamic setting. We call the modified algorithm Dynamic-NFDH, or D-NFDH, which is short for Dynamic Next Fit Decreasing Height.

3. NFDH and D-NFDH

In two dimensions, the offline algorithm NFDH is a level (shelf) algorithm which uses the next-fit approach to pack the sorted (by non-increasing height) list of rectangles. A level algorithm creates levels by drawing horizontal lines of length 1, and placing rectangles between consecutive pairs of lines, including the bottom and top of the bin. The two lines which form a level are called the "floor" and "ceiling" of the level or the "bottom" and "top" of the level. The additional lines are typically created online starting from the bottom, where the "floor" of the first level is the bottom. The rectangles are packed, left-justified on the floor of a level until the next rectangle does not fit. This rectangle is used to define the height of a new level and the packing continues on this level. A new level is defined just above the previous level. If a new level of the given height cannot be opened, a new bin is opened and the rectangle defines the first level of that bin. As opposed to the algorithm First Fit Decreasing Height (FFDH), earlier levels are not revisited. The levels are numbered from bottom to top, according to the order they are created. In this section we use $\frac{1}{k}$ as an upper bound on the side of items. We let $k \geqslant 2$.

The next theorem holds for rectangles, and consequently for squares as well, however for squares or cubes, the same utilization of the volume in the bin can be proved for any dimension (this property will be proved later). The proof of the following theorem is similar to proofs in [8].

Theorem 1. *Let L be a list of rectangles p_i with sizes of sides (h_i, w_i) where $h_i, w_i \leqslant \frac{1}{k}$ for all i. Let P be the packing into unit square bins obtained by applying the algorithm NFDH to the list L. Then each set of rectangles packed in a bin used by P (except for possibly one last bin) has volume greater than $(1-\frac{1}{k})^2$.*

Proof. Suppose the i-th rectangle in the decreasing order of height does not fit into bin k and has to be packed by NFDH into the next bin. We virtually rearrange the rectangles of bin k to show that the total volume of rectangles already packed in the bin is at least $(1-\frac{1}{k})^2$. Move the contents of each shelf to the one on top of it (including the top non-empty shelf into the empty one on top, that may have height zero). Now each rectangle covers the height of the shelf it was moved to, since every rectangle in shelf j has height greater or equal to shelf $j+1$, whose height was defined by a rectangle of smaller height. All shelves but the bottom shelf (whose height is at most $\frac{1}{k}$) are covered with respect to their height and full by a fraction of at least $1-\frac{1}{k}$ of their width, since the width is also bounded by $\frac{1}{k}$. Therefore we get a covered volume of $(1-\frac{1}{k})^2$. □

3.1. A modified version of NFDH for the two-dimensional dynamic problem

We are now ready to describe the dynamic algorithm for two dimensions, 2-Dynamic-NFDH. Upon arrival of item p_i, the algorithm packs item p_i in the bin of smallest index in which it is possible. The index of a bin is determined by the time stamp the bin was opened. The algorithm does in fact re-position items inside the bins. In order to determine if the new item p_i can fit into a specific bin, p_i is merged to the sorted list of the bin, the items of the bin are packed again using the sub-routine NFDH for two dimensions to the same bin. If the new packing only uses one bin, it means that p_i can be assigned to that bin. Otherwise the packing of the bin returns to its original packing before time a_i and p_i attempts to fit the next bin. If the item cannot fit any of the used bins, a new bin is opened and p_i is placed there.

Lemma 1. *Let L be an input list of items to the dynamic bin packing problem. If L consists of rectangles with maximum width and height of $\frac{1}{k}$, then the asymptotic competitive ratio of 2-Dynamic-NFDH is at most $\frac{1}{(1-\frac{1}{k})^2}$.*

Proof. Recall that in the dynamic model the items arrive online. Therefore the next item that the algorithm tried to add to an existing bin may have larger height than the height of items already packed in the bin.

We first show that if a new item p_i does not fit in some bin m, the items already in the bin have a total volume of at least $(1-\frac{1}{k})^2$. Denote by L_m the set of items packed into bin m. Denote by P_m the packing of L_m in bin m and by P'_m the packing of the list $L'_m = L_m \cup \{p_i\}$ using NFDH. Since the algorithm reaches the conclusion that p_i cannot be added to bin m, the resulting packing consists of more than one bin.

For the analysis we use the following observations regarding P'_m.

1. The last item of L'_m cannot be placed in the bin. Specifically, at least one item does not fit, and actually there may be an arbitrary number of items that do not fit as a result of inserting p_i.
2. The items of every level in the first bin in the packing P'_m have total width of at least $1-\frac{1}{k}$.
3. Let $j \geqslant 1$. Adding the first item of level $j+1$ to level j, where j is any level excluding the last non-empty level and the last empty level, to the packing of the first bin of P'_m, we get that the total width of items in this level exceeds 1. This is true since otherwise the item from level $j+1$, that is added, would be packed in level j. For the last non-empty level, adding the first item of the next bin results in a total width of items which exceeds 1 due to the same reasoning.

If the new item p_i is the last item in the sorted list L'_m, or any item which is not packed into the first bin of P'_m, then by a similar argument of the proof of Theorem 1, if every level has a total width of at least $1-\frac{1}{k}$, we get a total volume of $(1-\frac{1}{k})^2$ of items only in L_m.

If p_i is positioned in some level ℓ in the first bin of the packing of P'_m, we perform the following virtual repositioning of items. Remove p_i from level ℓ. If the total width of level ℓ is more than $1-\frac{1}{k}$ after the removal, then by Observation 2, every level has a total width of items of more than $1-\frac{1}{k}$, and therefore, similarly to the proof of Theorem 1, the total volume of L_m is above $(1-\frac{1}{k})^2$.

Otherwise, remove the first item of the next level $\ell+1$, p_j, and position it in level ℓ. By Observation 3, level ℓ with both p_i and p_j has a total width of above 1, therefore since the width of p_i is at most $\frac{1}{k}$ we get a total width of above $1-\frac{1}{k}$ in level ℓ. Now in level $\ell+1$, we may have less than a total width of $1-\frac{1}{k}$. In that case proceed in a similar way, i.e., remove the first item of level $\ell+2$ and position it in level $\ell+1$. We keep re-positioning items in the same way until the last level is reached, or the items of some level exceed a total width of $1-\frac{1}{k}$ without adding the first item of the next level. If we reach the last level we add the first item in the second bin. By Observation 1, such an item must exist.

By the end of the re-positioning process, we have in every level a total width of at least $1-\frac{1}{k}$. Although we did not lower the height of the levels after re-positioning, every item in every level has a height of at least the height of the next level. This is true since we only move a single item from the next level (in the same bin or the second bin), and this is the item that determined the height of the next level. Therefore by a similar argument given in the proof of Theorem 1, we get

that the list L_m has a total volume of $(1-\frac{1}{k})^2$. If 2-Dynamic-NFDH uses q bins to pack the items at a given time, then at some earlier point in time, when the first item was placed at bin q, bins $1,\ldots,q-1$ contained a total volume of items of $(q-1)(1-\frac{1}{k})^2$. Since a bin may contain a total volume of at most 1, this total volume can be packed by *OPT* using at least $(q-1)(1-\frac{1}{k})^2$ bins, thus getting an asymptotic competitive ratio of $\frac{1}{(1-\frac{1}{k})^2}$ asymptotically. □

3.2. NFDH for multiple dimensions and its modified version for the dynamic problem

Note that the two-dimensional version of algorithm NFDH works for both squares and rectangles. NFDH in multiple dimensions for cubes was first described by Meir and Moser [20] and used also in [17,2].

Algorithm NFDH for d-dimensional cubes as described in [2]

We use a recursive definition. Assume that we know how to perform NFDH in $d-1$ dimensions. The d-dimensional NFDH heuristic will focus on a facet F of the bin and pack cubes on it using the $(d-1)$-dimensional version of the algorithm. After it finishes packing the facet forming a d-dimensional slice with base F and size equal to the largest cube in the level, it will proceed packing the remaining list in the rest of the bin. For the analysis we make use of the following result from Meir and Moser [20].

Theorem 2. *Let L be a list of d-dimensional cubes, whose sides are bounded from above by $\frac{1}{k}$. Then L can be packed by algorithm NFDH into a d-dimensional $a_1 \times a_2 \times \cdots \times a_d$ parallelepiped if $V(L) \leqslant (\frac{1}{k})^d + (a_1 - \frac{1}{k}) \cdot (a_2 - \frac{1}{k}) \cdot \ldots \cdot (a_d - \frac{1}{k})$.*

In [17] the following corollary was shown.

Corollary 1. *(See [17].) Let P be the packing into unit bins obtained by applying the algorithm NFDH to a list L consisting of d-dimensional cubes whose sides are at most $\frac{1}{k}$. Then each set of cubes packed in a bin used by P (except for possibly one last bin) has volume greater than $(1-\frac{1}{k})^d$.*

Proof. Consider a bin packed by the algorithm which is not the last one. Let L be the list of items packed into it. The next item x does not fit into the bin, therefore, according to Theorem 2, since a bin is a unit cube, the volume of the items of L and x is more than $(\frac{1}{k})^d + (1-\frac{1}{k})^d$. Since the volume of x is at most $(\frac{1}{k})^d$, we get that the volume of L is more than $(1-\frac{1}{k})^d$. □

Note that for squares (i.e., for $d=2$), the claim of Corollary 1 follows from Theorem 1.

d-Dynamic-NFDH for cubes

This algorithm is very similar to the two-dimensional dynamic algorithm 2-Dynamic-NFDH with one exception. Instead of using the two-dimensional algorithm NFDH, the algorithm uses the d-dimensional version of NFDH described above. As in the proof of Lemma 1, if an item does not fit into previous bins, and a new bin is created, all previous bins contain a volume of at least $(1-\frac{1}{k})^d$ each, as stated in Corollary 1.

Offline d-NFDH for boxes

Naturally, the generalization of NFDH for packing multi-dimensional boxes into unit multi-dimensional cubes is more complicated. Sorting cubes by decreasing order according to one dimension implies a decreasing order of every dimension. However, for general boxes, a decreasing order of one dimension is not necessarily a decreasing order in other dimensions. Therefore the simple recursive scheme used for cubes does not work. We first define an offline recursive algorithm called d-unsorted-NFDH.

d-Unsorted-NFDH

We insert items in an arbitrary order, i.e. the items are not sorted by size. The algorithm packs (or tries to pack) item p_i in the current active bin. In order to determine if the new item p_i can fit into the active bin, p_i is merged into the sorted (in non-increasing order) list of the (d-th) component of the bin, the items of the bin are packed again using the sub-routine $(d-1)$-unsorted-NFDH for $d-1$ dimensions (ignoring the d-th component). Boxes are considered in the order specified by the sorted list of last (d-th) component of the size vector of the boxes. If the new packing only uses one bin, (i.e., the sum of the d-th components of first items in $(d-1)$-dimensional "bins" created by the sub-routine is at most 1, further details are given below) it means that p_i can be assigned to that bin. Otherwise the packing of the bin returns to its original packing before the insertion of p_i and the bin is considered closed. A new bin is opened and regarded as the active bin and p_i is placed there.

To check whether an item fits into bin m using $(d-1)$-unsorted-NFDH we do the following. First the list of items in bin m is re-sorted by the d-th component of the items (i.e., the new item is inserted into its correct place in the non-increasing sorted list). Note that the list is sorted by dimension d and not by dimension $d-1$, but the packing of previous items may be

adapted upon insertion of each item. The current list has an arbitrary order of the $d-1$ dimension. As in the d-dimensional algorithm, the items are considered in a non-sorted order. The algorithm $(d-1)$-unsorted-NFDH recursively packs the items into $(d-1)$-dimensional bins without taking into consideration the d-th component of the items. The algorithm d-unsorted-NFDH then receives a number of $(d-1)$-dimensional bins. These bins are levels in a d-dimensional cube. The highest d-dimensional value of an item in each $(d-1)$-dimensional bin (level) is the size of the level which algorithm d-unsorted-NFDH opens for this $(d-1)$-dimensional bin. The algorithm performs a simple test determining whether the resulting packing is in fact a d-dimensional packing of a single bin. This test is very simple, it needs to check whether the sum of sizes of all levels exceeds 1. If so, then the packing needs more than one bin, and we say that item p_i does not fit bin m. If the sum of the values of the d-dimensional levels are below 1 we say that p_i fits bin m. Since the elements are inserted in a decreasing order of their d-th component, the values of the levels decrease. The d-th component of each item in the active $(d-1)$-dimensional bin is smaller than the d-th component of every item in the preceding closed bins. For the case $d=2$ no recursion is needed.

d-Dynamic-NFDH for boxes

Upon arrival of item p_i, the algorithm packs item p_i in the bin of smallest index in which it is possible. The algorithm repositions items inside bins, but it is forbidden to let items migrate between bins. In order to determine if the new item p_i can fit into a specific bin, p_i is merged to the list of the bin, the items of the bin are packed again using the sub-routine d-unsorted-NFDH for d dimensions to the same bin. If the new packing only uses one bin, it means that p_i can be assigned to that bin. Otherwise the packing of the bin returns to its original packing before time a_i and p_i attempts to fit the next bin. If the item cannot fit any of the used bins, a new bin is opened and p_i is placed there.

A formal presentation of the above algorithms is given in Appendix A (Algorithms 1–3).

Theorem 3. *Let L be a list of d-dimensional boxes (for $d \geqslant 3$), where the vector of sizes for an item p_i is $(h_{i_1}, h_{i_2}, \ldots, h_{i_d})$, so that $h_{i_1}, h_{i_2}, \ldots, h_{i_d} \leqslant \frac{1}{k}$. Let P be the packing (into unit bins) obtained by applying the algorithm d-unsorted-NFDH to the list L. Then each set of d-dimensional boxes packed in a bin used by P (except for possibly one last bin) has volume greater than $[([(1-\frac{1}{k})^2(1-\frac{1}{k}) - \frac{1}{k^3}]\cdot[(1-\frac{1}{k})] - \frac{1}{k^4})\cdots][1-\frac{1}{k}] - \frac{1}{k^d}$. In other words, let $V_{k,j} = V_{k,j-1}\cdot(1-\frac{1}{k}) - \frac{1}{k^j}$ for $j > 2$ and $V_{k,2} = (1-\frac{1}{k})^2$ then the volume of each bin (except for possibly one last bin) is greater than $V_{k,d}$.*

The theorem provides a lower bound on the volume which can be packed. Unfortunately, already for $d=2$ and $k=3$ the bound in not tight. It was shown in [24] that a bin which cannot receive an additional item has a packed volume of at least $\frac{9}{16}$ rather than $\frac{4}{9}$. However, the proof of [24] holds only for this special case and relies on case analysis. Therefore, it seems hard to provide a better general lower bound.

Proof. We start by analyzing the three-dimensional algorithm. The 3-unsorted-NFDH algorithm uses as a sub-routine the algorithm 2-unsorted-NFDH. 2-unsorted-NFDH has similar properties to 2-Dynamic-NFDH and therefore by Theorem 1, the 2-unsorted-NFDH algorithm uses at least a total volume of $(1-\frac{1}{k})^2$ in every bin. The three-dimensional value of every item in level ℓ is at most the value of level $\ell+1$. Therefore, by a similar argument given in Theorem 1 if the items are already sorted by the third dimension and are inserted in that order we might lose an additional multiplicative factor of $(1-\frac{1}{k})$. This can be proved by shifting the two-dimensional packed "bins" towards the top of the bin, thus the height of each cell is covered, but the first cell remained uncovered. In that case the algorithm packs a total volume of at least $(1-\frac{1}{k})^3$. Since the items may arrive in arbitrary order of size, the last item (which does not fit) is omitted from an unknown spot in the bin, which causes a loss of a volume of at most $\frac{1}{k^3}$, therefore we are getting a total volume of at least $(1-\frac{1}{k})^3 - \frac{1}{k}^3$ per bin to which an additional item cannot be added.

Further, for the general dimension d, d-unsorted-NFDH algorithm recursively uses $(d-1)$-unsorted-NFDH. For every x, algorithm x-unsorted-NFDH loses an additional volume of $\frac{1}{k^x}$ due to similar reasons. Since the algorithm is recursive, for every algorithm x-unsorted-NFDH, where $x \geqslant 3$, the total utilized volume is the volume utilized by $(x-1)$-unsorted-NFDH (which is denoted by $V_{k,x-1}$), multiplied by $(1-\frac{1}{k})$, but an additive factor of $\frac{1}{k^x}$ must be subtracted. Thus we get the volume indicated in the theorem. □

Lemma 2. *Let L be an input list of items to the dynamic bin packing problem.*

(i) *Let $d \geqslant 3$. If L consists of d-dimensional boxes with maximum size $\frac{1}{k}$ in each dimension. Then the asymptotic competitive ratio of d-Dynamic-NFDH is at most $\frac{1}{[([(1-\frac{1}{k})^2(1-\frac{1}{k})-\frac{1}{k^3}]\cdot[(1-\frac{1}{k})]-\frac{1}{k^4})\cdots][1-\frac{1}{k}]-\frac{1}{k^d}}$.*

(ii) *Assume that L consists of d-dimensional cubes with maximum size $\frac{1}{k}$. Then the asymptotic competitive ratio of d-Dynamic-NFDH is at most $\frac{1}{(1-\frac{1}{k})^d}$.*

Proof. For every input sequence, OPT uses at least the volume of the sequence divided by the volume of a unit capacity multi-dimensional bin. In Corollary 1 and Theorem 3 a lower bound is given on the volume used for each bin (except for

possibly the last bin). The asymptotic competitive ratio is at most the ratio of the volume of a unit bin (which is exactly 1) and the lower bound on the volumes for each bin as stated in Corollary 1 and Theorem 3.

Since OPT can use at most a volume of 1 for each multi-dimensional bin (this is true for every value of d), the ratio between the total volume used by OPT and the dynamic algorithm indicates the asymptotic competitive ratio. □

We use the following lemma later to design competitive algorithms for packing multi-dimensional boxes.

Lemma 3. *The asymptotic competitive ratio of d-Dynamic-NFDH is at most* $\frac{3^d}{2^{d-1}+1} \leqslant \frac{3^d}{2^{d-1}}$ *for* $k = 3$.

Proof. We prove the claim by induction on d. We prove that the inverse of the asymptotic competitive ratio is at least $\frac{2^{d-1}+1}{3^d}$. For $d = 3$, we get that this value is $\frac{5}{27}$, whereas the inverse of the asymptotic competitive ratio is $\frac{7}{27}$. Assume now that the claim is true for $d-1$. Then the value for d is at least $\frac{2}{3}(\frac{2^{d-2}+1}{3^{d-1}}) - \frac{1}{3^d} = \frac{2^{d-1}+1}{3^d}$. □

d-NFDH for the offline problem with multiple dimensions

We summarize this section by presenting a slightly better offline algorithm than d-unsorted-NFDH. To the best knowledge of the authors, NFDH for the offline packing problem of d-dimensional boxes was never presented or analyzed. We now define the offline algorithm d-NFDH. We perform a small modification compared to the algorithm d-unsorted-NFDH. Instead of using an arbitrary order of the elements we first sort all the elements by their d-th component. Everything else remains the same. However, after one dimension is sorted we face the same problem as before with any other dimension. The suggested order does not imply an ordering in these dimensions. So the proposed algorithm would be the same in the sense that all the sub-routines still need to apply $(d-1)$-unsorted-NFDH to deal with that issue.

This offline algorithm has a slightly better asymptotic competitive ratio since the d-dimensional order is given in advance. Therefore this version does not lose an additional volume of $\frac{1}{k^d}$ for each bin, and therefore the approximation ratio is slightly lower than the asymptotic competitive ratio of the dynamic algorithm.

4. The two-dimensional problem

In this section we consider the two-dimensional dynamic bin packing problem. Namely, dynamic packing of rectangles and squares. We present both lower bounds and upper bounds for these variants. For convenience we present a sketch of the lower bound of [5] for one-dimensional dynamic bin packing.

Lemma 4. *(See [5].) Any online algorithm for one-dimensional dynamic bin packing has an asymptotic competitive ratio of at least* 2.428.

Proof. The sequence of items is given in n stages. At each point in time, OPT uses at most $F = n!(n-1)!$ bins. In each stage, some items which arrived in the previous stages depart and a number of items, which have the same total size as the departing items are given.

In the first stage, $F \cdot n$ items of size $\frac{1}{n}$ are given. The online algorithm A uses at least F bins to pack the Fn items. If A uses more than F bins, all items in bins other than the first F bins depart.

In each of the subsequent stages, i.e., stage i, for $2 \leqslant i \leqslant n$, there are two steps. (1) For each occupied bin, all its items except the smallest one depart. (2) Let R_i be the total size of the remaining items. The adversary then presents $(F - R_i)(n - i + 1)$ items of size $\frac{1}{n-i+1}$. At this point, the total size of all active items (i.e., items that did not depart), including those given in previous stages, is F.

In each stage, there is a minimum number of bins that the online algorithm has to open. If the online algorithm uses more than that the adversary lets all items in the additional used bins depart. Since $F = n!(n-1)!$, all R_i are integers (see [5]). Using a computer program, Chan et al. showed that this process results in a lower bound of 2.428 (achieved for $n = 12794$). □

4.1. Square packing

In this version squares are packed into unit squares in an online fashion. We provide a 4.2154-competitive algorithm.

Algorithm SQP. We perform an online partition of the input squares into three sub-sequences according to the height (or width) of the squares. Each one of the sub-sequences created in this way is packed separately. The sub-sequences are defined as follows. For every input square I of side a, we let $I \in S_1$ if $a \leqslant \frac{1}{3}$, $I \in S_2$ if $a \in (\frac{1}{3}, \frac{1}{2}]$ and $I \in S_3$ if $a > \frac{1}{2}$.

Sub-Algorithm A_{S_1}**.** Pack items in S_1 using the algorithm 2-Dynamic-NFDH.

Sub-Algorithm A_{S_2}**.** Each bin is divided into four sub-bins of size $\frac{1}{2} \times \frac{1}{2}$, each of which can contain at most one item. Pack a new item $I \in S_2$, in the first available sub-bin. Open a new bin for this class if all the existent ones contain exactly four items.

Sub-Algorithm A_{S_3}**.** Pack every new item in S_3 into an empty bin.

Theorem 4. *Algorithm SQP has an asymptotic competitive ratio of at most* 4.2154.

Proof.

Sub-Algorithm A_{S_1}**.** According to Lemma 2 the asymptotic competitive ratio for S_1 is at most $\frac{1}{(1-\frac{1}{3})^2} = \frac{9}{4} = 2.25$. But for this particular partition, a smaller asymptotic competitive ratio can be achieved due to [24]. In this paper it is shown that if all squares packed by NFDH are in S_1, each bin contains a total volume of $\frac{9}{16}$. Since the items are inserted in an unsorted order, we may lose an volume equal to the volume of the last item (which did not fit), which in this case can be at most $\frac{1}{9}$. Therefore the total volume used by 2-Dynamic-NFDH is $\frac{9}{16} - \frac{1}{9} = \frac{65}{144}$. This gives an asymptotic competitive ratio of at most $\frac{144}{65} \simeq 2.2154$.

Sub-Algorithm A_{S_2}**.** At most four items in S_2 can be packed in a single bin. It is not hard to see that OPT (of S_2) cannot pack more than four items in a bin. If a new bin is opened for this class, let m be the number of bins for this class at that time. The number of items of this class, which arrived already and did not depart yet is $4m + 1$. Therefore $OPT(S_2) \geqslant m + 1$. Since A_{S_2} has four packed items in every bin if a new bin is opened, we have an asymptotic competitive ratio of 1.

Sub-Algorithm A_{S_3}**.** In every packing of S_3, exactly one item of this set can be packed in each bin. Since both OPT and A_{S_3} accommodate one item of S_3 per bin, this class is packed optimally similarly to the previous class and which results in an asymptotic competitive ratio of 1 for this class.

Summing up the asymptotic competitive ratios of the three sub-algorithms of algorithm SQP, we have an asymptotic competitive ratio of at most 4.2154. □

Theorem 5. *Any online algorithm for dynamic square packing has an asymptotic competitive ratio of at least* 2.2307.

Proof. The following observations and lemmas are introduced to be used in the proof. The lower bound sequence contains among other items, squares of side $\frac{1}{2}$ and squares of side $\frac{1}{3}$. We are interested in the effect of placing these two types of items in the same bin (i.e., the numbers of such items that can fit together).

Divide a unit square into 36 identical squares of side $\frac{1}{6}$. These squares can be formed by drawing five horizontal lines with distances of $\frac{1}{6}$ between any two horizontal lines (including the horizontal sides of the bin). Vertical lines are created similarly. We make two observations regarding packings in which only these two sizes of items are packed, and at least one item of each one of the two sizes is packed.

Observation 4. There exists an optimal placement for squares of side $\frac{1}{2}$ where all these squares are located in the corners of the bin.

Proof. Suppose a square of these dimensions is not placed in one of four corners. If the square is not aligned to the right border of the bin, then either there is no room for any additional square of sides $\frac{1}{3}$ or $\frac{1}{2}$ to the left of this items or to its right (or both). In that case there is one side with a distance of less than $\frac{1}{3}$ to the frame of the bin. The square can be pushed to that side only making room for possibly more squares. Using a similar argument, we can shift the square towards either the top or the bottom of the bin. □

Observation 5. The best positions for squares of side $\frac{1}{3}$ are the 25 following positions. These positions are squares which are sets of four sub-squares aligned to the horizontal and vertical lines (including the frame of the bin). Note that the larger squares are aligned with the lines, according to the observation above.

Proof. Suppose a square of this size is not adjusted to a vertical line. In that case there is one side with a distance of less than $\frac{1}{6}$ to the frame of the bin or to a line. If this space is free, the square can be pushed to that side only making room for possibly more squares. Otherwise, it can be shifted until it reaches another square and then a similar argument can be applied to both squares together (there cannot be a set of three such squares since three squares occupy a width or height of 1). □

Lemma 5. *If six squares of side* $\frac{1}{3}$ *are placed in a bin, no squares of* $\frac{1}{2}$ *can be packed.*

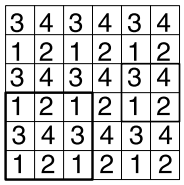

Fig. 1. Coloring of the unit square into four colors. Note that the marked square of side $\frac{1}{2}$ contains four appearances of the color 1. The marked square of side $\frac{1}{3}$ contains one instance of each color.

Proof. We prove this by showing that if a square of side $\frac{1}{2}$ is placed in a bin, then there is only room for at most five squares of side $\frac{1}{3}$.

Partition the unit square into 36 identical squares of side $\frac{1}{6}$ as above. Now color each $\frac{1}{6} \times \frac{1}{6}$ square with one of four colors. The bottom row of squares are colored by the colors 1 and 2 interchangeably, i.e., the first, third and fifth squares are colored with color 1 and the rest with color 2. Do the same with the second row with the colors 3 and 4. The odd rows will be colored the same way as the first row and the even rows exactly as the second row. See Fig. 1. Consider an optimal packing where squares are positioned according to the two observations. Note that every square of side $\frac{1}{3}$ contains one square of every color. Also note, that for any possible location of a square of side $\frac{1}{2}$, there exists a color that appears four times in the square i.e., contains four colored $\frac{1}{6} \times \frac{1}{6}$ squares of the same color. See Fig. 1. Since there are nine colored squares of each color and since a $\frac{1}{2} \times \frac{1}{2}$ square contains four squares of one color, there is only room for at most five additional squares of side $\frac{1}{3}$. □

Lemma 6. *If four squares of side $\frac{1}{3}$ are placed in a bin, at most one square of side $\frac{1}{2}$ can be packed.*

Proof. We prove this by showing that if two squares of side $\frac{1}{2}$ are placed in a bin, then there is only room for at most three squares of side $\frac{1}{3}$.

Suppose the squares of side $\frac{1}{2}$ are placed according to the first observation. There are two possible placements for these squares. In the first one, the squares are in the same row or column and in the second one, they are in opposite corners. The first case an empty rectangle of size $1 \times \frac{1}{2}$. At most three squares of side $\frac{1}{3}$ can be placed in this rectangle.

The second case leaves two squares of size $\frac{1}{2} \times \frac{1}{2}$ for the squares of side $\frac{1}{3}$. Only one such square can be located in every empty square. Therefore in this case, it is possible to pack at most two squares of side $\frac{1}{3}$. □

Lemma 7. *If two squares of side $\frac{1}{3}$ are placed in a bin, at most two squares of side $\frac{1}{2}$ can be packed.*

Proof. We prove this by showing that if three squares of side $\frac{1}{2}$ are placed in a bin, then there is only room for at most one square of side $\frac{1}{3}$. Any packing of three $\frac{1}{2} \times \frac{1}{2}$ squares that follows the first observation leaves a single square of size $\frac{1}{2} \times \frac{1}{2}$. This square can contain at most only one square of side $\frac{1}{3}$. □

Lemma 8. *If one square of side $\frac{1}{3}$ are placed in a bin, at most three squares of side $\frac{1}{2}$ can be packed.*

Proof. We prove this by showing that if four squares of side $\frac{1}{2}$ are placed in a bin, then there is no room for squares of side $\frac{1}{3}$. This is trivial since if four squares of side $\frac{1}{2}$ are placed in a bin, then the bin is completely full. □

Next, we describe the lower bound sequence. The input sequence consists of three or four steps. We build the sequence with accord to the behavior of an online algorithm, using four cases. Throughout the sequence we have $OPT = N$, for a very large integer value N whose properties are as follows. N is a square of an integer number. OPT always packs the items which are not going to leave in separate bins. Items that are going to leave are packed by OPT in other bins, which get emptied before new items arrive.

The first step consists of the arrival of N^2 squares of side ε, where $\varepsilon = \frac{1}{\sqrt{N}}$. Any feasible packing must pack the input into at least N bins. We let all items leave except for a single item per bin, in exactly N bins. We call these bins "initial bins". OPT packs these items (that do not leave) into a single bin.

Next $9(N-1)$ items of side $\frac{1}{3}$ arrive, again $OPT = N$. Note that only eight such items can still fit into each initial bin. Therefore at least $9N - 9 - 8N = N - 9$ items need to be packed into new bins and thus $\frac{N}{9} - 1$ new bins are created.

At this time, consider only the new bins packed by the algorithm. Let γ_1 be the number of bins which contain at least six items, γ_2 the number of bins with four or five items, γ_3 the number of bins with two or three items and γ_4 the number of bins with a single item. Note that the initial bins can receive at most eight items each, but the new bins can receive at

most 9 items each and therefore $9\gamma_1 + 5\gamma_2 + 3\gamma_3 + \gamma_4 \geqslant N - 9$. We sort the new bins according to the number of items in them starting from largest. We let some items leave, in a way that the first γ_1 bins contain exactly six items, the next γ_2 bins contain exactly four items, the next γ_3 bins contain exactly two items and the last γ_4 bins remain with a single item each. All items from the initial bins leave, except for one small item from each such bin. OPT packs these items using in at most $\lceil \frac{6\gamma_1+4\gamma_2+2\gamma_3+\gamma_4}{9} \rceil \leqslant 6\frac{\gamma_1}{9} + 4\frac{\gamma_2}{9} + 2\frac{\gamma_3}{9} + \frac{\gamma_4}{9} + 1$ bins. Moreover, in the optimal packing we construct, the bins are packed so that the items that are not going to leave (one item per new bin of the online algorithm) are packed first, and only then the rest of the items, which will leave later, and empty all bins containing these items, but possible one bin that received also items that are not planned to leave.

There is an optional step in which items of side $\frac{1}{2}$ arrive. The number of such items is $x = 4(N - 1 - \lceil \frac{6\gamma_1}{9} + \frac{4\gamma_2}{9} + \frac{2\gamma_3}{9} + \frac{\gamma_4}{9} \rceil)$. Out of these items, at most three can be packed into each initial bin. According to Lemma 5, each open bin among the first γ_1 new bins, cannot receive items. According to Lemma 6, each open bin among the next γ_2, can only receive a single item. According to Lemma 7, each open bin among the next γ_3, can receive at most two items. According to Lemma 8, each open bin among the last γ_4, can receive at most three items. If the value $\mu = \lceil \frac{x-3N-\gamma_2-2\gamma_3-3\gamma_4}{4} \rceil$ is positive, we get at least this number of additional bins. If this step is performed, all items of side $\frac{1}{2}$ leave expect for a single item in μ bins. OPT packs these items in $\lceil \frac{\mu}{4} \rceil$ bins. If we did not perform this step, let $\mu = 0$.

Next, all items of side $\frac{1}{3}$ leave except for one per bin of the online algorithm. Finally, items of side 1 arrive. The number of such items is $N' = N - 1 - \lceil \frac{\gamma_1+\gamma_2+\gamma_3+\gamma_4}{9} \rceil - \lceil \frac{\mu}{4} \rceil$. They are packed all in new bins. Therefore, the number of bins used by the algorithm is $N' + \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 + \mu + N \geqslant 2N - 3 + \frac{8(\gamma_1+\gamma_2+\gamma_3+\gamma_4)}{9} + \frac{3\mu}{4}$.

Since if $\mu = 0$, $\frac{x-3N-\gamma_2-2\gamma_3-3\gamma_4}{4}$ is negative and thus smaller than μ, the expression only decreases if we replace μ by $\frac{x-3N-\gamma_2-2\gamma_3-3\gamma_4}{4}$. We use this value for μ in both cases. We get a lower bound of $\frac{35N}{16} - 5 + \frac{56\gamma_1+53\gamma_2+50\gamma_3+59\gamma_4}{144} \geqslant \frac{35N}{16} - 5 + 9\gamma_1 + 5\gamma_2 + 3\gamma_3 + \gamma_4 \frac{56}{1296} \geqslant \frac{35N}{16} - 5 + \frac{56(N-9)}{1296} \geqslant \frac{2891N}{1296} - 6 \approx 2.23071N - 6$.

For large enough N, we get a lower bound slightly larger than 2.23. Note that this bound is higher than the one achieved by running a computer program that applies the method of Lemma 4 on squares, which gives the lower bound 2.2239. □

4.2. Rectangle packing

In this section we use the analysis of the upper bound for the one-dimensional First Fit algorithm presented in [7] for dynamic bin packing, where the following lemma was proved.

Lemma 9. *(See [7].) Let L be a sequence of items presented to the dynamic bin packing problem. If each $p_i \in L$ satisfies $p_i \leqslant \frac{1}{k}$. The First Fit algorithm has an asymptotic competitive ratio of at most $\frac{k+1}{k} + \frac{1}{k-1} \ln \frac{k^2}{k^2-k+1}$.*

Algorithm RCP. We perform an online partition of the items into several sub-sequences according to the width and the height of the rectangles. The sub-sequences are defined as follows.

For every arriving rectangle I with the size vector (a, b), if $a > \frac{1}{2}$ and $b > \frac{1}{2}$, $I \in S_1$. If $a > \frac{1}{2}$ and $b \leqslant \frac{1}{2}$, $I \in S_2$. If $a \leqslant \frac{1}{2}$ and $b > \frac{1}{2}$ let $I \in S_3$. Otherwise $I \in S_4$ (in this case, $a \leqslant \frac{1}{2}$ and $b \leqslant \frac{1}{2}$).

Sub-Algorithm A_{S_1}**.** Pack each new item of S_1 in an empty bin.

Sub-Algorithm A_{S_2}**.** Treat every new item $I \in S_2$ as a one-dimensional item of size $b < \frac{1}{2}$ (i.e., the "size" of an item is its height). Use one-dimensional (dynamic) First Fit (FF) with respect to the value of b. The rectangles are stacked one on top of the other in each bin.

Sub-Algorithm A_{S_3}**.** Similarly to the previous case, treat every $I \in S_3$ as a one-dimensional item of size $a < \frac{1}{2}$ (i.e., the "size" of an item is its width). Use one-dimensional (dynamic) First Fit (FF) with respect to the value of a.

Sub-Algorithm A_{S_4}**.** Pack the items of S_4 using the algorithm 2-Dynamic-NFDH.

Theorem 6. *Algorithm RCP has an asymptotic competitive ratio of at most* 8.5754.

Proof.

Sub-Algorithm A_{S_1}**.** Only one item in S_1 can be packed into each bin. Since both OPT and A_{S_1} can accommodate exactly one item of S_1 in every bin we have an asymptotic competitive ratio of 1.

Sub-Algorithm A_{S_2}**.** The height of every $I \in S_2$ is strictly greater than $\frac{1}{2}$ (satisfies $a > \frac{1}{2}$). Therefore no item can be positioned next to another item. Hence, using FF, we position the rectangles only according to height, which is a one-dimensional value. Additionally, since the height of every $I \in S_2$ is smaller or equal to half (satisfies $b \leqslant \frac{1}{2}$), using [7], we deduce that the asymptotic competitive ratio of FF which is at most 1.787682.

Sub-Algorithm A_{S_3}**.** By a similar argument to the one for A_{S_2}, we get an asymptotic competitive ratio of at most 1.787682 for A_{S_3}.

Sub-Algorithm A_{S_4}. Since for every $I \in S_4$ the width and the height of I both are bounded from above by $\frac{1}{2}$. By Lemma 1 we get an asymptotic competitive ratio of at most 4.

To complete the analysis we add up the asymptotic competitive ratios of all the sub-algorithms and get $1+2\cdot 1.787682+4=8.5754$. □

Theorem 7. *Any online algorithm for dynamic rectangle packing has an asymptotic competitive ratio of at least* 3.70301.

Proof. The presented items are of two types. The first type of rectangles have width 1 and variable heights. The second type have height of 1 and an variable widths. Observe that given the restriction on the items, a pair of items from different types cannot be assigned to the same bin. Therefore items positioned in a bin must be of the same type.

The sequence of rectangles consists of n phases. Each phase is composed of two parts. In the first part of phase i, items of the first type and height $\frac{1}{n-i+1}$ are given and some of them depart right away. In the second part, items of the second type and width $\frac{1}{n-i+1}$ are given and some of them depart right away.

At each point in time, OPT uses at most F bins. At the beginning of each part of every phase, all bins of the online algorithm contain a single item.

Denote the volume of items accommodated in the used bins by the online algorithm before the beginning of phase i by $R_{i,1}$, and $R_{i,2}$ before the beginning of the second part of the phase. Since OPT can put sets of items of the same size together in the same bin and utilize the whole content of the bin (except for one bin for item size), OPT uses at most $R_{i,1}+2(i-1)$ bins before the beginning of phase i, and at most $R_{i,2}+2(i-1)+1$ bins before the beginning of the second part of the phase.

By presenting $(n-i+1)(F-R_{i,1}-2i+2)$ items in the first part and $(n-i+1)(F-R_{i,2}-2i+1)$ in the second part, the number of bins used by OPT does not exceed F.

The online algorithm can place items of the first type of phase i in bins containing items of previous phases. Since the rectangles are given in an increasing order of size, the online algorithm can pack $n-i$ items in every such bin. Bins of items of different type cannot be used and therefore, the rest of the items must be located in new bins. At every time, it is possible to compute a lower bound on the number of new bins. We denote the lower bound on the number of new bins for the first and second parts of phase i by $L_{i,1}$ and $L_{i,2}$. When items of these phases leave, only this number of bins of the algorithm contains one item. All other new bins, if such bins exist, are emptied completely.

The last phase only consists of items of one type since both types are the same item with width and height of 1.

Let F be the number of bins used by OPT. The lower bound is the sum of all numbers of new bins over all phases, divided by F. The number F can be pre-computed so the number of bins that must be opened in every part of every phase is integer and also that the total volume of items of the same type are integers as well.

Using a computer program which calculates the values $R_{i,j}$, using $n=10\,000$, we get a lower bound of 3.70301. □

The trivial approaches give worse lower bounds. For example, by presenting all items of type one (ordered by side) and subsequently all items of type two, we get a lower bound of 3.53. Also by presenting small items of different types and then a sequence of one type of items also leads to worse results.

By presenting items with different types interchangeably, more items of small size are given and a bigger portion of the bins is left unused.

5. Multiple dimensions

5.1. d-Dimensional cube packing

Algorithm *d*-Dynamic-CP. We perform an online partition of the items into several sub-sequences according to size. Let k be an integer value chosen later. The sub-sequences are defined as follows. For every cube I of side a, if $a \leqslant \frac{1}{k}$ then $I \in S_{Small}$. For every $i=1,\ldots,k-1$, $I \in S_i$ if $a \in (\frac{1}{i+1}, \frac{1}{i}]$.

The sequences S_i, for $i=1,\ldots,k-1$, are packed using a dynamic multi-dimensional version of Harmonic [18]. Every bin that are used to pack items of these sequences, contains items of a single sequence. We use the following algorithm to pack these items.

Sub-Algorithm A_{S_i}. Each bin is partitioned into i^d cubes (sub-bins), each having a volume of $\frac{1}{i^d}$ (the sub-bins create a grid of i strips in each dimension). Each such sub-bin can contain exactly one item of type i. On arrival of item in S_i, it is assigned to the first used bin that has a free sub-bin (and placed anywhere inside this sub-bin). If all sub-bins of used bins are taken, a new bin is opened and partitioned into sub-bins and the item is placed in one of the sub-bins of the new bin.

Sub-Algorithm $A_{S_{Small}}$. Use d-Dynamic-NFDH to pack S_{Small}.

Lemma 10. *The asymptotic competitive ratio of d-Dynamic-CP for a given value of k is at most* $k-1+(\frac{k}{k-1})^d$.

Proof. Since the sub-sequences are packed independently, the asymptotic competitive ratio is bounded by their sum. For each S_i ($1 \leqslant i \leqslant k-1$), a packed bin of any algorithm can contain at most i^d items (see, e.g., [13]). Therefore upon arrival of a new item, the packing of d-Dynamic-CP for S_i is optimal. These sub-sequences contribute $k-1$ to the asymptotic competitive ratio. By Lemma 2(ii), the asymptotic competitive ratio for S_{Small} is at most $(\frac{k}{k-1})^d$. □

Theorem 8. *Taking $k = \frac{2d}{\ln d} + 1$, algorithm d-Dynamic-CP has an asymptotic competitive ratio of $O(\frac{d}{\ln d})$.*

Proof. We use the previous lemma. Take $k = \kappa = \frac{2d}{\ln d} + 1$. We get the upper bound on the competitive ratio, as follows:

$$\kappa - 1 + \left(1 + \frac{1}{\kappa - 1}\right)^d = \kappa - 1 + \left(\left(1 + \frac{1}{\kappa - 1}\right)^{\kappa-1}\right)^{\frac{d}{\kappa-1}} \leqslant \frac{2d}{\ln d} + e^{\frac{\ln d}{2}} \leqslant \frac{2d}{\ln d} + \sqrt{d} = O\left(\frac{d}{\ln d}\right). \qquad \square$$

Theorem 9. *Any d-dimensional dynamic cube packing online algorithm has an asymptotic competitive ratio of at least 2.*

Proof. The construction is divided into two parts. In the first part, cubes of volume ε^d are given and a large portion of them departs. In the second part, items of side 1 are given. We show that at any point in time, OPT uses at most F bins and any online algorithm eventually uses at least $2F - 1$ bins. This is shown for large values of F. In the first part we introduce F^{d+1} items of side $(\frac{1}{F})$. These items have a total volume of F, and moreover, F^d items can fit in one bin, therefore the algorithm will assign them to at least F bins, and OPT can use exactly F bins, the first one of which would contain the F items that are not going to leave. In each of the first F bins opened by an online algorithm, all the items depart except for one item per bin. If the online algorithm had used more than F bins, all the items of these bins depart. In the second part of the construction, $F - 1$ items of side 1 arrive. Since these items need a complete bin for each one of them, every item must be placed in a new bin. The total number of bins used by the online algorithm is therefore $2F - 1$. OPT has a single packed bin before these items arrive and thus it still has a cost of F after their arrival. This means that OPT uses only F bins for both parts. Since F can be made arbitrary large, the lower bound is arbitrarily close to 2. □

5.2. d-Dimensional box packing

Algorithm *d*-Dynamic-BP. We perform an online partition of the items into several sub-sequences according to the sides of different components of the size vectors of the d-dimensional boxes. We consider a partition into two cases for each dimension, the case where the size is strictly above $\frac{1}{3}$ and the complement case where the size is smaller or equal to $\frac{1}{3}$. Each sub-sequence $S_{(s_1,s_2,\ldots,s_d)}$ is defined as follows.

For a d-dimensional box I, let $(h_1, h_2, \ldots, h_d)$ be a vector of its sizes. $I \in S_{(s_1,s_2,\ldots,s_d)}$ where $s_i \in \{0, 1\}$ for all i, and moreover for every $1 \leqslant i \leqslant d$, $s_i = 1$ if $h_i \in (\frac{1}{3}, 1]$ and otherwise $s_i = 0$. For example the three-dimensional box $(\frac{1}{4}, \frac{3}{4}, \frac{1}{3})$ is in the sub-sequence $S_{(0,1,0)}$.

For every sub-sequence $S_{(s_1,s_2,\ldots,s_d)}$, if $s_i = 1$, the i-th coordinate of every item in $S_{(s_1,s_2,\ldots,s_d)}$ is strictly above $\frac{1}{3}$. This means that for this specific sub-sequence, in any packing, dimension i can contain at most two layers of items at every point. That is, any ray in the i-th direction intersects with at most two items, in each bin dedicated to this sub-sequence. We make use of only one layer, thus we lose a multiplicative factor of at most two. Therefore the position of items in the i-th direction is irrelevant (for this sub-sequence).

Hence for every sub-sequence $S_{(s_1,s_2,\ldots,s_d)}$ we may consider only the zero values of s_i. If $S_{(s_1,s_2,\ldots,s_d)}$ consists of x zero values, and $d - x$ non-zero values, we use x-Dynamic-NFDH algorithm to position the items. For all the dimensions with $s_i = 1$, we align the items to the facet of that direction in the bin.

For the special case of the sub-sequence $S_{(1,1,\ldots,1)}$ which consists of only items with size greater than $\frac{1}{3}$ in every direction, we assign each new item in an empty bin.

Theorem 10. *Algorithm d-Dynamic-BP has an asymptotic competitive ratio of at most $2 \cdot 3.5^d$.*

Proof. We run a number of sub-routines of D-NFDH with different dimensions. The number of m-Dynamic-NFDH sub-algorithms equals to the number of sequences of length d with m values equal to 0. This value is $\binom{d}{m}$. For every $s_i = 1$ we lose a multiplicative factor of 2 as noted above. According to Lemma 3, the asymptotic competitive ratio of m-Dynamic-NFDH is at most $\frac{3^d}{2^d-1}$. The asymptotic competitive ratio is the sum over all asymptotic competitive ratios. Using the binomial theorem, we get the following. $\sum_{i=1}^{d} (\binom{d}{i} \frac{3^i}{2^i-1} \cdot 2^{d-i}) = 2 \cdot 3.5^d$. □

Theorem 11. *Any d-dimensional online dynamic box packing algorithm has an asymptotic competitive ratio of at least $d + 1$.*

Proof. We present a sequence of d-dimensional boxes. Through the whole process OPT uses at most $k \gg d$ bins. The sequence is built using $d+1$ phases. In phase i for $i = 1, \ldots, d$ we present boxes of size $\varepsilon = \frac{1}{k}$ in the i-th dimension and size 1 in any other dimension. Note that boxes from different phases cannot be placed in the same bin. After the online algorithm accommodates these items, we keep exactly one item in each bin and release all other items. At the end of phase i every bin consists of single item from some phase in $1, \ldots, i$. The last phase is different. In the last step we present $k - d$ boxes with size 1 in each direction (i.e., unit cubes).

In phase i we present $k(k - i + 1)$ items (of volume ε). Since these boxes cannot be placed with items from previous bins, the minimum number of bins required to pack these items is $k - i + 1$. We leave a single item in the first $k - i + 1$ new opened bins and release all other items that were presented in the phase. Observe that since $\varepsilon = \frac{1}{k}$, all the boxes left in the new bins can be placed in a single bin by OPT.

After phase d, the number of bins that OPT uses to pack all the items left equals to d. Therefore by presenting $k - d$ boxes of volume 1, OPT can pack the entire sequence using exactly k bins. The number of bins used by any online algorithm is therefore as follows:

$$\sum_{i=1}^{d}(k - i + 1) + k - d = \sum_{i=1}^{d+1}(k - i + 1) = (d+1)k - \frac{d(d+1)}{2}.$$

Since OPT can use only k bins, using a large enough value of k, we get a lower bound of $d + 1$. □

Appendix A. A formal description of the algorithms used in the paper

Let n_i be the number of bins used before the arrival of p_i.
Let L_j be a list of items in bin j, $j = 1, \ldots, n_i$ before the arrival time a_i of item p_i.
On a new arrival of item p_i,

```
for k = 1 to n_i do
    merge p_i to L_k and sort the new list by the second component.
    Perform NFDH on the sorted list L_k ∪ p_i
    if NFDH uses only one bin to pack the sorted list L_k ∪ p_i then
        Pack p_i in the position indicated by NFDH and re-position all the elements of L_k according to the new packing of L_k ∪ p_i by NFDH and stop.
    else
        The packing of bin k is returned to the original packing of L_k.
    end if
end for
if the item p_i, does not fit in any of the bins k = 1, ..., n_i then
    Open a new bin n_i + 1 and pack p_i in that bin.
end if
```

Algorithm 1. 2-Dynamic-NFDH.

Let n_i be the number of bins used before the arrival of p_i.
Let L_j be a list of items in bin j, $j = 1, \ldots, n_i$ before the arrival time a_i of item p_i.
On a new arrival of item p_i,

```
for k = 1 to n_i do
    merge p_i to L_k.
    Perform d-unsorted-NFDH on the sorted (by last component) list L_k ∪ p_i
    if d-unsorted-NFDH uses only one bin to pack the sorted list L_k ∪ p_i then
        Pack p_i in the position indicated by d-unsorted-NFDH and re-position all the elements of L_k according to the new packing of L_k ∪ p_i by
        d-unsorted-NFDH and stop.
    else
        The packing of bin k is returned to the original packing of L_k.
    end if
end for
if the item p_i, does not fit in any of the bins k = 1, ..., n_i then
    Open a new bin n_i + 1 and pack p_i in that bin.
end if
```

Algorithm 2. d-Dynamic-NFDH.

Let n_i be the number of bins used before the insertion of p_i.
Let L_j be the sorted list of items in bin j, $j = 1, \ldots, n_i$ by the d-dimensional value, before the insertion of item p_i.
On a new arrival of item p_i,

for $k = 1$ to n_i **do**
  merge p_i to L_k and sort the new list by the d-dimensional value of the items, denote this list by $L_{k,new}$.
  Perform $(d-1)$-unsorted-NFDH on the sorted list $L_{k,new}$, i.e., re-insert the items in the order of $L_{k,new}$ to the algorithm $(d-1)$-unsorted-NFDH ignoring the items' d-dimensional value.
  Associate each $(d-1)$-dimensional bin output by the algorithm $(d-1)$-unsorted-NFDH with the largest d-dimensional value of the items in the bin. This is called the value of the level of the bin.
  if the sum of the values of the levels of the bins are below 1 **then**
    Pack p_i in the position indicated by the recursive algorithm $(d-1)$-unsorted-NFDH and re-position all the elements of L_k according to the new packing of $L_k \cup p_i$ by $(d-1)$-unsorted-NFDH and stop.
  end if
end for
if the item p_i, does not fit in the active bin **then**
  Close bin n_i and open a new bin $n_i + 1$ which is considered to be the active bin and pack p_i in that bin.
end if

Algorithm 3. d-Unsorted-NFDH.

Appendix B. The three-dimensional problem

B.1. Cube packing

Algorithm 3D-CubeP. We perform an online partition of the three-dimensional cubes into four sub-sequences according to the sides of the cubes. The sub-sequences are defined as follows.

For a cube I of side a, if $a \leqslant \frac{1}{4}$ then $I \in S_1$. If $a \in (\frac{1}{4}, \frac{1}{3}]$, $I \in S_2$. If $a \in (\frac{1}{3}, \frac{1}{2}]$, $I \in S_3$ and finally if $a > \frac{1}{2}$, $I \in S_4$.

Sub-Algorithm A_{S_1}**.** Pack S_1 using the algorithm 3-Dynamic-NFDH for cubes.

Sub-Algorithm A_{S_2}**.** Each bin is partitioned into 27 identical cube sub-bins, all of them having a side of $\frac{1}{3}$. We can pack at most 27 items in a bin. A new item $I \in S_2$, is packed into the first such sub-bin that is available. If no such sub-bin exists, a new bin is opened for this class.

Sub-Algorithm A_{S_3}**.** Each bin is partitioned into eight identical cube sub-bins, all of them having a side of $\frac{1}{2}$. We can pack at most eight items in a bin. A new item $I \in S_3$, is packed into the first such sub-bin that is available. If no such sub-bin exists, a new bin is opened for this class.

Sub-Algorithm A_{S_4}**.** Every $I \in S_4$ is packed into an empty bin.

Theorem 12. *Algorithm 3D-CubeP has an asymptotic competitive ratio of at most* 5.37037.

Proof. The proof is similar to the proof of Theorem 4.

Sub-Algorithm A_{S_1}**.** According to Lemma 2 the asymptotic competitive ratio on S_1 is at most $\frac{1}{(1-\frac{1}{4})^3} = 2.37037$.

Sub-Algorithm A_{S_2}**.** At most 27 items in S_2 can be packed in a single bin. Since both OPT and A_{S_2} have at most 27 items in a bin we have an asymptotic competitive ratio of 1.

Sub-Algorithm A_{S_3}**.** At most 8 items in S_2 can be packed in a single bin. Since both OPT and A_{S_3} have 8 items in a bin we have an asymptotic competitive ratio of 1.

Sub-Algorithm A_{S_4}**.** Only one item in S_4 can be packed in a single bin. Since both OPT and A_{S_4} accommodate one item of S_4 in a bin we have an asymptotic competitive ratio of 1.

Summing up the asymptotic competitive ratios of the three sub-algorithms of algorithm SQP, we have an asymptotic competitive ratio of 5.37037. □

Lemma 11. *Given a three-dimensional bin with four items, which are cubes of side* $\frac{1}{3}$*, packed in it, the maximum number of cubes of side* $\frac{1}{2}$ *which can fit into the bin is at most five.*

Proof. We prove that if we can pack six items with side $\frac{1}{2}$, then only three smaller items can fit. We first show that without loss of generality, each larger item is located in a corner. Assume that one such item is not in a corner. Do the following process for each of the three dimensions. Take the projection of the bin and item on one axis. Since the item is a cube of side $\frac{1}{2}$, its projection is an interval of that length. Therefore, either to its left or to its right, there are no projections of other items (of side $\frac{1}{3}$ or $\frac{1}{2}$). Shift the larger item towards that direction until it is located at the leftmost or rightmost position. After this is done in all three dimensions, the item is located at a corner.

Consider now a cube where six larger items are located. This leaves two cubes of side $\frac{1}{2}$ free to receive items. If they are located inside the bin in a way that only their corners are touching, they can receive only one smaller item each. If they are connected, we have a box of sides $\frac{1}{2}, \frac{1}{2}, 1$ ready to receive smaller items. The dimensions of sides $\frac{1}{2}$ can actually only receive $\frac{1}{3}$, and therefore we can fit exactly three smaller items. □

Theorem 13. *The asymptotic competitive ratio of any algorithm for dynamic three-dimensional cube packing is at least* 2.11696.

Proof. The input sequence consists of three or four steps. We build the sequence with accord to the behavior of an online algorithm. Throughout the sequence we have $OPT = N$, for a very large integer value N whose properties are as follows. N is a cube of an integer number, and it is divisible by 3^4K, where $K = 13\,807$.

OPT always packs the items which are not going to leave in separate bins, and those that will leave in other bins that get emptied before new items arrive.

The first step consists of the arrival of N^2 cubes of side $\frac{1}{\varepsilon}$, where $\varepsilon = \frac{1}{N^{1/3}}$. The input must be packed into at least N bins. We let all items leave except for a single item per bin, in exactly N bins, we call these bins "initial bins". OPT packs these items into a single bin.

Next $27(N-1)$ items of side $\frac{1}{3}$ arrive, again $OPT = N$. Note that only 26 such items can still fit into each initial bin. Therefore at least $\frac{27N-27-26N}{27} = \frac{N}{27} - 1$ new bins are created.

At this time, let γ be the number of new bins which contain at least four items. We consider two cases.

Case 1. If $\gamma \leqslant \frac{1097}{3K}N - 1$, and since for large enough N, the number of new bins is larger than this number, this means that the total number of new bins is actually at least $\gamma + \frac{N-27-27\gamma}{3} = \frac{N}{3} - 9 - 8\gamma \geqslant \frac{1677}{K}N - 1$, since the γ bins with at least four items, contain at most 27 items each, and the other new bins contain at most three items each.

In this case, all items of side $\frac{1}{3}$ leave, except for $\frac{1677}{K}N - 1$ of the new bins, where one item is left in each. OPT packs these items into $\frac{559}{9K}N$ bins.

Next, $N - \frac{559}{9K}N - 1$ items of side 1 arrive, which OPT can pack one per bin (and so does the algorithm). The total number of bins that the algorithm opens is $N + \frac{1677}{K}N - 1 + \frac{123\,704}{9K}N - 1 = \frac{18K+14\,534}{9K}N - 2 \approx 2.1169616N - 2$.

Case 2. Otherwise, $\gamma \geqslant \frac{1097}{3K}N$. All items of side $\frac{1}{3}$ leave, except for four items in each of $\frac{1097}{41\,421}N$ bins out of the bins that are not initial and contain at least four items each. OPT packs these items into $\frac{4388}{3^4K}N$ bins. Out of these bins, $\frac{1097}{3^4K}N$ bins contain items that will not leave later (one item per bin of the online algorithm), and the rest contain items that will leave in the last step.

The next step is the arrival of items of side $\frac{1}{2}$. The number of such items is $8(N - \frac{4388}{3^4K}N - 1)$ so that $OPT = N$ again. Out of these items, at most seven can be packed into each initial bin. According to Lemma 11, each other previously open bin can receive at most five such items. Therefore, there are at least $(8(N - \frac{4388}{4^3K}N - 1) - 7N - 5(\frac{1097}{41\,421}N))/8 \geqslant \frac{116\,896}{4^3K}N - 1$ new bins.

Next, all items of side $\frac{1}{3}$ leave but one per bin. All items of side $\frac{1}{2}$ leave but one per bin in $\frac{116\,896}{4^3K}N - 5$ bins. OPT packs these items of side $\frac{1}{2}$ in $\frac{14\,612}{4^3K}N$ bins. Finally, items of side 1 arrive. The number of such items is $N - 1 - \frac{1097}{4^3K}N - \frac{14\,612}{4^3K}N = N - \frac{15\,709}{4^3K}N - 1$. Therefore, the number of bins used by the algorithm is $2N - \frac{15\,709}{4^3K}N - 1 + \frac{1097}{3K}N - 1 + \frac{116\,896}{4^3K}N - 1 = 2N + \frac{130\,806}{4^3K}N - 3 \approx 2.1169616N - 3$.

Note that this bound is slightly higher than the one achieved by running a computer program, which gives the lower bound 2.11634. □

B.2. Box packing

Algorithm 3D-BoxesP. We perform an online partition of the boxes into several sub-sequences according to the size of each dimension. The sub-sequences are defined as follows.

For every box I with a size vector (a, b, c),

- $I \in S_1$ if $a \leqslant \frac{1}{3}$, $b \leqslant \frac{1}{3}$ and $c \leqslant \frac{1}{3}$.
- $I \in S_{2,1}$ if $a \leqslant \frac{1}{3}$, $b \leqslant \frac{1}{3}$ and $c > \frac{1}{3}$.
- $I \in S_{2,2}$ if $a \leqslant \frac{1}{3}$, $b > \frac{1}{3}$ and $c \leqslant \frac{1}{3}$.
- $I \in S_{2,3}$ if $a > \frac{1}{3}$, $b \leqslant \frac{1}{3}$ and $c \leqslant \frac{1}{3}$.
- $I \in S_{3,1}$ if $a \leqslant \frac{1}{3}$, $b > \frac{1}{3}$ and $c > \frac{1}{3}$.
- $I \in S_{3,2}$ if $a > \frac{1}{3}$, $b \leqslant \frac{1}{3}$ and $c > \frac{1}{3}$.
- $I \in S_{3,3}$ if $a > \frac{1}{3}$, $b > \frac{1}{3}$ and $c \leqslant \frac{1}{3}$.
- $I \in S_4$ if $a > \frac{1}{3}$, $b > \frac{1}{3}$ and $c > \frac{1}{3}$.

Sub-Algorithm A_{S_1}. Use algorithm 3-Dynamic-NFDH on S_1.

Sub-Algorithm $A_{S_{2,i}}$. Use algorithm 2-Dynamic-NFDH on $S_{2,i}$ for $i = 1, 2, 3$ taking into account only the dimensions with size at most $\frac{1}{3}$. Since for every item $S_{2,i}$ for $i = 1, 2, 3$ there are exactly two dimensions with such values, algorithm 2-Dynamic-NFDH can be used. The component of size greater than $\frac{1}{3}$ is ignored. Only one layer of items will be placed in that direction.

Sub-Algorithm $A_{S_{3,i}}$. Use algorithm First Fit for the dimension with size of at most $\frac{1}{3}$ on $S_{3,i}$ for $i = 1, 2, 3$. In the other two dimensions the items have size of at least $\frac{1}{3}$, place only one layer of items in these directions.

Sub-Algorithm A_{S_4}. Pack each new item into an empty bin.

Theorem 14. *Algorithm 3D-BoxesP has an asymptotic competitive ratio of at most* 35.346.

Proof.

Sub-Algorithm A_{S_1}. By Lemma 2 the asymptotic competitive ratio is at most 3.338.

Sub-Algorithm $A_{S_{2,i}}$. Since every item in $S_{2,i}$ for $i = 1, 2, 3$ has two dimensions with size at most $\frac{1}{3}$, all the items presented to the algorithm 2-Dynamic-NFDH have width and height of at most $\frac{1}{3}$. Therefore by Lemma 1 the asymptotic competitive ratio of that algorithm is at most 2.25. In the third dimension, where the boxes have size greater than $\frac{1}{3}$, we lose another factor of 2 due to the following reasoning. Since in this direction, every item has size of strictly larger than $\frac{1}{3}$, an optimal offline algorithm can utilize at most two layers of items at every point. Since in this algorithm only one layer is used, we lose the additional multiplicative factor of 2. Finally, since we have three sub-sequences of $S_{2,i}$ for $i = 1, 2, 3$, the asymptotic competitive ratio is multiplied by three. Therefore the asymptotic competitive ratio is at most $2.25 \cdot 2 \cdot 3 = 13.5$.

Sub-Algorithm $A_{S_{3,i}}$. By Lemma 9 the asymptotic competitive ratio of (dynamic) First Fit, when applied to items of size at most $\frac{1}{3}$, is 1.459. Similarly to the previous argument, we lose an additional factor of 2 for every dimension with value of strictly greater than $\frac{1}{3}$. We get for the three sub-sequences an asymptotic competitive ratio of $2 \cdot 2 \cdot 1.459 \cdot 3 = 5.836$.

Sub-Algorithm A_{S_4}. As argued before, only one item in S_4 can be packed into a single bin. Since both OPT and A_{S_4} can accommodate one item of S_4 per bin, the asymptotic competitive ratio is 1.

To complete the analysis we add up the asymptotic competitive ratios of the above sub-algorithms and get $3.338 + 13.5 + 17.508 + 1 = 35.346$. □

Theorem 15. *Any online algorithm for dynamic box packing in three dimensions has an asymptotic competitive ratio of at least* 4.85383.

Proof. Similarly to the lower bound of two dimensions, the presented items are now of three types. The first type of boxes have a variable size (a unit fraction in $A = \{\frac{1}{n}, \ldots, \frac{1}{2}, 1\}$) in the first dimension and size of 1 in the other two dimensions. The second type have size in A in the second dimension, and size of 1 on the other two dimensions. We define the third type similarly. Every phase contains these three types of items, except for the last one where all three types are identical. By the same technique, each phase can be broken down into 3 parts. In each part, items of the same size and type are given and depart. Using a computer we get for $n = 10\,000$, a lower bound of 4.85383. □

References

[1] N. Bansal, A. Caprara, M. Sviridenko, Improved approximation algorithms for multidimensional bin packing problems, in: Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 2006, pp. 697–708.

[2] N. Bansal, J.R. Correa, C. Kenyon, M. Sviridenko, Bin packing in multiple dimensions: Inapproximability results and approximation schemes, Mathematics of Operations Research 31 (1) (2006) 31–49.

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

[4] A. Caprara, Packing d-dimensional bins in d stages, Mathematics of Operations Research 33 (1) (2008) 203–215.

[5] W.T. Chan, T.W. Lam, P.W.H. Wong, Dynamic bin packing of unit fractions items, Theoretical Computer Science 409 (3) (2008) 521–529.

[6] W.T. Chan, P.W.H. Wong, F.C.C. Yung, On dynamic bin packing: an improved lower bound and resource augmentation analysis, Algorithmica 53 (2) (2009) 172–206.

[7] E.G. Coffman, M.R. Garey, D.S. Johnson, Dynamic bin packing, SIAM Journal on Computing 12 (1983) 227–258.

[8] E.G. Coffman, M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level oriented two-dimensional packing algorithms, SIAM Journal on Computing 9 (1980) 808–826.

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

[10] J. Csirik, A. van Vliet, An online algorithm for multidimensional bin packing, Operations Research Letters 13 (1993) 149–158.

[11] L. Epstein, R. van Stee, Optimal online bounded space multidimensional packing, in: Proc. of 15th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'04), 2004, pp. 207–216.

[12] L. Epstein, R. van Stee, Online square and cube packing, Acta Informatica 41 (9) (2005) 595–606.

[13] L. Epstein, R. van Stee, Optimal online algorithms for multidimensional packing problems, SIAM Journal on Computing 35 (2) (2005) 431–448.

[14] X. Han, D. Ye, Y. Zhou, Improved online hypercube packing, in: Proc. of the 4th International Workshop Approximation and Online Algorithms (WAOA 2006), 2006, pp. 226–239.
[15] D.S. Johnson, Near-optimal bin packing algorithms, PhD thesis, MIT, Cambridge, MA, 1973.
[16] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM Journal on Computing 3 (1974) 256–278.
[17] Y. Kohayakawa, F.K. Miyazawa, P. Raghavan, Y. Wakabayashi, Multidimensional cube packing, Algorithmica 40 (3) (2004) 173–187.
[18] C.C. Lee, D.T. Lee, A simple online bin packing algorithm, Journal of the ACM 32 (3) (1985) 562–572.
[19] K. Li, K.H. Cheng, A generalized harmonic algorithm for on-line multi-dimensional bin packing, Technical Report UH-CS-90-2, University of Houston, January 1990.
[20] A. Meir, L. Moser, On packing of squares and cubes, J. Combinatorial Theory Ser. A 5 (1968) 116–127.
[21] F.K. Miyazawa, Y. Wakabayashi, Cube packing, Theoretical Computer Science 297 (1–3) (2003) 355–366.
[22] S.S. Seiden, R. van Stee, New bounds for multi-dimensional packing, Algorithmica 36 (3) (2003) 261–293.
[23] J.D. Ullman, The performance of a memory allocation algorithm, Technical Report 100, Princeton University, Princeton, NJ, 1971.
[24] R. van Stee, An approximation algorithm for square packing, Operations Research Letters 32 (2004) 535–539.