Online Square and Cube Packing[*]

Leah Epstein[†] Rob van Stee[‡]

Abstract

In online square packing, squares of different sizes arrive online and need to be packed into unit squares which are called bins. The goal is to minimize the number of bins used. Online cube packing is defined analogously. We show an upper bound of 2.2697 and a lower bound of 1.6406 for online square packing, and an upper bound of 2.9421 and a lower bound of 1.6680 for online cube packing. The upper bound for squares can be further reduced to 2.24437 using a computer proof. These results improve on the previously known results for the two problems. We also show improved lower bounds for higher dimensions.

1 Introduction

In this paper, we consider the problems of online square and cube packing. We first define the simplest problem. In square packing, we receive a sequence σ of *squares* $p_1, p_2, \ldots, p_n$. Each square p has a fixed *size* $s(p)$, which is the length of its sides. We have an infinite number of *bins*, each of which is a unit square. Each item must be assigned to a bin and a position $(x(p), y(p))$, where $0 \leq x(p)$, $0 \leq y(p)$, $x(p) + s(p) \leq 1$ and $y(p) + s(p) \leq 1$. Further, the positions must be assigned in such a way that no two items in the same bin overlap. A bin is *empty* if no item is assigned to it, otherwise it is *used*. The goal is to minimize the number of bins used. The items arrive online; each item must be assigned in turn, without knowledge of the next items. Note that for $d = 1$, the square packing problem reduces to exactly the classic online bin packing problem.

The cube packing problem is defined in an analogous fashion. We also consider the problem of packing hypercubes in more than three dimensions in this paper. The offline version of

[*]Preliminary versions of different parts of this paper appeared in Proc. Symp. Discr. Alg. 2004 (SODA 2004) and Proc. Eur. Symp. Alg. 2004 (ESA 2004).

[†]Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il. Research supported by Israel Science Foundation (grant no. 250/01).

[‡]Fakultät für Informatik, Universität Karlsruhe, Germany. vanstee@ira.uka.de. Work performed while the author was at Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands. Research supported by the Netherlands Organization for Scientific Research (NWO), project number SION 612-061-000.

square packing is NP-hard [8]. We believe that this also holds for (hyper-)cube packing, but this remains an open question.

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

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

Let $\mathcal{O}$ be some class of bin packing algorithms (for instance online algorithms). The *optimal asymptotic performance* ratio for $\mathcal{O}$ is defined to be $\mathcal{R}^{\infty}_{\mathcal{O}} = \inf_{\mathcal{A}\in\mathcal{O}} \mathcal{R}^{\infty}_{\mathcal{A}}$. Given $\mathcal{O}$, our goal is to find an algorithm with asymptotic performance ratio close to $\mathcal{R}^{\infty}_{\mathcal{O}}$.

Previous Results: The classic online bin packing problem was first investigated by Ullman [14]. He showed that the First Fit algorithm has performance ratio $\frac{17}{10}$. This result was published in [6]. Johnson [5] showed that the Next Fit algorithm has performance ratio 2. Yao showed that Revised First Fit has performance ratio $\frac{5}{3}$.

Define $u_1 = 2, u_{i+1} = u_i(u_i - 1) + 1$, and $h_\infty = \sum_{i=1}^{\infty} \frac{1}{u_i - 1} \approx 1.69103$. Lee and Lee showed that the Harmonic algorithm, which uses bounded space (i.e. can have only a constant number of active bins at any time), achieves a performance ratio arbitrarily close to h_∞ [7]. They further showed that no bounded space online algorithm achieves a performance ratio less than h_∞ [7]. In addition, they developed the Refined Harmonic algorithm, which they showed to have a performance ratio of $\frac{273}{228} < 1.63597$. The next improvements were Modified Harmonic and Modified Harmonic 2. Ramanan, Brown, Lee and Lee showed that these algorithms have performance ratios of $\frac{538}{333} < 1.61562$ and $\frac{239091}{148304} < 1.61217$, respectively [11]. Currently, the best known upper bound is 1.58889 due to Seiden [12]. As for lower bounds, Yao showed that no online algorithm has performance ratio less than $\frac{3}{2}$ [16]. Brown and Liang independently improved this lower bound to 1.53635 [1, 9]. The lower bound currently stands at 1.54014, due to van Vliet [15].

Coppersmith and Raghavan [2] showed an upper bound of $43/16 = 2.6875$ for online square packing, and an upper bound of 6.25 for online cube packing. They also show a lower bound of $4/3$ for any dimension $d \geq 2$. The upper bound for square packing was improved to $395/162 < 2.43828$ by Seiden and van Stee [13]. They also presented a lower bound of 1.6217. For $d = 3$, Miyazawa and Wakabayashi [10] showed an upper bound of 3.954. A lower bound of 1.60185 was given by [13]. The same paper gave lower bounds for higher dimensions as well, but they all have lower values than the bound for $d = 3$.

Epstein and van Stee [4] presented an *optimal* bounded space algorithm for hypercube packing for any dimension $d \geq 2$. For $d = 2$, their algorithm has an asymptotic performance ratio of at most 2.3722 and at least 2.35656, while for $d = 3$, the asymptotic performance ratio is in the interval $(2.94457, 3.0672]$.

Our Results: In this paper, we present improved results for online square packing:

- We present a new unbounded space algorithm for online square packing with an asymptotic performance ratio of at most 2.2697, which cannot be attained by a bounded space algorithm. This algorithm is a slight modification of the unbounded space algorithm in [4], which attained an asymptotic performance ratio of at most 2.2709. Using a computer program, we can show that another modification has the performance ratio of at most 2.24437.

- We present a new unbounded space algorithm for online cube packing with an asymptotic performance ratio of at most 2.9421, which again is better than the best possible bounded space algorithm.

- We give new lower bounds for unbounded space hypercube packing. The lower bound for $d = 2$ is 1.6406, while for $d = 3$ it is 1.6680. A preliminary version of this proof appeared in [3].

We use the weighting functions technique. As in [3, 4], unlike in [13], we define weighting functions directly for multidimensional algorithms, without using one-dimensional algorithms as subroutines.

2 Harmonic type algorithms

In this section we discuss the important one-dimensional Harmonic algorithm [7] and possible variations on it. In the next sections we adapt these algorithms to the two- and three-dimensional cases.

The fundamental idea of these algorithms is to first classify items by size, and then pack an item according to its class (as opposed to letting the exact size influence packing decisions).

For the classification of items, we need to partition the interval $(0, 1]$ into subintervals. The standard Harmonic algorithm uses $n - 1$ subintervals of the form $(1/(i + 1), 1/i]$ for $i = 1, \ldots, n - 1$ and one final subinterval $(0, 1/n]$. Each bin will contain only items from one subinterval (type). Items in subinterval i are packed i to a bin for $i = 1, \ldots, n - 1$ and the items in interval n are packed in bins using Next Fit(i.e. a greedy algorithm that opens a new active bin whenever an item does not fit into the current active bin, and never uses the previous bins).

A disadvantage of Harmonic is that items of type 1, that is, the items larger than 1/2, are packed one per bin, possibly wasting a lot of space in each single bin. To avoid this large waste of space, later algorithms used two extra interval endpoints, of the form $\Delta \in [1/3, 1/2)$ and $1 - \Delta > 1/2$. Then, some small items can be combined in one bin together with an item of size $\in (1/2, 1-\Delta]$. Items larger than $1-\Delta$ are still packed one per bin as in Harmonic. These algorithms furthermore use parameters α^i $(i = 3, \ldots, n)$ which represent the fraction of bins allocated to type i where the algorithm will reserve space for items $\in (1/2, 1-\Delta]$. The remaining bins with items of type i still contain i items per bin. Clearly, such adaptations are no longer bounded space.

The first such modification of Harmonic is Refined Harmonic [7] which only combines some items in the interval $[1/3, \Delta)$ with items in $[1/2, 1-\Delta)$. All other intervals are packed separately. The value of n used in this algorithm was 20.

The algorithm Modified Harmonic (MH) is defined by $n = 38$ and $\Delta = 265/684$. Further, $\alpha^2 = \frac{1}{9}$, $\alpha^3 = \frac{1}{12}$, $\alpha^4 = \alpha^5 = 0$, $\alpha^i = \frac{37-i}{37(i+1)}$, for $6 \leq i \leq 36$ and $\alpha^{37} = \alpha^{38} = 0$. The results of [11] imply that the asymptotic performance ratio of MH is at most $\frac{538}{333} < 1.61562$.

The bounded space algorithms in [4] are extensions of Harmonic to the multidimensional case. In this paper we extend the Modified Harmonic algorithm to the two- and three-dimensional cases. We present algorithms that use less intervals. The reason for this is that the analysis in more than one dimension is substantially harder. The number of relatively large items that fit in a bin grows with the dimension, and there may be many distinct ways to pack the same set of squares in a bin.

3 Online square packing

In this section we define a two-dimensional version of Modified Harmonic that we call MH^2.

It uses seven item types (intervals in $(0, 1]$), denoted by $1, 1a, 2, 2a, 3, 4, 5$ in order of decreasing size. The algorithm uses a variable $\Delta \in (1/3, 0.385)$. The upper bound for type i $(i = 1, \ldots, 5)$ is $1/i$. The upper bound for type $1a$ is $1 - \Delta$, and for $2a$ it is Δ.

MH^2 packs all items of size at most 1/5 (type 5) using the algorithm from [4] for small items. That is, we divide the items into 5 subtypes, and for each item of subtype i we call the function AssignSmall(i) (we take $M = 5$).

Items of type $2a$ are partially put three to a bin, in such a way that a type $1a$ item could be put in the same bin with them, and partially put four to a bin, in the four corners. These items are colored red and blue, respectively. Similarly, red type 3 items are put five to a bin, and blue type 3 items are put nine to a bin. We use a parameter α to denote the fraction of type $2a$ items that are colored red, and β denotes the fraction of type 3 items that are colored red. The values of α, β and Δ will be chosen in such a way that the asymptotic performance ratio of MH^2 is minimized. We will find $\alpha \approx 0.1752$, $\beta = 31/256 \approx 0.1211$ and $\Delta \approx 0.3730$.

Items of types $1, 2, 3$, and 4 are packed in bins that only contain items of one type. Full bins contain $1, 4, 9$, and 16 items, respectively. By the proof in [4], full bins containing items of type 5 have occupied area of at least $(M^d - 1)/(M + 1)^d = 24/36 = 2/3$.

We define two sets of weights for the items. This is a weighting system, which is a special case of general weighting systems defined in [12]. The proof of the following theorem follows directly from that paper.

Theorem 1 *Denote by w_i the maximum amount of weight that can be packed into a single bin according to measure W_i ($i = 1, 2$). Then the asymptotic performance ratio of* MH^2 *is upper bounded by* $\max(w_1, w_2)$.

We give an overview of the types, weights and expansions in Table 1. The first column states the type of an item, i.e. the name of the interval it belongs to. The second column states the maximum size of any item in this interval (the lower bound on the size follows from the maximum size in the next interval). The next two columns are the weights assigned to an item of each interval. Finally, the last two columns are the expansions. The expansion of an item is the ratio between its weight and its minimum area. This is useful since in order to use the theorem we need to compute the maximal ratio of weight to area in a single bin.

We give a short intuitive explanation of the weight functions and Theorem 1. Consider the final packing created by MH^2 on some input σ. In this final packing, let x be the number of bins containing red items, let y be the number of type 1a items, and let z be the number of bins containing blue items of type other than 1a. The total number of bins is just $\max\{x, y\} + z = \max\{x + z, y + z\}$. We have chosen our weighting functions so that $\sum_{p\in\sigma} W_1(p) = y + z + O(1)$ and $\sum_{p\in\sigma} W_2(p) = x + z + O(1)$. In both W_1 and W_2, the weight of a blue item of type other than 1a is just the fraction of a bin that it occupies. W_1 counts type 1a items, but ignores red items. W_2 ignores type 1a items, but counts bins containing red items. For a formal proof, we refer the reader to [12].

The expansion of type 5 items (small items) is $3/2$. To simplify the calculations, we will assume that it is instead $25/16$. This can only make the asymptotic performance ratio worse, so we will calculate an upper bound for the asymptotic performance ratio. Using this value enables us to ignore the items of type 4, since they also have expansion $25/16$: we will assume that after taking some items from types $1, 1a, 2, 2a$, and 3 we can fill up the rest of the bin entirely with items of expansion $25/16$ (this is the worst case).

To find a bin with maximal weight, we need to try all possible combinations of items of

Type	Max. size	W_1		W_2		E_1		E_2	
1	1	1	1	1	1	$\frac{1}{(1-\Delta)^2}$	2.5438	$\frac{1}{(1-\Delta)^2}$	2.5438
$1a$	$1-\Delta$	1	1	0	0	4	4	0	0
2	1/2	1/4	0.25	1/4	0.25	$1/(4\Delta^2)$	1.7968	$1/(4\Delta^2)$	1.7968
$2a$	Δ	$\frac{1-\alpha}{4}$	0.2062	$\frac{3+\alpha}{12}$	0.2646	$\frac{9}{4}(1-\alpha)$	1.8559	$\frac{9+3\alpha}{4}$	2.3814
3	1/3	$\frac{1-\beta}{9}$	0.0977	$\frac{5+4\beta}{45}$	0.1219	$\frac{16}{9}(1-\beta)$	1.5625	$\frac{16(5+4\beta)}{45}$	1.95
4	1/4	1/16	0.0625	1/16	0.0625	25/16	1.5625	25/16	1.5625
5	1/5	$\frac{3}{2}x$	$1.5x$	$\frac{3}{2}x$	$1.5x$	*25/16*	*1.5625*	*25/16*	*1.5625*

Table 1: Item weights and expansions for MH^2. The decimal values are found by taking $\alpha = \frac{25}{36}(9\Delta^2 - 1)$, $\beta = \frac{31}{256}$ and $\Delta = -1/3 + \frac{\sqrt{404130}}{900} \approx 0.3730$. The expansion for type 5 is in reality 3/2, but we use the higher value 31/256 to simplify the calculations. See the text.

types $1, 1a, 2, 2a$, and 3 and fill up the rest with "sand" (small items). The weight is maximized by minimizing the size of all the large items that we use (since the weight they contribute does not depend on their size, and it maximizes the area available for other large items and for small items). We take $\alpha = \frac{25}{36}(9\Delta^2 - 1)$. This choice of α ensures that if we have a bin with a type 2 item (of minimal size), and we replace it with a type $2a$ item (of minimal size) and fill the remaining area with sand, the total weight is unchanged when we use measure W_1: we have $1/4 = \frac{1-\alpha}{4} + \frac{25}{16}(\Delta^2 - 1/9)$. Moreover, we take $\beta = 31/256$ such that type 3 items are equivalent to sand under measure W_1.

We use the following lemma from [13].

Lemma 1 *If a square of size strictly greater than* $1/2$ *is packed in the unit square then at most 5 squares of size strictly greater than* $1/4$ *can be packed with it.*

Theorem 2 MH^2 *maintains an asymptotic performance ratio of at most 2.26967.*

Proof We look for a bin with maximal weight.

Case 1. First of all, suppose that there is no item of type 1 or $1a$ in the bin. For the remaining items, $W_1(p) \le W_2(p)$ for all items p. Thus we only need to consider W_2. The type with the highest expansion is type $2a$, which fits at most four times in a bin. All other types have expansion smaller than 2 by our choice of Δ, so the total weight of the bin in this case is bounded by $4 \cdot \frac{3+\alpha}{12} + 2(1 - 4 \cdot \frac{1}{9}) < 2.17$.

Suppose there is an item of type 1 or 1a. By Lemma 1, at most 5 items of type $2, 2a$, or 3 can be packed with such an item. As usual we assume that the rest of the bin is filled with items of expansion 25/16. Since the items of type $2, 2a$ and 3 have expansion greater than

25/16, the weight is maximized by taking five of them.

Case 2. Suppose there is an item of type 1. Then there is no item of type $1a$; for all other items, $W_2(p) \geq W_1(p)$, so we only need to consider $W_2(p)$. In this case items of type 2 do not fit in the bin. If we place i items of type $2a$ in the bin, this leaves room for at most $5 - i$ items of type 3. We assume that $5 - i$ such items can always be placed. (This is a worst-case assumption.) Since the weight of type $2a$ items is higher than that of type 3 items, and the total number of type $2a$ items and type 3 items is assumed to be independent of the number of type $2a$ items, we maximize the number of type $2a$ items. However, at most 4 items of size strictly greater than 1/3 fit in one bin, so at most 3 items of type $2a$ can be placed in this bin together with the item of type 1. This leaves room for two items of type 3.

The total weight is at most $1 + 3 \cdot \frac{3+\alpha}{12} + 2 \cdot \frac{5+4\beta}{45} + \frac{25}{16}(1 - (1-\Delta)^2 - 3 \cdot \frac{1}{9} - 2 \cdot \frac{1}{16})$.

Case 3. Now suppose there is an item of type $1a$. Then the weight of the bin is maximal if we use W_1 (using W_2, the total weight is now lower than in Case 1).

There are at most 5 items of type $2, 2a$, or 3. By the remarks above Lemma 1, we do not need to distinguish between bins that only differ in that one of them has a type 2 item where the other has a type 2a item and more sand. Moreover, we can ignore items of type 3 since they are equivalent to sand. We therefore take 3 items of type 2 and fill the rest of the bin with sand.

The total weight is at most $1 + 3 \cdot \frac{1}{4} + \frac{25}{16}(1 - \frac{1}{4} - 3 \cdot \Delta^2)$.

Taking $\Delta = -1/3 + \frac{1}{900}\sqrt{404130} \approx 0.3730$, the weights in Cases 2 and 3 become less than 2.269661. This is an upper bound for the asymptotic performance ratio of MH^2. □

Using a computer program, it is possible to enumerate all dominant patterns (that cannot be augmented by an additional item of one of the types that the bin contains), also when smaller items are involved. We can take $M = 6$ and pack items of size in $(1/6, 1/5]$ in separate bins as well (25 per bin). In this case, the expansion of small items (i.e. the items of size at most 1/6) becomes only 7/5. We take $\alpha = \frac{28}{5}(\Delta^2 - \frac{1}{9}) \approx 0.153297$ and $\beta = \frac{154}{1000}$. In this way, again a type 2 item is equivalent to a type $2a$ item plus sand, and moreover a type 3 item is equivalent to a type 4 item plus sand (all under W_1). By enumerating all patterns and calculating the weights (both in measure W_1 and W_2), we find that by taking $\Delta = 0.372137$, this modified algorithm has an asymptotic performance ratio of at most 2.244361.

The source code for this C++ program can be downloaded from the second author's website, http://i10www.ira.uka.de/vanstee/program/.

4 Online cube packing

In this section we define a three-dimensional version of Modified Harmonic that we call MH^3. This algorithm is a three-dimensional version of MH^2 that we discussed in the previous section. It has the same structure and also has $M = 5$, but it uses different values for α, β and Δ. The analysis is by weighting systems and according to Theorem 1.

Red items of type $2a$ are now placed seven to a bin (in seven corners), and red items of type 3 are placed 19 to a bin. For the red type 3 items, we divide the bin into 27 cubes of size $1/3$ and leave a block of 2 by 2 by 2 cubes empty. In this block, an item of type $1a$ can be placed. The remaining 19 cubes will all contain one item of type 3.

As before, the values of α, β and Δ will be chosen in such a way that the asymptotic performance ratio of MH^3 is minimized. We will find $\alpha \approx 0.154880$, $\beta = 721/4096 \approx 0.176025$ and $\Delta \approx 0.360753$. For the moment we will only use $\alpha < 1/4$ and $\Delta > 0.36$. We give an overview of the types, weights and expansions in Table 2.

Type	Max. size	W_1		W_2		E_1		E_2	
1	1	1	1	1	1	$\frac{1}{(1-\Delta)^3}$	3.8282	$\frac{1}{(1-\Delta)^3}$	3.8282
$1a$	$1-\Delta$	1	1	0	0	8	8	0	0
2	$1/2$	$1/8$	0.125	$1/8$	0.125	$1/(8\Delta^3)$	2.6624	$1/(8\Delta^3)$	2.6624
$2a$	Δ	$\frac{1-\alpha}{8}$	0.1056	$\frac{7+\alpha}{56}$	0.1278	$\frac{27}{8}(1-\alpha)$	2.8523	$\frac{27(7+\alpha)}{56}$	3.4497
3	$1/3$	$\frac{1-\beta}{27}$	0.0305	$\frac{19+8\beta}{513}$	0.0398	$\frac{64}{27}(1-\beta)$	1.9531	$\frac{64(19+8\beta)}{513}$	2.5461
4	$1/4$	$1/64$	0.0156	$1/64$	0.0156	$125/64$	1.9531	$125/64$	1.9531
5	$1/5$	$\frac{54}{31}x$	$1.7419x$	$\frac{54}{31}x$	$1.7419x$	*125/64*	*1.9531*	*125/64*	*1.9531*

Table 2: Item weights and expansions for MH^3

By the proof in [4], full bins containing items of type 5 have occupied area of at least $(M^d - 1)/(M + 1)^d = 31/54$. This explains that the expansion of type 5 items (small items) is $54/31$. To simplify the calculations, similarly to in Section 3 we assume that it is instead $125/64$, so that we can again ignore items of type 4. Also analogously to before, we take $\alpha = \frac{125}{8}(\Delta^3 - 1/27)$. (See Section 3 for a motivation.) Moreover, we take $\beta = 1 - 3^3 \cdot 5^3/4^6 \approx 0.176025$, so that type 3 items are equivalent to sand under measure W_1.

Lemma 2 *If a cube of size strictly greater than* $1/2$ *is packed in the unit cube, then at most 19 cubes of size strictly greater than* $1/4$ *can be packed with it.*

Proof Note that inside a cube which is strictly greater than $1/2$, we can place 8 cubes of size strictly greater than $1/4$ by cutting halfway along all three coordinate axes. Thus, any packing of the unit cube that contains a cube which is strictly greater than 1/2 can be replaced by a packing without that cube and with 8 more cubes that are strictly greater than 1/4.

Since a unit cube cannot contain more than 27 cubes that are strictly larger than 1/4 by Claim 4 in [4], the lemma follows. □

Theorem 3 MH^3 *maintains a asymptotic performance ratio of at most 2.9421.*

Proof As before, we look for a bin with maximal weight.

Case 1. First of all, suppose that there is no item of type 1 or 1a in the bin. For the remaining items, $W_1(p) \leq W_2(p)$ for all items p. Thus we only need to consider W_2. The type with the highest expansion is type 2a, which fits at most 8 times in a bin. All other types have expansion at most 2.68 by our choice of Δ and β, so the total weight of the bin in this case is at most $8 \cdot \frac{7+\alpha}{56} + 2.68 \cdot (1 - 8 \cdot \frac{1}{27}) < 2.94$.

Next, suppose there is an item of type 1 or 1a. By Lemma 2, at most 19 items of type $2, 2a$, or 3 can be packed with them. As usual we assume that the rest of the bin is filled with items of expansion 125/64. Since the items of type $2, 2a$ and 3 have expansion greater than 125/64, the weight is maximized by taking 19 of them.

Case 2. Suppose there is an item of type 1. Then there is no item of type 1a; for all other items, $W_2(p) \geq W_1(p)$, so we only need to consider $W_2(p)$. In this case items of type 2 do not fit in the bin. If we place i items of type 2a in the bin, this leaves room for at most $19 - i$ items of type 3. We assume that $19 - i$ such items can always be placed. (This is a worst-case assumption.) Since the weight of type 2a items is higher than that of type 3 items, and the total number of type 2a items and type 3 items is assumed to be independent of the number of type 2a items, we maximize the number of type 2a items. However, at most 8 items of size strictly greater than 1/3 fit in one bin, so at most 7 items of type 2a can be placed in this bin together with the item of type 1. This leaves room for 12 items of type 3.

The total weight is at most $1 + 7 \cdot \frac{(7+\alpha)}{56} + 12 \cdot \frac{19+8\beta}{513} + \frac{125}{64}(1 - (1-\Delta)^3 - 7 \cdot \frac{1}{27} - 12 \cdot \frac{1}{64})$.

Case 3. Now suppose there is an item of type 1a. Then the weight of the bin is maximal if we use W_1 (using W_2, the total weight is now lower than in Case 1).

There are at most 19 items of type $2, 2a$, or 3. By the remarks above Lemma 2, we do not need to distinguish between bins that only differ in that one of them has a type 2 item where the other has a type 2a item and more sand. Moreover, we can ignore items of types 3 and 4 since they are equivalent to sand. We therefore take seven items of type 2 and fill the rest of the bin with sand.

The total weight is at most $1 + 7 \cdot \frac{1}{8} + \frac{125}{64}(1 - \frac{1}{8} - 7 \cdot \Delta^3)$.

Taking $\Delta = 0.360753$ and $\alpha = 0.15488$, the weights in Cases 2 and 3 become less than 2.9421. This is an upper bound for the asymptotic performance ratio of MH^3. □

5 Lower bounds

We construct a sequence of items to prove a general lower bound for hypercube packing in d dimensions. In this problem, all items to be packed are hypercubes. Take an integer $\ell > 1$. Let $\varepsilon > 0$ be a number smaller than $1/2^\ell - 1/(2^\ell + 1)$. The input sequence is defined as follows. We use a large integer N. Let

$$x_i = (2^{\ell+1-i} - 1)^d - (2^{\ell+1-i} - 2)^d \qquad i = 1, \dots, \ell$$
$$x_0 = 2^{\ell d} - (2^\ell - 1)^d$$

In step 0, we let Nx_0 items of size $s_0 = (1+\varepsilon)/(2^\ell + 1)$ arrive. In step $i = 1, \dots, \ell$, we let Nx_i items of size $s_i = (1+\varepsilon)/2^{\ell+1-i}$ arrive. For $i = 1, \dots, \ell - 1$, item size s_i in this sequence divides all the item sizes $s_{i+1}, \dots, s_\ell$. Furthermore, $\sum_{i=0}^{\ell} s_i = (1+\varepsilon)(1 - 1/2^\ell + 1/(2^\ell + 1)) < 1$ by our choice of ε.

The online algorithm receives steps $0, \dots, k$ of this input sequence for some (unknown) $0 \le k \le \ell$.

A pattern is a multiset of items that fits (in some way) in a unit bin. A pattern is *dominant* if when we increase the number of items of the smallest size in that pattern, the resulting multiset no longer fits in a unit bin. A pattern is *greedy* if the largest item in it appears as many times as it can fit in a bin, and this also holds for all smaller items, each time taking the larger items that are in the pattern into account. Note that not all possible sizes of items need to be present in the bin.

For a pattern P of items that all have sizes in $\{s_0, \dots, s_\ell\}$, denote the number of items of size s_i by P_i.

Lemma 3 *For any pattern P for items with sizes in $\{s_0, \dots, s_\ell\}$,*

$$P_i \le (2^{\ell+1-i} - 1)^d - \sum_{j=i+1}^{\ell} 2^{(j-i)d} P_j \quad i = 1, \dots, \ell, \qquad P_0 \le 2^{\ell d} - \sum_{j=1}^{\ell} 2^{(j-1)d} P_j \qquad (1)$$

Proof Suppose (1) does not hold for some i, and consider the smallest i for which it does not hold. Consider an item of size s_j with $j > i$ that appears in P. If there is no such j, we have a contradiction, since at most $(2^{\ell+1-i} - 1)^d$ items of size s_i fit in the unit hypercube for $i = 1, \dots, \ell$, or at most $2^{\ell d}$ items of size s_0. This follows from Claim 4 in [4].

First suppose $i > 0$. If we replace an item of size s_j with $2^{(j-i)d}$ items of size s_i, the resulting pattern is still feasible: all the new size s_i items can be placed inside the hypercube that this size s_j item has vacated.

We can do this for all items of size s_j, $j > i$ that appear in the pattern. This results in a pattern with only items of size s_i or smaller. Since every size s_j item is replaced by $2^{(j-i)d}$ items of size s_i, the final pattern has more than $(2^{\ell+1-i} - 1)^d$ items of size s_i, a contradiction.

Now suppose $i = 0$. In this case $\lfloor (2^\ell + 1)/2^{\ell+1-j} \rfloor = 2^{j-1}$ items of size s_0 fit in a hypercube of size s_j, and the proof continues analogously. □

We define a *canonical packing* for dominant patterns. We create a grid in the bin as follows. One corner of the bin is designated as the origin O. We assign a coordinate system to the bin, where each positive axis is along some edge of the bin. The grid points are those points inside the bin that have all coordinates of the form $m_1(1+\varepsilon)/2^{m_2}$ for $m_1, m_2 \in \mathbb{N} \cup \{0\}$.

We pack the items in order of decreasing size. Each item of size s_i $(i = 1, \ldots, \ell)$ is placed at the available grid point that has all coordinates smaller than $1 - s_i$, all coordinates equal to a multiple of s_i and is closest to the origin. So the first item is placed in the origin. Each item of size s_0 is placed at the available grid point that has all coordinates equal to a multiple of s_1 (and not s_0) and is closest to the origin. Note that for these items we also use grid points with some coordinates equal to $(2^\ell - 1)(1+\varepsilon)/2^\ell$, unlike for items of size s_1. This is feasible because $(2^\ell - 1)(1+\varepsilon)/2^\ell + (1+\varepsilon)/(2^\ell + 1) = (1+\varepsilon)(1 - 1/2^\ell + 1/(2^\ell + 1)) < 1$.

In each step i we can place a number of items which is equal to the upper bound in Lemma 3. For $i = 1, \ldots, \ell$, this is because a larger item of size s_j takes away exactly $2^{(j-i)d}$ grid points that are multiples of s_i, and originally there are $(2^{\ell+1-i} - 1)^d$ grid points of this form that have all coordinates smaller than $1 - s_i$. For $i = 0$, $2^{(j-1)d}$ grid points that are multiples of s_1 are taken away by an item of size s_j, from an original supply of $2^{\ell d}$. This shows that all patterns can indeed be packed in canonical form.

Lemma 4 *For this set of item sizes, any dominant pattern that is not greedy is a convex combination of dominant patterns that are greedy.*

Proof We use induction to construct a convex combination of greedy patterns for a given dominant pattern P. The induction hypothesis is as follows: the vector that describes the numbers of items of the t smallest types which appear in the pattern is a convex combination of greedy vectors for these types. Call such a pattern t-greedy.

The base case is $t = 1$. We consider the items of the smallest type that occurs in P. Since P is dominant, for this type we have that as many items as possible appear in P, given the larger items. Thus P is 1-greedy.

We now prove the induction step. Suppose that in P, items of type i appear fewer times than they could, given the larger items. Moreover, P contains items of some smaller type. Let i' be the largest smaller type in P. By induction, we only need to consider patterns in which all the items of type less than i that appear, appear as many times as possible, starting with items of type i'. (All other patterns are convex combinations of such patterns.)

We define two patterns P' and P'' such that P is a convex combination of them. First suppose $i' > 0$. P' is defined as follows: modify P by removing all items i and adding the largest smaller item that appears in P, of type i', $2^{(i-i')d}$ times per each item i. When creating

P', we thus add the maximum amount of items of type i' that can fit for each removed item of type i. P is greedy with respect to all smaller items, and $s_{i'}$ divides s_i. Therefore the multiset P' defined in this way is a pattern, and is $(t+1)$-greedy.

P'' on the other hand is created by adding items of phase i to P and removing items of type i'. In particular, in the canonical packing for P, at each grid point for type i that is not removed due to a higher-type item, we place an item of size s_i and remove all items that overlap with this item. Since all items smaller than s_i appear as many times as possible given the larger items, all the removed items are of the next smaller type i' that appear in P. This holds because the items are packed in order of decreasing size, so this area will certainly be filled with items of type i' if the item of size i is not there.

In P'', the number of items of type i is now maximized given items of higher types. Only type i' items are removed, and only enough to make room for type i, so type i' remains greedy. Thus P'' is $(t+1)$-greedy. Each time that we add an item i, we remove exactly $2^{(i-i')d}$ items of type i'. So by adding an item i in creating P'', we remove exactly the same number of items of type i' as we add when we remove an item i while creating P'. Therefore, P is a convex combination of P' and P'', and we are done.

Now suppose $i' = 0$. (This is a special case of the induction step for $t+1=2$.) In this case, in P' each item of type i is replaced by $2^{(i-1)d}$ items of type 0. In the canonical packing, this is exactly the number of type 1 grid points that become available when removing an item of type i. Thus in P', the number of type 0 items is maximal (since it is sufficient to use type 1 grid points for them), and P' is 2-greedy.

Similarly, it can be seen that to create P'', we need to remove $2^{(i-1)d}$ items of type 0 in the canonical packing in order to place each item of type i. Then P'' is 2-greedy, and again P is a convex combination of P' and P''. $\square$

We can now formulate a linear program to lower bound the asymptotic performance ratio of any unbounded space online algorithm as in [13]. We will need the offline cost to pack any prefix of the full sequence. This is calculated as follows.

To pack the items of the largest type k, which have a size of $(1+\varepsilon)2^{k-\ell-1}$, we need $Nx_k/(2^{\ell+1-k}-1)^d$ bins because there are Nx_k such items. These are all packed identically: using a canonical packing, we pack as many items of smaller types in bins with these items as possible. (Thus we use a greedy pattern.) Some items of types $1,\ldots,k-1$ still remain to be packed. It is straightforward to calculate the number of items of type $k-1$ that still need to be packed, and how many bins this takes. We continue in the same manner until all items are packed.

Solving this linear program for $\ell = 11$ and several values of d gives us the following results.

d	1	2	3	4	5	6	7	8	9	10
old	1.5401	1.6217	1.60185	1.5569	(Bounds here were below 1.5569)					
new	(1.5)	1.6406	1.6680	1.6775	1.6840	1.6887	1.6920	1.6943	1.6959	1.6973

6 Conclusions

The gaps between the upper and lower bounds remain disappointingly large for these problems. For square packing, the fundamental problem seems to be that it is (NP-)hard to determine whether a given set of squares fits into a bin or not. However, in the analysis of our algorithm MH^2, it is noticeable that the first 1.75 of the upper bound is "caused by" items larger than 1/3, which in theory are easy to pack. Obviously 1.75 is already well above the lower bound of 1.64, yet we see no way to significantly reduce the gap. This is a challenging open problem.

References

[1] Donna J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Sci. Lab., Urbana, Illinois, 1979.

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

[3] Leah Epstein and Rob van Stee. On variable-sized multidimensional packing. In *Algorithms - ESA 2004, Proceedings Twelfth Annual European Symposium*, volume 3221 of *Lecture Notes in Computer Science*, pages 287–298. Springer, 2004.

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

[5] David S. Johnson. Fast algorithms for bin packing. *Journal Computer Systems Science*, 8:272–314, 1974.

[6] David 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.

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

[8] J.Y.T. Leung, T.W. Lam, C.S. Wong, G.H. Young, and F.Y.L. Chin. Packing squares into a square. *Journal on Parallel and Distributed Computing*, 10:271–275, 1990.

[9] F. M. Liang. A lower bound for online bin packing. *Information Processing Letters*, 10:76–79, 1980.

[10] Flavio Keidi Miyazawa and Yoshiko Wakabayashi. Cube packing. *Theoretical Computer Science*, 297(1-3):355–366, 2003.

[11] 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.

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

[13] Steve S. Seiden and Rob van Stee. New bounds for multi-dimensional packing. *Algorithmica*, 36(3):261–293, 2003.

[14] Jeffrey D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.

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

[16] A. C. C. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.