

Improved Online Hypercube Packing

Xin Han¹ Deshi Ye² Yong Zhou³

¹ School of Informatics, Kyoto University, Kyoto 606-8501, Japan

hanxin@kuis.kyoto-u.ac.jp

² Department of Computer Science, The University of Hong Kong, Hong Kong

yedeshi@cs.hku.hk

³ Hokkaido Univerisity, Japan

zhou@castor.sci.hokudai.ac.jp

Abstract

In this paper, we study online multidimensional bin packing problem when all items are hypercubes. Based on the techniques in one dimensional bin packing algorithm Super Harmonic by Seiden, we give a new framework for online hypercube packing and obtain new upper bounds of asymptotic competitive ratios. For square packing, we get an upper bound of 2.1439, which is better than 2.24437. For cube packing, we also give a new upper bound 2.6852 which is better than 2.9421 by Epstein and van Stee.

1 Introduction

The classical one-dimensional Bin Packing is one of the oldest and most well-studied problems in computer science [2], [5]. In the early 1970's it was one of the first combinatorial optimization problems for which the idea of worst-case performance guarantees was investigated. It was also in this domain that the idea of proving lower bounds on the performance of online algorithm was first developed. In this paper, we consider an generalization of the classical bin packing–hypercube packing.

Problem Definition. Let $d \geq 1$ be an integer. In the d -dimensional bin packing problem we receive a sequence δ of items p_1, p_2, \dots, p_n . Each item p has a fixed size, which is $s(p) \times \dots \times s(p)$, i.e., $s(p)$ is the size of p in any dimension. We have an infinite number of bins, each of which is a d -dimensional unit hypercube. Each item must be assigned to a bin and a position $(x_1(p), \dots, x_d(p))$, where $0 \leq x_i(p)$ and $x_i(p) + s(p) \leq 1$ for $1 \leq i \leq d$. Further, the positions must be assigned in such a way that no two items in the same bin overlap. Note that for $d = 1$ the problem reduces to exactly the classic bin packing problem. In this paper, we study the *online* version of this problem, i.e., each item must be assigned in turn, without knowledge of the next items.

Asymptotic competitive ratio. To evaluate an online algorithms for bin packing we use the standard measure defined as follows.

Given an input list L and an online algorithm A , we denote by $OPT(L)$ and $A(L)$, respectively, the cost (number of bins used) by an optimal (offline) algorithm and the cost by online algorithm A for packing list L . The *asymptotic competitive ratio* R_A^∞ of algorithm A is defined by

$$R_A^\infty = \lim_{k \rightarrow \infty} \sup_L \{A(L)/OPT(L) | OPT(L) = k\}.$$

Previous results. On the classic online bin packing, Johnson, Demers, Ullman, Garey and Graham [9] showed that the First Fit algorithm has the competitive ratio 1.7. Yao [17] gave an upper bound of $5/3$. Lee and Lee [11] showed the Harmonic algorithm has the competitive ratio 1.69103 and improved it to 1.63597. Ramanan, Brown, Lee and Lee [13] improved the upper bound to 1.61217. Currently, the best known upper bound is 1.58889 by Seiden [14]. On the lower bounds, Yao [17] showed no online algorithm has performance ratio less than 1.5. Brown [1] and Liang [10] independently improved this lower bound to 1.53635. The lower bound currently stands at 1.54014, due to van Vliet [16].

On online hypercube packing, Coppersmith and Raghavan [3] showed an upper bound of $43/16 = 2.6875$ for online square packing and an upper bound 6.25 for online cube packing. The upper bound for square packing was improved to $395/162 < 2.43828$ by Seiden and van Stee [15]. For online cube packing, Miyazawa and Wakabayashi [12] showed an upper bound of 3.954. Epstein and van Stee [6] gave an upper bound of 2.2697 for square packing and an upper bound of 2.9421 for online cube packing. By using a computer program, the upper bound for square packing was improved to 2.24437 by Epstein and van Stee [8]. They [8] also gave lower bounds of 1.6406 and 1.6680 for square packing and cube packing, respectively.

Our contributions. Based on the techniques in one dimensional bin packing algorithm Super Harmonic by Seiden [14], we give a new framework for online hypercube packing and get new upper bounds for asymptotic competitive ratios. For square packing, we get an upper bound 2.1439 which is better than 2.24437[8]. For cube packing, we also give a new upper bound 2.6852, which is better than 2.9421[8].

Definition: If an item p of size (side length) $s(p) \leq 1/M$, where M is a fixed integer, then call p *small*, otherwise *large*.

2 Previous Research

We first introduce previous research on online hypercube packing that is useful in our algorithm.

2.1 Online packing small items

The following algorithm for packing small items is based on [4], [7]. The key ideas are below:

1. Classify all *small* squares into M groups. In detail, for an item p of size $s(p)$, we classify it into group i such that $2^k s(p) \in (1/(i+1), 1/i]$, where $i \in \{M, \dots, 2M-1\}$ and k is an integer.
2. Exclusively pack items of the same group into bins, i.e., each bin is used to pack items belonged to the same group. During packing, one bin may be partitioned into sub-bins.

Definition: An item is defined to be of type i if it belongs to group i . A sub-bin which received an item is said to be *used*. A sub-bin which is not used and not cut into smaller sub-bins is called *empty*. A bin is called *active* if it can still receive items, otherwise *closed*.

Given an item p of type i , where $2^k s(p) \in (1/(i+1), 1/i]$, *algorithm* AssignSmall(i) works as the following.

1. If there is an empty sub-bin of size $1/(2^k i)$, then the item is simply packed there.
2. Else, in the current bin, if there is no empty sub-bin of size $1/(2^j i)$ for $j < k$, the bin is closed and a new bin is opened and partitioned into sub-bins of size $1/i$. If $k = 0$ then pack the item in one of sub-bins of size $1/i$. Else goto next step.

3. Take an empty sub-bin of size $1/(2^j i)$ for a maximum $j < k$. Partition it into 2^d identical sub-bins. If the resulting sub-bins are larger than $1/(2^k i)$, then take *one* of them and partition it in the same way. This is done until sub-bins of size $1/(2^k i)$ are reached. Then the item is packed in one such sub-bin.

The following results are from [7].

Lemma 1 *In the above algorithm, i) at all items there are at most M active bins. ii) in each closed bin of type $i \geq M$, the occupied volume is at least $(i^d - 1)/(i + 1)^d \geq (M^d - 1)/(M + 1)^d$.*

So, roughly speaking, a small item with size x takes at most a $\frac{(M+1)^d}{(M^d-1)} \times x^d$ bin.

3 Algorithm \mathcal{A} for online hypercube packing

The key points in our online algorithm are

1. divide all items into *small* and *large* groups.
2. pack small items by algorithm AssignSmall, pack large items by extended Super Harmonic algorithm.

Classification of large items: Given an integer $M \geq 11$, let $t_1 = 1 > t_2 > \dots > t_{N+1} = 1/M > t_{N+2} = 0$, where N is a fixed integer. We define the interval I_j to be $(t_{j+1}, t_j]$ for $j = 1, \dots, N + 1$ and say a large item p of size $s(p)$ has type i if $s(p) \in I_i$.

Definition: An item of size s has type $\tau(s)$, where

$$\tau(s) = j \quad \Leftrightarrow \quad s \in I_j.$$

Parameters in algorithm \mathcal{A} : An instance of the algorithm is described by the following parameters: integers N and K ; real numbers $1 = t_1 > t_2 > \dots > t_N > t_{N+1} = 1/M$, $\alpha_1, \dots, \alpha_N \in [0, 1]$ and $0 = \Delta_0 < \Delta_1 < \dots < \Delta_K < 1/2$, and a function $\phi : \{1, \dots, N\} \mapsto \{0, \dots, K\}$.

Next, we give the operation of our algorithm, essentially, which is quite similar with the Super Harmonic algorithm [14]. Each *large* item of type j is assigned a color, *red* or *blue*. The algorithm uses two sets of counters, e_1, \dots, e_N and s_1, \dots, s_N , all of which are initially zero. s_i keeps track of the total number of type i items. e_i is the number of type i items which get colored red. For $1 \leq i \leq N$, the invariant $e_i = \lfloor \alpha_i s_i \rfloor$ is maintained, i.e. the percentage of type i items colored red is approximately α_i .

We first introduce some parameters used in Super Harmonic algorithm, then give the corresponding ones for d -dimensional packing. In one dimensional packing, a bin can be placed at most $\beta_i = \lfloor 1/t_i \rfloor$ items with size t_i . After packing β_i type i items, there is $\delta_i = 1 - t_i \beta_i$ space left. The rest space can be used for red items. However, we sometimes use less than δ_i in a bin in order to simplify the algorithm and its analysis, i.e., we use $\mathcal{D} = \{\Delta_1, \dots, \Delta_K\}$ instead of the set of δ_i , for all i . $\Delta_{\phi(i)}$ is the amount of space used to hold red items in a bin which holds blue items of type i . We therefore require that ϕ satisfy $\Delta_{\phi(i)} \leq \delta_i$. $\phi(i) = 0$ indicates that no red items are accepted. To ensure that every red item potentially can be packed, we require that $\alpha_i = 0$ for all i such that $t_i > \Delta_K$, that is, there are no red items of type i . Define $\gamma_i = 0$ if $t_i > \delta_K$ and $\gamma_i = \max\{1, \lfloor \Delta_1/t_i \rfloor\}$, otherwise. This is the number of red item of type i placed in a bin.

In d -dimensional packing, we place β_i^d blue items of type i into a bin and introduce a new parameter θ_i instead of γ_i . Let

$$\theta_i = \beta_i^d - (\beta_i - \gamma_i)^d.$$

This is the number of red items of type i that the algorithm places together in a bin. In details, if $t_i > \Delta_K$, then $\theta_i = 0$, i.e., we do not pack type i items as red items. So, in this case, we require $\alpha_i = 0$. Else if $t_i \leq \Delta_1$, then $\theta_i = \beta_i^d - (\beta_i - \lfloor \Delta_1/t_i \rfloor)^d$. If $\Delta_1 < t_i \leq \Delta_K$, we set $\theta_i = \beta_i^d - (\beta_i - 1)^d$.

Here, we illustrate the structure of a bin for $d = 2$.

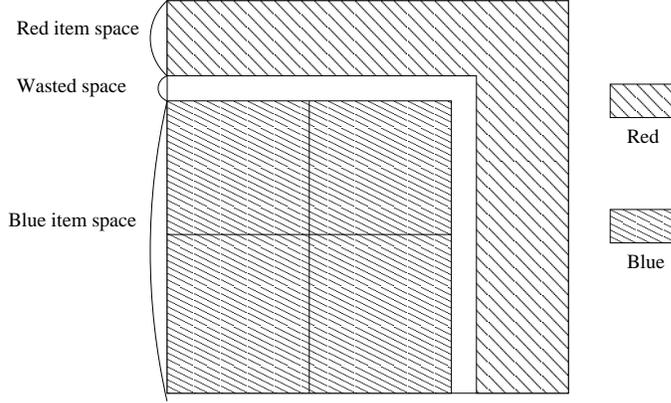


Figure 1: If the bin is a (i, j) or $(i, ?)$ bin, the amount of space for blue items is $(t_i\beta_i)^2$. The amount of space left is $1 - (t_i\beta_i)^2$. The amount of this space actually used for red items is $1 - (1 - \Delta_{\phi(i)})^2$, where $\Delta_{\phi(i)} \leq \delta_i = 1 - t_i\beta_i$.

Naming bins: Bins are named as follows:

$$\begin{aligned} & \{i | \phi_i = 0, 1 \leq i \leq N, \} \\ & \{(i, ?) | \phi_i \neq 0, 1 \leq i \leq N, \} \\ & \{(? , j) | \alpha_j \neq 0, 1 \leq j \leq N, \} \\ & \{(i, j) | \phi_i \neq 0, \alpha_j \neq 0, \gamma_j t_j \leq \Delta_{\phi(i)}, 1 \leq i, j \leq N\}. \end{aligned}$$

We call these groups *monochromatic*, *indeterminate blue*, *indeterminate red* and *bichromatic*, respectively. And we call the monochromatic and bichromatic groups *final* groups.

The monochromatic group i contains bins that hold only blue items of type i . There is only one open bin in each of these groups; this bin has fewer than β_i^d items. The closed bins all contain β_i^d items.

The bichromatic group (i, j) contains bins that contain blue items of type i along with red items of type j . A closed bin in this group contains β_i^d type i items and θ_j type j items. There are at most three open bins.

The indeterminate blue group $(i, ?)$ contains bins that hold only blue items of type i . These bins are all open, but only one has fewer than β_i^d items.

The indeterminate red group $(?, j)$ contains bins that hold only red items of type j . These bins are all open, but only one has fewer than θ_j items.

Essentially, the algorithm tries to minimize the number of indeterminate bins, while maintaining all the aforementioned invariants. That is, we try to place red and blue items together whenever possible; when this is not possible we place them in indeterminate bins in hope that they can later be so combined.

Algorithm A: A formal description of algorithm \mathcal{A} is given as blow:

Initialize $e_i \leftarrow 0$ and $s_i \leftarrow 0$ for $1 \leq i \leq M + 1$.

For a small item p , call algorithm AssignSmall.

For a large item p :

$$i \leftarrow \tau(p), \quad s_i \leftarrow s_i + 1.$$

If $e_i < \lfloor \alpha_i s_i \rfloor$:

$$e_i \leftarrow e_i + 1.$$

Color p red.

If there is an open bin in group $(?, i)$ with fewer than θ_i type i items, then pack p in this bin.

If there is an open bin in group (j, i) with fewer than θ_i type i items, then pack p in this bin.

Else if there is some bin in group $(j, ?)$ such that $\Delta_{\phi(j)} \geq \gamma_i t_i$ then place p in it and change the group of this bin to (j, i) .

Otherwise, open a new group $(?, i)$ bin and place p in it.

Else:

Color p blue.

If $\phi(i) = 0$:

If there is an open bin in group i with fewer than β_i^d items, then place p in it.

Otherwise, open a new group i bin and pack p there.

Else:

If, for any j , there is an open bin (i, j) with fewer than β_i^d items, then place p in this bin.

Else, if there is some bin in group $(i, ?)$ with fewer than β_i^d items, then place p in this bin.

Else, if there is some bin in group $(?, j)$ such that $\Delta_{\phi(i)} \geq \gamma_j t_j$ then pack p in it and change the group of this bin to (i, j) .

Otherwise, open a new group $(i, ?)$ bin and pack p there.

4 The analyses for square and cube packing

In this section, we specify values for parameters in the algorithm and use weighting functions to analyze the competitive ratios for square and cube packing respectively.

4.1 An instance of algorithm \mathcal{A}

Let $M = 11$, i.e., a *small* item has its side length as most $1/11$. And the parameters in \mathcal{A} are given in the following tables.

i	$(t_{i+1}, t_i]$	β_i	δ_i	$\phi(i)$	γ_i
1	(0.7, 1]	1	0	0	0
2	(0.65, 0.7]	1	0.3	2	0
3	(0.60, 0.65]	1	0.35	3	0
4	(0.5, 0.60]	1	0.4	4	0
5	(0.4, 0.5]	2	0	0	0
6	(0.35, 0.4]	2	0.2	1	1
7	(1/3, 0.35]	2	0.3	2	1
8	(0.30, 1/3]	3	0	0	0
9	(1/4, 0.30]	3	0.1	0	1
10	(1/5, 1/4]	4	0	0	1
11	(1/6, 1/5]	5	0	0	1
12	(1/7, 1/6]	6	0	0	1
13	(1/8, 1/7]	7	0	0	1
14	(1/9, 1/8]	8	0	0	1
15	(0.1, 1/9]	9	0	0	1
16	(1/11, 0.1]	10	0	0	2
17	(0, 1/11]	*	*	*	*

$j = \phi(i)$	Δ_j	Red items accepted
1	0.20	11..16
2	0.30	9..16
3	0.35	7, 9..16
4	0.40	6..7, 9..16

Observation: From the above tables, in a $(4, ?)$ bin, there is $\Delta_4 = 0.4$ space left. So, all red items with size at most 0.4 can be packed in $(4, ?)$ bins. In the same ways, all red items with size at most 0.35 can be packed in $(4, ?)$ and $(3, ?)$ bins, all red items with size at most 0.30 can be packed in $(4, ?)$, $(3, ?)$, $(7, ?)$ and $(2, ?)$ bins, all red items with size at most 0.2 can be packed in $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$, $(6, ?)$ bins.

Next, we define weight function $W(p)$ for a given item p with size x . Roughly speaking, a weight of an item is the maximal portion of a bin that it can occupy. Given a small item p with size x , by Lemma 1, averagely it occupies a $\frac{x^d(11+1)^d}{11^d-1}$ bin. So, we define

$$W(p) = \frac{x^d(11+1)^d}{11^d-1}.$$

Given a large item p , we define its weight based on whether there are $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$, $(6, ?)$ bins existed after packing. For example, consider the case in which there are no $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$ bins existed, but there are some $(6, ?)$ bins. Then by the above tables, there is $\Delta_1 = 0.2$ space left in a $(6, ?)$ bin, and the bin can accept type i items, where $11 \leq i \leq 16$. So, there are no $(?, i)$ bins for $11 \leq i \leq 16$, otherwise red type i items should be packed in $(6, ?)$ bins by our algorithm. But there may be (i) bins and the total number of type i items in these bins is bounded by $(1 - \alpha_i)s_i$ in our algorithm, where s_i is the total number of type i items, and $11 \leq i \leq 16$. Since in a closed (i) bin, there are β_i^d items, then averagely, we can see a type i items takes $\frac{1-\alpha_i}{\beta_i^d}$ bin, where $11 \leq i \leq 16$. In the same ways, a type i takes $\frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i}$ bin, where $i = 6, 9, 10$, and a type i item takes $\frac{1}{\beta_i^d}$ bin, where $i = 1, 5, 8$. Since there are no $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$ bins, a type i item takes zero bin, where $i = 2, 3, 4$. But there may have $(?, 7)$ or $(j, 7)$ bins and the total number of type 7 items in these bins is $\alpha_7 s_7$, then a type 7 item takes $\frac{\alpha_7}{\theta_7}$. (For details, refer to the definition of $W_4(p)$)

Let n_1 be the number of $(4, ?)$ bins, n_2 be the number of $(3, ?)$ bins, n_3 be the number of $(7, ?)$ and $(2, ?)$ bins, n_4 be the number of $(6, ?)$ bins. After packing, based on the values of n_i , where $1 \leq i \leq 4$, we have five cases as below.

Case 1 $n_1 \neq 0$, i.e., there are some $(4, ?)$ bins.

Case 2 $n_1 = 0$ and $n_2 \neq 0$, i.e., no $(4, ?)$ bins, but there are some $(3, ?)$ bins.

Case 3 $n_1 = n_2 = 0$ and $n_3 \neq 0$, no $(4, ?)$, $(3, ?)$ bins, but there are some $(7, ?)$ or $(2, ?)$ bins.

Case 4 $n_1 = n_2 = n_3 = 0$, but $n_4 \neq 0$, i.e., no $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$ bins, but only $(6, ?)$ bins.

Case 5 $n_1 = n_2 = n_3 = n_4 = 0$, no $(4, ?)$, $(3, ?)$, $(7, ?)$, $(2, ?)$, $(6, ?)$ bins.

According to the above observation, we define weighting function W_i for case i as follows, where p is an item with size x .

$$\begin{aligned}
W_1(p) &= \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1..5, 8. \\ \frac{1-\alpha_i}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 6, 7, 9..16. \end{cases} \\
W_2(p) &= \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1, 2, 3, 5, 8. \\ 0 & \text{if } x \in I_4. \\ \frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 6. \\ \frac{1-\alpha_i}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 7, 9..16. \end{cases} \\
W_3(p) &= \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1, 2, 5, 8. \\ 0 & \text{if } x \in I_i, \text{ for } i = 3, 4. \\ \frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 6, 7. \\ \frac{1-\alpha_i}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 9..16. \end{cases} \\
W_4(p) &= \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1, 5, 8. \\ 0 & \text{if } x \in I_2, I_3, I_4 \\ \frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 6, 9..10 \\ \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 7. \\ \frac{1-\alpha_i}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 11..16. \end{cases} \\
W_5(p) &= \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1, 5, 8. \\ 0 & \text{if } x \in I_2, I_3, I_4. \\ \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 6, 7. \\ \frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 9..16. \end{cases}
\end{aligned}$$

Definition: A set of items X is a feasible set if all items in it can be packed into a bin. And,

$$W_i(X) = \sum_{p \in X} W_i(p) \quad \text{where } 1 \leq i \leq 5.$$

Define $\mathcal{P}(W) = \max\{W_i(X)\}$ over all feasible sets X .

We defined five sets of weighting functions for all items. This is a weighting system, which is a special case of general weighting system defined in [14]. So, the following lemma follows directly from [14].

Lemma 2 *The asymptotic performance ratio of \mathcal{A} is upper bounded by $\mathcal{P}(W)$.*

4.2 Upper bounds for square and cube packing

Next, we estimate $\mathcal{P}(W)$ for square and cube packing. For convenience of calculations, we combine two weighting functions W_4 and W_5 into a new weight function W_4 . Actually, through calculations, we find out this does not affect the estimate of the upper bounds. In the following calculations, we use the new function W_4 defined as follows.

$$W_4(p) = \begin{cases} \frac{1}{\beta_i^d} & \text{if } x \in I_i, \text{ for } i = 1, 5, 8. \\ 0 & \text{if } x \in I_2, I_3, I_4 \\ \frac{1-\alpha_i}{\beta_i^d} + \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 6, 9..16 \\ \frac{\alpha_i}{\theta_i} & \text{if } x \in I_i, \text{ for } i = 7. \end{cases}$$

Theorem 1 *The asymptotic performance ratio of \mathcal{A} for square packing is at most 2.1439.*

Proof. For square packing, we set parameters α_i according to the following table.

i	1-4	5	6	7	8	9	10	11	12	13	14	15	16
α_i	0	0	0.12	0.2	0	0.2546	0.2096	0.15	0.1	0.1	0.1	0.1	0.05
θ_i	0	0	3	3	0	5	7	9	11	13	15	17	36
β_i^2	1	4	4	4	9	9	16	25	36	49	64	81	100

Given an item p with size x , define an efficient function $E_i(p)$ as $W_i(p)/x^2$, then we have the following table.

i	$(t_{i+1}, t_i]$	$W_1(p)$	$E_1(p)$	$W_2(p)$	$E_2(p)$	$W_3(p)$	$E_3(p)$	$W_4(p)$	$E_4(p)$
1	(0.7, 1]	1	2.05	1	2.05	1	2.05	1	2.05
2	(0.65, 0.7]	1	2.37	1	2.37	1	2.37	0	0
3	(0.6, 0.65]	1	2.7778	1	2.7778	0	0	0	0
4	(0.5, 0.6]	1	4	0	0	0	0	0	0
5	(0.4, 0.5]	1/4	1.5625	1/4	1.5625	1/4	1.5625	1/4	1.5625
6	(0.35, 0.4]	0.22	1.8	0.26	2.123	0.26	2.123	0.26	2.123
7	(1/3, 0.35]	0.2	1.8	0.2	1.8	0.8/3	2.4	0.2/3	0.6
8	(0.3, 1/3]	1/9	1.235	1/9	1.235	1/9	1.235	1/9	1.235
9	(1/4, 0.3]	0.0829	1.327	0.0829	1.327	0.0829	1.327	0.1338	2.141
10	(1/5, 1/4]	0.0494	1.235	0.0494	1.235	0.0494	1.235	0.0794	1.99
11	(1/6, 1/5]	0.034	1.224	0.034	1.224	0.034	1.224	0.05067	1.824
12	(1/7, 1/6]	0.025	1.225	0.025	1.225	0.025	1.225	0.03410	1.6705
13	(1/8, 1/7]	0.01837	1.1756	0.01837	1.1756	0.01837	1.1756	0.02606	1.6679
14	(1/9, 1/8]	0.9/64	1.2	0.9/64	1.2	0.9/64	1.2	0.02073	1.6791
15	(0.1, 1/9]	0.1/9	1.2	0.1/9	1.2	0.1/9	1.2	0.017	1.7
16	(1/11, 0.1]	0.0095	1.2	0.0095	1.2	0.0095	1.2	0.01089	1.3176
17	(0, 1/11]	$1.2x^2$	1.2	$1.2x^2$	1.2	$1.2x^2$	1.2	$1.2x^2$	1.2

Definition Let $m_i \geq 0$ be the number of type i items in X . Based on the number of items m_i , we have 5 cases for estimating the upper bound of $\mathcal{P}(W) = \max\{W_i(X)\}$.

Case 1: There are no type 1, 2, 3, 4 items in X , i.e., $m_i = 0$ for $1 \leq i \leq 4$. Then,

$$W_4(X) = \sum_{p \in X} E_4(p)s(p)^2 \leq 2.141 \sum_{p \in X} s(p)^2 \leq 2.141.$$

On the other side, we have $m_6 + m_7 \leq 4$ and $W_1(X) \leq W_2(X) \leq W_3(X)$,

$$W_3(X) = \sum_{p \in X} W_3(p) \leq 0.26m_6 + \frac{0.8m_7}{3} + 1.5625(1 - 0.35^2m_6 - \frac{m_7}{9}) < 2.$$

So, $\mathcal{P}(W) \leq \max\{W_i(X)\} \leq 2.141$.

Case 2: $m_1 = 1$, there is one type 1 item in X . So, $m_i = 0$, where $2 \leq i \leq 8$. Then

$$\mathcal{P}(W) \leq 2.141.$$

Case 3: $m_2 = 1$, one type 2 item in X . So no type 1, 3, 4, 5, 6 items in X . Then we have $W_4(X) \leq 2.141$. And $W_1(X) \leq W_2(X) \leq W_3(X)$. Since no type 5, 6 items in X and $m_7 + m_9 \leq 5$, $m_7 \leq 3$, then

$$W_3(X) \leq 1 + \frac{0.8m_7}{3} + 0.0829m_9 + 1.235(1 - 0.65^2 - m_7/9 - m_9/16) < 2.12.$$

The last inequality follows from $m_7 = 3$ and $m_9 = 2$. So, $\mathcal{P}(W) \leq 2.141$.

Case 4: $m_3 = 1$, one type 3 item in X . So no type 1, 2, 4, 5 items in X . Then

$$\max\{W_3(X), W_4(X)\} \leq 0 + 2.4(1 - 0.6^2) < 1.6.$$

Else, $W_1(X) \leq W_2(X)$, $m_6 + m_7 \leq 3$ and $m_6 + m_7 + m_9 \leq 5$,

$$\begin{aligned} W_2(X) &\leq 1 + 0.26m_6 + 0.2m_7 + 0.0829m_9 \\ &\quad + 1.235(1 - 0.6^2 - 0.35^2m_6 - m_7/9 - m_9/16) \\ &< 2.134. \end{aligned}$$

The last inequality follows from $m_6 = 3$ and $m_9 = 2$. So, $\mathcal{P}(W) \leq 2.134$.

Case 5: $m_4 = 1$, one type 4 item in X . So no type 1, 2, 3 items in X . Then

$$\max\{W_2(X), W_3(X), W_4(X)\} \leq 0 + 2.4(1 - 0.5^2) \leq 1.8.$$

Else $m_5 + m_6 + m_7 \leq 3$ and $m_6 + m_7 + m_9 \leq 5$,

$$\begin{aligned} W_1(X) &\leq 1 + m_5/4 + 0.22m_6 + 0.2m_7 + 0.0829m_9 \\ &\quad + 1.235(1 - 0.5^2 - 0.4^2m_5 - 0.35^2m_6 - m_7/9 - m_9/16) \\ &< 2.1439. \end{aligned}$$

The last inequality follows from $m_6 = 3$ and $m_9 = 2$. So, $\mathcal{P}(W) \leq 2.1439$. □

Theorem 2 *The asymptotic performance ratio of \mathcal{A} for cube packing is at most 2.6852.*

Proof. For cube packing, we set parameters α_i and θ_i in the following table.

i	1 - 4	5	6	7	8	9	10	11	12 - 16
α_i	0	0	0.12	0.2	0	0.325	0.2096	0.15	0
θ_i	0	0	7	7	0	19	37	61	0
β_i^3	1	8	8	8	27	27	64	125	$(i - 6)^3$

Here we set $\alpha_i = 0$ for $12 \leq i \leq 16$. So, their weights are defined as $1/\beta_i^3$.

Given an item p with size x , an efficient function $E_i(p)$ is defined as $W_i(p)/x^3$, then we have the following table.

i	$(t_{i+1}, t_i]$	$W_1(p)$	$E_1(p)$	$W_2(p)$	$E_2(p)$	$W_3(p)$	$E_3(p)$	$W_4(p)$	$E_4(p)$
1	(0.7, 1]	1	2.9155	1	2.9155	1	2.9155	1	2.9155
2	(0.65, 0.7]	1	3.65	1	3.65	1	3.65	0	0
3	(0.6, 0.65]	1	4.63	1	4.63	0	0	0	0
4	(0.5, 0.6]	1	8	0	0	0	0	0	0
5	(0.4, 0.5]	1/8	1.9532	1/8	1.9532	1/8	1.9532	1/8	1.9532
6	(0.35, 0.4]	0.11	2.5656	0.1272	2.966	0.1272	2.966	0.1272	2.966
7	(1/3, 0.35]	0.1	2.7	0.1	2.7	0.1286	3.472	0.03	0.81
8	(0.3, 1/3]	1/27	1.372	1/27	1.372	1/27	1.372	1/27	1.372
9	(1/4, 0.3]	0.025	1.6	0.025	1.6	0.025	1.6	0.04211	2.6948
10	(1/5, 1/4]	0.0124	1.55	0.0124	1.55	0.0124	1.55	0.01802	2.252
11	(1/6, 1/5]	0.0068	1.4688	0.0068	1.4688	0.0068	1.4688	0.0093	2
12	(1/7, 1/6]	1/216	1.59	1/216	1.59	1/216	1.59	1/216	1.59
13	(1/8, 1/7]	1/7 ³	1.5						
14	(1/9, 1/8]	1/8 ³	1.43						
15	(0.1, 1/9]	1/9 ³	1.38						
16	(1/11, 0.1]	0.001	1.331	0.001	1.331	0.001	1.331	0.001	1.331
17	(0, 1/11]	1.3x ³	1.3						

Let $m_i \geq 0$ be the number of type i items in X . We consider the following 5 cases to estimate $\mathcal{P}(W)$.

Case 1: There are no type 1, 2, 3, 4 items in X , i.e., $m_i = 0$ for $1 \leq i \leq 4$. Since $m_6 + m_9 \leq 27$ and $m_6 \leq 8$,

$$W_4(X) \leq 0.1272m_6 + 0.04211m_9 + 2.252(1 - 0.35^3m_6 - m_9/64) \leq 2.63.$$

The last inequality holds for $m_6 = 8$ and $m_9 = 19$.

On the other hand, $W_1(X) \leq W_2(X) \leq W_3(X)$ and $m_6 + m_7 \leq 8$,

$$W_3(X) \leq 0.1272m_6 + 0.1286m_7 + 1.96(1 - 0.35^3m_6 - m_7/27) \leq 2.41.$$

The last inequality holds for $m_6 = 0$ and $m_7 = 8$. So, we have $\mathcal{P}(W) \leq \max\{W_i(X)\} \leq 2.63$.

Case 2: $m_1 = 1$, there is one type 1 item in X . So, $m_i = 0$, where $2 \leq i \leq 8$. And $m_9 \leq 19$, then

$$\mathcal{P}(W) \leq 1 + 19 \times 0.04211 + 2.252(1 - 0.7^3 - 19/64) \leq 2.62.$$

Case 3: $m_2 = 1$, one type 2 item in X . So no type 1, 3, 4, 5, 6 items in X . Then as proved in Case 1, $W_4(X) \leq 2.63$. Since no type 5, 6 items in X and $m_7 \leq 7$, then

$$\max\{W_1(X), W_2(X), W_3(X)\} \leq 1 + 0.1286 \times 7 + 1.6(1 - 0.65^3 - 7/27) < 2.646.$$

So, $\mathcal{P}(W) < 2.646$.

Case 4: $m_3 = 1$, one type 3 item in X . So no type 1, 2, 4, 5 items in X . Then by the proof in Case 1

$$\max\{W_3(X), W_4(X)\} < 2.63.$$

Else we just need to consider $W_2(X)$, for $W_1(X) \leq W_2(X)$. Since $m_1 + m_2 + m_4 + m_5 = 0$ and $m_6 + m_7 \leq 7$,

$$W_2(X) \leq 1 + 0.1272m_6 + 0.1m_7 + 1.6(1 - 0.6^3 - 0.35^3m_6 - m_7/27) \leq 2.6646.$$

The last inequality follows from $m_6 = 7$ and $m_7 = 0$. So, $\mathcal{P}(W) < 2.6646$.

Case 5: $m_4 = 1$, one type 4 item in X . So no type 1, 2, 3 items in X . Then by the proof in Case 1

$$\max\{W_2(X), W_3(X), W_4(X)\} < 2.63.$$

Else $m_5 + m_6 + m_7 \leq 7$ and $m_1 + m_2 + m_3 = 0$,

$$\begin{aligned} W_1(X) &\leq 1 + m_5/8 + 0.11m_6 + 0.1m_7 \\ &\quad + 1.6(1 - 0.5^3 - 0.4^3m_5 - 0.35^3m_6 - m_7/27) \\ &< 2.6852. \end{aligned}$$

The last inequality follows from $m_7 = 7$ and $m_5 = m_6 = 0$. So, $\mathcal{P}(W) < 2.6852$. \square

5 Concluding Remarks

In this page, we reduce the gaps between the upper and lower bounds of online square packing and cube packing. But the gaps are still large. It seems possible to use computer proof as the one in [14] to get a more precise upper bound. So, how to reduce the gaps is a challenging open problem.

References

- [1] D.J. Brown, A lower bound for on-line one-dimensional bin packing algorithms, *Technical report R864*, Coordinated Sci. Lab., Urbana, Illinois, 1979.
- [2] E.G. Coffman, M.R. Garey and D.S. Johnson, Approximation algorithms for bin packing: a survey. In *Approximation Algorithms for NP-hard Problems*, D. Hochbaum, Ed. PWS, Boston, MA, 1997, chapter 2.
- [3] D. Coppersmith, P. Paghavan, Multidimensional on-line bin packing: Algorithms and worst case analysis, *Oper. Res. Lett.* 8:17-20, 1989.
- [4] J. Csirik, A. van Vliet, An on-line algorithm for multidimensional bin packing, *Oper. Res. Lett.* 13: 149-158, 1993.
- [5] J. Csirik and G.J. Woeginger, Shelf algorithm for on-line strip packing, *Information Processing Letters* 63, 171-175, 1997.
- [6] L. Epstein, R. van Stee, Optimal online bounded space multidimensional packing, *SODA 2004*: 214-223.
- [7] L. Epstein, R. van Stee, Optimal Online Algorithms for Multidimensional Packing Problems. *SIAM J. Computing*, 35(2): 431-448, 2005.
- [8] L. Epstein, R. van Stee, Online square and cube packing, *Acta Inf.* 41(9): 595-606, 2005.
- [9] D.S. Johnson, A.J. Demers, J.D. Ullman, M. R. Garey, R. L. Graham, Worst-Case performance bounds for simple one-dimensional packing algorithms, *SIAM J. Comput.* 3(4): 299-325, 1974.
- [10] F.M. Liang, A lower bound for online bin packing, *Information processing letters* 10,76-79,1980.
- [11] C.C. Lee and D.T. Lee, A simple on-line bin-packing algorithm, *J. ACM* 32, 562-572, 1985.

- [12] F.K. Miyazawa, Y. Wakabayashi, Cube packing, *Theor. Comput. Sci.* 1-3(297): 355-366, 2003.
- [13] P.V. Ramanan, D.J. Brown, C.C. Lee, and D. T. Lee, On-line bin packing in linear Time, *J. Algorithms* 10, 305-326, 1989.
- [14] S.S. Seiden, On the online bin packing problem, *J. ACM* 49, 640-671, 2002.
- [15] S.S. Seiden, R. van Stee, New bounds for multidimensional packing, *Algorithmica* 36(3): 261-293, 2003.
- [16] A. van Vliet, An improved lower bound for on-line bin packing algorithms, *Inform. Process. Lett.* 43, 277-284, 1992.
- [17] A.C.-C. Yao, New Algorithms for Bin Packing, *J. ACM* 27, 207-227, 1980.