An 8/3 Lower Bound for Online Dynamic Bin Packing

Prudence W.H. Wong, Fencol C.C. Yung, and Mihai Burcea[⋆]

Department of Computer Science, University of Liverpool, UK
{pwong,m.burcea}@liverpool.ac.uk, ccyung@graduate.hku.hk

Abstract. We study the dynamic bin packing problem introduced by Coffman, Garey and Johnson. This problem is a generalization of the bin packing problem in which items may arrive and depart dynamically. The objective is to minimize the maximum number of bins used over all time. The main result is a lower bound of $8/3 \sim 2.666$ on the achievable competitive ratio, improving the best known 2.5 lower bound. The previous lower bounds were 2.388, 2.428, and 2.5. This moves a big step forward to close the gap between the lower bound and the upper bound, which currently stands at 2.788. The gap is reduced by about 60% from 0.288 to 0.122. The improvement stems from an adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only and this can be adapted appropriately regardless of the current load of other bins that have already been opened by $\mathcal{A}$. Comparing with the previous 2.5 lower bound, this basic step gives a better way to derive the complete adversary and a better use of items of slightly different sizes leading to a tighter lower bound. Furthermore, we show that the 2.5-lower bound can be obtained using this basic step in a much simpler way without case analysis.

1 Introduction

Bin packing is a classical combinatorial optimization problem [6, 8, 9]. The objective is to pack a set of items into a minimum number of unit-size bins such that the total size of the items in a bin does not exceed the bin capacity. The problem has been studied extensively both in the offline and online settings. It is well-known that the problem is NP-hard [11]. In the online setting [14, 15], items may arrive at arbitrary time; item arrival time and item size are only known when an item arrives. The performance of an online algorithm is measured using competitive analysis [3]. Consider any online algorithm $\mathcal{A}$. Given an input I, let $OPT(I)$ and $\mathcal{A}(I)$ be the maximum number of bins used by the optimal offline algorithm and $\mathcal{A}$, respectively. Algorithm $\mathcal{A}$ is said to be c-competitive if there exists a constant b such that $\mathcal{A}(I) \le c\,OPT(I) + b$ for all I.

Online dynamic bin packing. Most existing work focuses on "static" bin packing in the sense that items do not depart. In some potential applications like

[⋆] Supported by EPSRC Studentship.

warehouse storage, a more realistic model takes into consideration of dynamic arrival and departures of items. In this natural generalization, known as dynamic bin packing [7], items arrive over time, reside for some period of time, and may depart at arbitrary time. Each item has to be assigned to a bin from the time it arrives until it departs. The objective is to minimize the maximum number of bins used over all time. Note that migration to another bin is not allowed. In the online setting, the size and arrival time is only known when an item arrives and the departure time is only known when the item departs.

In this paper, we focus on online dynamic bin packing. It is shown in [7] that First-Fit has a competitive ratio between 2.75 and 2.897, and a modified first-fit algorithm is 2.788-competitive. A lower bound of 2.388 is given for any deterministic online algorithm. This lower bound has later been improved to 2.428 [4] and then 2.5 [5]. The problem has also been studied in two- and three-dimension as well as higher dimension [10, 16]. Other work on dynamic bin packing considered a restricted type of items, namely unit-fraction items [2, 4, 12]. Furthermore, Ivkovic and Lloyd [13] studied the fully dynamic bin packing problem, which allows repacking of items for each item arrival or departure and they gave a 1.25-competitive online algorithm for this problem. Balogh et al. [1] studied the problem when a limited amount of repacking is allowed.

Our contribution. We improve the lower bound of online dynamic bin packing for any deterministic online algorithm from 2.5 to $8/3 \sim 2.666$. This makes a big step forward to close the gap with the upper bound, which currently stands at 2.788 [7]. The improvement stems from an adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only and this can be adapted appropriately regardless of the load of current bins opened by $\mathcal{A}$. Comparing with the previous 2.5 lower bound, this basic step gives a better use of items of slightly different sizes leading to a tighter lower bound. Furthermore, we show in Section 3.3 that the 2.5-lower bound can be obtained using this basic step in a much simpler way without case analysis. It is worth mentioning that we consider optimal packing without migration at any time.

The adversarial sequence is composed of two operations, namely Op-Inc and Op-Comp. Roughly speaking, Op-Inc uses a load of at most s to make $\mathcal{A}$ open s bins, this is followed by some item departure such that each bin is left with only one item and the size is increasing across the bins. Op-Comp then releases items of complementary size such that for each item of size x, items of size $1 - x$ are released. The complementary size ensures that the optimal offline algorithm $\mathcal{O}$ is able to pack all these items using s bins while the sequence of arrival ensures that $\mathcal{A}$ has to pack these complementary items into separate bins.

2 Preliminaries

In dynamic bin packing, items arrive and depart at arbitrary time. Each item comes with a size. We denote by *s-item* an item of size s. When an item arrives, it must be assigned to a unit-sized bin immediately without exceeding the bin capacity. At any time, the *load* of a bin is the total size of items currently assigned to that bin that have not yet departed. We denote by *ℓ-bin* a bin of load ℓ. Migration is not allowed, i.e., once an item is assigned to a bin, it cannot be moved to another bin. This also applies to the optimal offline algorithm. The objective is to minimize the maximum number of bins used over all time.

When we discuss how items are packed, we use the following notations:

– Item configuration ψ: y_{*z} describes a load y with $\frac{y}{z}$ items of size z, e.g., $\frac{1}{2}_{*\epsilon}$ means a load $\frac{1}{2}$ with $\frac{1}{2\epsilon}$ items of size ϵ. We skip the subscript when $y = z$.
– Bin configuration π: $(\psi_1, \psi_2, \cdots)$, e.g., $(\frac{1}{3}, \frac{1}{2}_{*\epsilon})$ means a bin has a load of $\frac{5}{6}$, with a $\frac{1}{3}$-item and an addition load $\frac{1}{2}$ with ϵ-items. In some cases, it is clearer to state the bin configuration in other ways, e.g., $(\frac{1}{2}, \frac{1}{2})$, instead of $1_{*\frac{1}{2}}$. Similarly, we will use $6 \times \frac{1}{6}$ instead of $1_{*\frac{1}{6}}$.
– Packing configuration ρ: $\{x_1{:}\pi_1, x_2{:}\pi_2, \cdots\}$ a packing where there are x_1 bins with bin configuration π_1, x_2 bins with π_2, and so on. E.g., $\{2k{:}1_{*\epsilon}, k{:}(\frac{1}{3}, \frac{1}{2}_{*\epsilon})\}$ means $2k$ bins are each packed with load 1 with ϵ-items and another k bins are each packed with a $\frac{1}{3}$-item and an addition load $\frac{1}{2}$ with ϵ-items.
– It is sometimes more convenient to describe a packing as $x{:}f(i)$, for $1 \le i \le x$, which means that there are x bins with different load, one bin with load $f(i)$ for each i. E.g., $k{:}\frac{1}{2}{-}i\delta$, for $1 \le i \le k$, means that there are k bins and one bin with load $\frac{1}{2}{-}i\delta$ for each i.

3 Op-Inc and Op-Comp

In this section, we discuss a process that the adversary uses to force an online algorithm $\mathcal{A}$ to open new bins. The adversary releases items of slightly different sizes in each stage and uses items of complementary sizes in different stages. Two operations are designed, namely, Op-Inc and Op-Comp. Op-Inc forces $\mathcal{A}$ to open some bins each with one item (of size $< \frac{1}{2}$) and the size of items is strictly increasing. Op-Comp then bases on the bins opened by Op-Inc and releases items of complementary size. This is to ensure that an item released in Op-Inc can be packed with a corresponding item released in Op-Comp into the same bin by an optimal offline algorithm. In the adversary, a stage of Op-Inc is associated with a corresponding stage of Op-Comp, but not necessarily consecutive, e.g., in one of the cases, Op-Inc is in Stage 1 and the corresponding Op-Comp is in Stage 4.

3.1 Operation Op-Inc

The aim of Op-Inc is to make $\mathcal{A}$ open at least s more bins, for some $s > 0$, such that each new bin contains one item with item size increasing over the s bins.
Pre-condition. Consider any value $0 < x < \frac{1}{2}$. Let h be the number of x-items that can be packed in existing bins.
Items to be involved. The items to be released have size in the range $[x, x + \epsilon]$, for some small ϵ, such that $x + \epsilon < \frac{1}{2}$. A total of $h + \lfloor \frac{s}{x} \rfloor$ items are to be released.
Outcome. $\mathcal{A}$ opens at least s more new bins with increasing load in each new bin and the load of current bins remains unchanged.

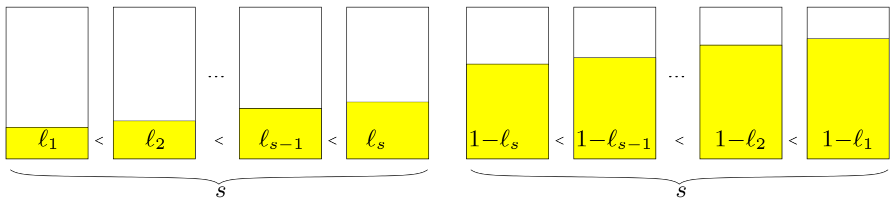

Fig. 1. Op-Comp: Assuming $k = 0$. The s bins on the left are bins created by Op-Inc. The s new bins on the right are due to Op-Comp. Note that each existing item has a complementary new item such that the sum of size is 1.

The adversary. The adversary releases items of size x, $x + \frac{\epsilon}{s}$, $x + \frac{2\epsilon}{s}$, $\cdots$. Let $z_i = x + \frac{i\epsilon}{s}$. In each step i, the adversary releases z_i-items until $\mathcal{A}$ opens a new bin. We stop releasing items when $h + \lfloor \frac{s}{x} \rfloor$ items have been released in total. By the definition of h, s and x, $\mathcal{A}$ would have opened at least s new bins. We then let z_i-items depart except exactly one item of size z_i, for $0 \le i < s$, in the i-th new bin opened by $\mathcal{A}$.

Using Op-Inc. When we use Op-Inc later, we simply describe it as Op-Inc releasing $h + \lfloor \frac{s}{x} \rfloor$ items with the understanding that it works in phases and that items depart at the end.

3.2 Operation Op-Comp

Op-Comp is designed to work with Op-Inc and assumes that there are s existing bins each with load in the range $[x, y]$ where $x < y < \frac{1}{2}$. The outcome of Op-Comp is that $\mathcal{A}$ opens s more bins. Figure 1 gives an illustration.

Pre-condition. Consider two values $x < y < \frac{1}{2}$. Suppose $\mathcal{A}$ uses s bins with load $x = \ell_1 < \ell_2 < \cdots < \ell_s = y$. Let $\ell = \sum_{1 \le i \le s} \ell_i$. Furthermore, suppose there are some additional bins with load smaller than x. Let h be the number of $(1-y)$-items that can be packed in other existing bins with load less than x.

Items to be involved. The items to be released have size in the range $[1-y, 1-x]$. Note that $1 - x > 1 - y > \frac{1}{2}$. In each step i, for $1 \le i \le s$, the number of $(1 - \ell_{s+1-i})$-items released is at most $h + s + 2 - i$.

Outcome. $\mathcal{A}$ opens s more bins, each with an item $1 - \ell_{s+1-i}$, for $1 \le i \le s$.

The adversary. Starting from the largest load ℓ_s, we release items of size $1-\ell_s$ until $\mathcal{A}$ opens a new bin. At most $h + s + 1$ items are needed. Then we let all $(1-\ell_s)$-items depart except the one packed in the new bin. In general, in Step i, for $1 \le i \le s$, we release items of size $1-\ell_{s+1-i}$ until $\mathcal{A}$ opens a new bin. Note that such items can only be packed in the first $s + 1 - i$ bins and so at most $h + s + 2 - i$ items are required to force $\mathcal{A}$ to open another bin. We then let all $(1-\ell_{s+1-i})$-items depart except the one packed in the new bin.

Using Op-Comp. Similar to Op-Inc, when we use Op-Comp later, we describe it as Op-Comp with h and s and the understanding is that it works in phases and there are items released and departure in between. Note that the ℓ_i- and $(1 - \ell_i)$-items are complementary and the optimal offline algorithm would pack each pair of complementary items in the same bin.

3.3 A 2.5 lower bound using Op-Inc and Op-Comp

We demonstrate how to use Op-Inc and Op-Comp by showing that we can obtain a 2.5 lower bound as in [5] using the two operations in a much simpler way.

Let k be some large even integer, $\epsilon = \frac{1}{k}$, and $\delta = \frac{\epsilon}{k+1}$. The adversary works in stages. In Stage 1, we release $\frac{k}{\epsilon}$ items of size ϵ. Any online algorithm $\mathcal{A}$ uses at least k bins. Let items depart until the configuration is $\{k{:}\epsilon\}$. In Stage 2, we aim to force $\mathcal{A}$ to use $\frac{k}{2}$ new bins. We use Op-Inc to release at most $2k$ items of size in $[\frac{1}{2}-\frac{k}{2}\delta, \frac{1}{2}-\delta]$. For each existing ϵ-bin, at most one such new items can be packed because $1-k\delta+\epsilon > 1$. The parameters for Op-Inc are therefore $x = \frac{1}{2}-\frac{k}{2}\delta$, $h = k$ and $s = \frac{k}{2}$. The configuration of $\mathcal{A}$ becomes $\{k{:}\epsilon, \frac{k}{2}{:}\frac{1}{2}-i\delta\}$, for $1 \le i \le \frac{k}{2}$. In Stage 3, we aim to force $\mathcal{A}$ to use $\frac{k}{2}$ new bins. We use Op-Comp to release items of size in the range $x = \frac{1}{2}+\delta$ to $y = \frac{1}{2}+\frac{k}{2}\delta$. At most one such item can be packed in the bins with an ϵ-item, i.e., $h = k$. The second $\frac{k}{2}$ bins contains items of complementary size to the items released in Stage 3, i.e., $s = \frac{k}{2}$. Note that at any time during Op-Comp, at most $\frac{3k}{2}+1$ items are released. $\mathcal{A}$ needs to open at least $\frac{k}{2}$ new bins with the configuration $\{k{:}\epsilon, \frac{k}{2}{:}\frac{1}{2}-i\delta, \frac{k}{2}{:}\frac{1}{2}+i\delta\}$, for $1 \le i \le \frac{k}{2}$. In the final stage, we release $\frac{k}{2}$ items of size 1 and $\mathcal{A}$ needs a new bin for each of these items. The total number of bins used by $\mathcal{A}$ becomes $\frac{5k}{2}$.

On the other hand, the optimal algorithm $\mathcal{O}$ can use $k+2$ bins to pack all items as follows and hence the competitive ratio is at least 2.5. In Stage 1, all the ϵ-items that never depart are packed in one bin and the rest in $k-1$ bins. In Stage 2, the new items are packed in k bins, with the $\frac{k}{2}$ bins with size $\frac{1}{2}-i\delta$, for $1 \le i \le \frac{k}{2}$, that never depart each packed in one bin, and the remaining $\frac{3k}{2}$ items in the remaining space. At the end of the stage, only one item is left in each of the first $\frac{k}{2}$ bins and the second $\frac{k}{2}$ bins are freed for Stage 3. In Stage 3, the complementary items that do not depart are packed in the corresponding $\frac{k}{2}$ bins, and the remaining in at most $\frac{k}{2}+1$ bins. Finally in Stage 4, the 1-items are packed in the $\frac{k}{2}$ bins freed in Stage 3.

4 The 8/3 Lower Bound

We give an adversary such that at any time, the total load of items released and not departed is at most $6k + O(1)$, for some large integer k. We prove that any online algorithm $\mathcal{A}$ uses $16k$ bins, while the optimal offline algorithm $\mathcal{O}$ uses at most $6k + O(1)$ bins. Then, the competitive ratio of $\mathcal{A}$ is at least $\frac{8}{3}$. The adversary works in stages and uses Op-Inc and Op-Comp in pairs. Let n_i be the number of new bins used by $\mathcal{A}$ in Stage i. Let $\epsilon = \frac{1}{6k}$ and $\delta = \frac{\epsilon}{16k}$.

In Stage 0, the adversary releases $\frac{6k}{\epsilon}$ items of size ϵ, with total load $6k$. It is clear that $\mathcal{A}$ needs at least $6k$ bins, i.e., $n_0 \ge 6k$. We distinguish between two cases: $n_0 \ge 8k$ and $8k > n_0 \ge 6k$. We leave the details of the easier first case in the full paper, and we consider only the complex second case in this paper.

Case 2: $6k \le n_0 < 8k$.

This case involves three subcases. We make two observations about the load of the n_0 bins. If less than $4k$ bins have load at least $\frac{1}{2} + \epsilon$, then the total load of all bins is at most $(4k-1) + 4k/2 = 6k-1$, contradicting the fact that total load of items released is $6k$. Similarly, if less than $5k$ bins have load at least $\frac{1}{4} + \epsilon$, then the total load of all bins is at most $(5k-1) + 3k/4 < 6k$, leading to a contradiction.

Observation 1 *At the end of Stage* 0 *of Case* 2, *(i) at least* $4k$ *bins have load at least* $\frac{1}{2} + \epsilon$; *(ii) at least* $5k$ *bins have load at least* $\frac{1}{4} + \epsilon$.

Stage 1. We aim at $n_1 \ge 2k$. We let ϵ-items depart until the configuration of $\mathcal{A}$ becomes

$$\{4k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}, k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}\epsilon\} \ ,$$

with $6k$ bins and a total load of $9k/4{+}O(1)$. We then use Op-Inc with $x = \frac{1}{4}{+}\delta$, $h = 8k$, and $s = 2k$. The first $4k$ bins can pack at most one x-item, the next k bins at most two, and the last k bins at most three, i.e., $h = 9k$. Any new bin can pack at most three items, implying that Op-Inc releases $15k = h + 3s$ items of increasing sizes, from $\frac{1}{4}{+}\delta$ to at most $\frac{1}{4}{+}15k\delta$. According to Op-Inc, $\mathcal{A}$ opens at least $2k$ bins, i.e., $n_1 \ge 2k$. We consider two subcases: $n_1 \ge 4k$ and $2k \le n_1 < 4k$.

Case 2.1: $6k \le n_0 < 8k$ and $n_1 \ge 4k$. In this case, we have $10k \le n_0 + n_1$.
Stage 2. We aim at $n_2 \ge 4k$. The configuration after Op-Inc becomes

$$\{4k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}\epsilon, 4k{:}\frac{1}{4}{+}i\delta\} \ , \ \text{for } 1 \le i \le 4k,$$

with $10k$ bins and a total load of $9k/4 + O(1)$. Note that in the last $4k$ bins, the load increases by δ from $\frac{1}{4}{+}\delta$ to $\frac{1}{4}{+}4k\delta$. We now use Op-Comp with $x = \frac{1}{4}{+}\delta$, $y = \frac{1}{4}{+}4k\delta$, $h = k$, and $s = 4k$. I.e., Op-Comp releases items of sizes from $\frac{3}{4}{-}4k\delta$ to $\frac{3}{4}{-}\delta$ and at any time, at most $5k + 1$ items are needed. None of these items can be packed in the first $5k$ bins, and only one can be packed in the next k bins, i.e., $h = k$ as said. According to Op-Comp, $\mathcal{A}$ requires $4k$ new bins.
Stage 3. We aim at $n_3 = 2k$. We let items depart until the configuration becomes

$$\{4k{:}\epsilon, k{:}\epsilon, k{:}\epsilon, 4k{:}\frac{1}{4}{+}i\delta, 4k{:}\frac{3}{4}{-}i\delta\} \ , \ \text{for } 1 \le i \le 4k,$$

with $14k$ bins and a load of $4k + O(1)$. We further release $2k$ items of size 1. $\mathcal{A}$ needs to open $2k$ new bins. In total, $\mathcal{A}$ uses $6k + 4k + 4k + 2k = 16k$ bins.

We note that each item with size $\frac{1}{4}{+}i\delta$ has a corresponding item $\frac{3}{4}{-}i\delta$ such that the sum of sizes is 1. This allows the optimal offline algorithm to have a better packing. The details will be given in the full paper.

Lemma 1. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage* 0 *and at least $4k$ bins in Stage* 1, *then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k + 4$ bins.*

Case 2.2: $6k \le n_0 < 8k$ and $2k \le n_1 < 4k$. In this case, the Op-Inc in Stage 1 is paired with an Op-Comp in Stage 4 (not consecutively), and in between, there is another pair of Op-Inc and Op-Comp in Stages 2 and 3, respectively. Let m be the number of bins among the n_1 new bins that have been packed two items. We further distinguish two subcases: $m \ge 2k$ and $m < 2k$.

Case 2.2.1: $6k \le n_0 < 8k$, $2k \le n_1 < 4k$ and $m \ge 2k$. In this case, we have $8k \le n_0 + n_1 < 10k$ and $m \ge 2k$. We make an observation about the bins containing some ϵ-items. In particular, we claim that there are at least k bins that are packed with

– either one ϵ-item and at least two $(\frac{1}{4}{+}i\delta)$-items,
– or one $(\frac{1}{4}{+}i\delta)$-item plus at least a load of $(\frac{1}{4}{+}\epsilon)_{*\epsilon}$.

We note that in Stage 1, $15k$ items are released, at most three items can be packed in any of the $n_1 < 4k$ new bins, i.e., at most $12k$ items. So, at least $3k$ of them have to been packed in the first $6k$ bins. Let a and b be the number of bins in the first $5k$ bins (with load at least $\frac{1}{4}{+}\epsilon$) that are packed at least one $(\frac{1}{4}{+}i\delta)$-item; z_1, z_2, z_3 be the number of bins in the next k bins (with one ϵ-item) that are packed one, two, and three $(\frac{1}{4}{+}i\delta)$-items, respectively. Note that $z_1 + z_2 + z_3 = k$. Since $3k$ items have to be packed in these bins, we have $a + 2b + z_1 + 2z_2 + 3z_3 \ge 3k$, hence $a + 2b + z_2 + 2z_3 \ge 2k$. The last inequality implies that $a + b + z_2 + z_3 \ge k$ and the claim holds.

Observation 2 *At the end of Stage* 1 *of Case* 2.2.1*, at least k bins are packed with either one ϵ-item and at least two $(\frac{1}{4}{+}i\delta)$-items, or one $(\frac{1}{4}{+}i\delta)$-item plus at least a load of $(\frac{1}{4}{+}\epsilon)_{*\epsilon}$.*

Stage 2. We aim at $n_2 \ge 2k$. Let $z = z_2 + z_3$. We let items depart until the configuration becomes

$$\{3k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}, k{-}z{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta), z{:}(\epsilon, \frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta), 2k{:}\epsilon, 2k{:}(\frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta)\} \ ,$$

with $8k$ bins and a total load of $3k + O(1)$.

Let $\delta' = \frac{\delta}{16k}$. We use Op-Inc with $x = \frac{1}{2}{-}6k\delta'$, $h = 2k$, and $s = 2k$. The x-items can only be packed in the $2k$ bins with load ϵ, at most one item in one bin, i.e., $h = 2k$. Any new bin can pack at most two, implying that Op-Inc releases $6k = h + 2s$ items of increasing sizes, from $\frac{1}{2}{-}6k\delta'$ to at most $\frac{1}{2}{-}\delta'$. According to Op-Inc, $\mathcal{A}$ has to open at least $2k$ new bins, i.e., $n_2 \ge 2k$.

Stage 3. In this stage, we aim at $n_3 \ge 2k$. We use Op-Comp which corresponds to Op-Inc in Stage 2. We let items depart until the configuration becomes

$$\{3k{:}(\tfrac{1}{2}{+}\epsilon)_{*\epsilon}, k{-}z{:}((\tfrac{1}{4}{+}\epsilon)_{*\epsilon}, \tfrac{1}{4}{+}i\delta), z{:}(\epsilon, \tfrac{1}{4}{+}i\delta, \tfrac{1}{4}{+}i\delta), 2k{:}\epsilon, 2k{:}(\tfrac{1}{4}{+}i\delta, \tfrac{1}{4}{+}i\delta), 2k{:}\tfrac{1}{2}{-}i\delta'\} \ ,$$

with $10k$ bins and a total load of $4k + O(1)$. We then use Op-Comp with $x = \frac{1}{2}{-}6k\delta'$, $y = \frac{1}{2}{-}5k\delta'$, $h = 2k$, and $s = 2k$. I.e., we release items of increasing size from $\frac{1}{2}{+}5k\delta'$ to $\frac{1}{2}{+}6k\delta'$, and at any time, at most $4k + 1$ items are needed. The $2k$ bins of load ϵ can pack one such item. Suppose there are w, out of $2k$,

ϵ-bins that are not packed with a $\frac{1}{2}+i\delta'$-item. According to Op-Comp, $\mathcal{A}$ has to open $2k+w$ new bins.

Stage 4. In this stage, we aim at $n_4 \geq 2k-w$. We use Op-Comp which corresponds to Op-Inc in Stage 1. We let items depart until the configuration is

$$\{3k{:}(\tfrac{1}{4}+\epsilon)_{*\epsilon}, k-z{:}(\tfrac{1}{4}+\epsilon)_{*\epsilon}, z{:}(\epsilon, \tfrac{1}{4}+i\delta), 2k-w{:}(\epsilon, \tfrac{1}{2}+i\delta'), w{:}\epsilon, 2k{:}\tfrac{1}{4}+i\delta, 2k{:}\tfrac{1}{2}-i\delta', 2k+w{:}\tfrac{1}{2}+i\delta', \}\ ,$$

with $12k+w$ bins and a total load of $9k/2+O(1)$. We then use Op-Comp with $x = \frac{1}{4}+\delta$, $y = \frac{1}{4}+2k\delta$, $h = w$, and $s = 2k-w$. I.e., we release items of sizes from $\frac{3}{4}-2k\delta$ to $\frac{3}{4}-\delta$ and at any time, at most $2k+1$ items are needed. Only w ϵ-bins can pack such item, i.e., $h = w$ as said. According to Op-Comp, $\mathcal{A}$ has to open $2k-w$ new bins.

Stage 5. In this final stage, we aim at $n_5 = 2k$. We let items depart until the configuration is

$$\{3k{:}\epsilon, k-z{:}\epsilon, z{:}\epsilon, 2k-w{:}\epsilon, w{:}\epsilon, 2k{:}\frac{1}{4}+i\delta, 2k{:}\frac{1}{2}-i\delta', 2k+w{:}\frac{1}{2}+i\delta', 2k-w{:}\frac{3}{4}-i\delta, \}\ ,$$

with $14k$ bins and a total load of $4k - \frac{w}{4} + O(1)$. Finally, we release $2k$ items of size 1 and $\mathcal{A}$ has to open $2k$ new bins. In total, $\mathcal{A}$ uses $3k + (k-z) + z + (2k-w) + w + 2k + 2k + (2k+w) + (2k-w) + 2k = 16k$. The packing of $\mathcal{O}$ will be given in the full paper.

Lemma 2. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage 0, $[2k, 4k)$ bins in Stage 1, and $m \geq 2k$, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k+3$ bins.*

Case 2.2.2: $6k \leq n_0 < 8k$, $2k \leq n_1 < 4k$ and $m < 2k$. We recall that in Stage 1, $15k$ items of size $\frac{1}{4}+i\delta$ are released and $\mathcal{A}$ uses $[2k, 4k)$ new bins for these items.

Observation 3 *(i) At most $8k$ items of size $\frac{1}{4}+i\delta$ can be packed to the n_1 new bins. (ii) At least k of the $\{k{:}\frac{1}{4}+i\delta, k{:}\epsilon\}$ bins have load more than $\frac{1}{2}$. (iii) At least $2k$ of the $\{4k{:}(\frac{1}{2}+\epsilon)_{*\epsilon}\}$ bins are packed with at least one $(\frac{1}{2}+i\delta)$-item.*

Let z_1 and z_2 be the number of new bins that are packed one and at least two, respectively, $(\frac{1}{4}+i\delta)$-items The following observation gives a bound on z.

Observation 4 *(i) At most $9k$ items of size $\frac{1}{4}+i\delta$ can be packed in existing bins. (ii) $z_2 \geq k$. (iii) $z_1 \geq 3(2k-z_2)$.*

Stage 2. We target $n_2 \geq z_2$. We let items depart until the configuration becomes

- $2k{:}(\frac{1}{2}+\epsilon)_{*\epsilon}$,
- $2k{:}((\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta)$, this is possible because of Observation 3(iii),
- $x{:}(\epsilon, \frac{1}{4}+i\delta, \frac{1}{4}+i\delta)$,
- $k-x{:}((\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta)$, this is possible because of Observation 3(ii),
- $k{:}\epsilon$,

- z_2:$(\frac{1}{4}+i\delta, \frac{1}{4}+i\delta)$, this is possible because of Observation 4(ii),
- $2(2k-z_2)$:$(\frac{1}{4}+i\delta)$, this is possible because of Observation 4(iii),

with $10k - z_2$ bins and a total load of $7k/2 + O(1)$. We then use Op-Inc with $x = \frac{1}{2}-5k\delta'$, $h = 5k - 2z_2$ and $s = z_2$. The x-items can only be packed in k of ϵ-bins and $2(2k - z_2)$ of $(\frac{1}{4}+i\delta)$-bins, i.e., $h = k + 2(2k - z_2) = 5k - 2z_2$ as said. Any new bin can pack at most two, implying that Op-Inc releases $5k = h + 2s$ items of increasing sizes from $\frac{1}{2}-5k\delta'$ to $\frac{1}{2}-\delta'$. According to Op-Inc, $\mathcal{A}$ has to open at least z_2 bins, i.e., $n_2 \geq z_2$.
Stage 3. We target $n_3 \geq z_2$. We let items depart until the configuration becomes

$$\{2k{:}(\frac{1}{2}+\epsilon)_{*\epsilon}, 2k{:}((\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta), x{:}(\epsilon, \frac{1}{4}+i\delta, \frac{1}{4}+i\delta), k-x{:}((\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta), k{:}\epsilon,$$
$$z_2{:}(\frac{1}{4}+i\delta, \frac{1}{4}+i\delta), 2(2k-z_2){:}\frac{1}{4}+i\delta, z_2{:}\frac{1}{2}-i\delta'\} \ ,$$

with $10k$ bins and a total load of $7k/2 + z_2/2 + O(1)$. We use Op-Comp with $s = z_2$ to release items of increasing size from $\frac{1}{2}+\delta'$. These items can only be packed in ϵ-bins (k of them) and $(\frac{1}{4}+i\delta)$-bins ($2(2k - z_2)$ of them). At any time, at most $(5k - z_2) + 1$ items are needed. According to Op-Comp, $\mathcal{A}$ has to open z_2 bins, i.e., $n_3 \geq z_2$.
Stage 4. We target $n_4 \geq (4k - z_2)$. We let items depart until the configuration becomes

$$\{4k-x{:}(\frac{1}{4}+\epsilon)_{*\epsilon}, k+x{:}(\epsilon, \frac{1}{4}+i\delta), k{:}\epsilon, 4k-z_2{:}\frac{1}{4}+i\delta, z_2{:}\frac{1}{2}-i\delta', z_2{:}\frac{1}{2}+i\delta'\} \ ,$$

with $10k+z_2+O(1)$ bins and a total load of $9k/4+3z_2/4$. We then use Op-Comp with $s = 4k - z_2$ and items of increasing size $\frac{3}{4}-i\delta$. Using similar ideas as before, $\mathcal{A}$ has to open $(4k - z_2)$ new bins.
Stage 5. We target $n_5 = 2k$. We let items depart until the configuration becomes

$$\{4k-x{:}\epsilon, k+x{:}\epsilon, k{:}\epsilon, 4k-z_2{:}\frac{1}{4}+i\delta, z_2{:}\frac{1}{2}-i\delta', z_2{:}\frac{1}{2}+i\delta', 4k-z_2{:}\frac{3}{4}-i\delta, \} \ ,$$

with $14k$ bins and a total load of $4k + O(1)$. We finally release $2k$ items of size 1 and $\mathcal{A}$ has to open $2k$ new bins. In total $\mathcal{A}$ uses $6k + 8k + 2k = 16k$ bins. The packing of $\mathcal{O}$ will be given in the full paper.

Lemma 3. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage 0, $[2k, 4k)$ bins in Stage 1, and $m < 2k$, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k + 5$ bins.*

Theorem 5. *No online algorithm can be better than 8/3-competitive.*

5 Conclusion

We have derived a $8/3 \ \sim 2.666$ lower bound on the competitive ratio for dynamic bin packing, improving the best known 2.5 lower bound [5]. We designed two operations that release items of slightly increasing sizes and items with complementary sizes. These operations make a more systematic approach to release items: the type of item sizes used in a later case is a superset of those used in an earlier case. This is in contrast to the previous 2.5 lower bound in [5] in which rather different sizes are used in different cases. Furthermore, in each case, we use one or two pairs of Op-Inc and Op-Comp, which makes the structure clearer and the proof easier to understand. We also show that the new operations defined lead to a much easier proof for a 2.5 lower bound. An obvious open problem is to close the gap between the upper and lower bounds.

References

1. J. Balogh, J. Békési, G. Galambos, and G. Reinelt. Lower bound for the online bin packing problem with restricted repacking. *SIAM J. Comput.*, 38:398–410, 2008.
2. A. Bar-Noy, R. E. Ladner, and T. Tamir. Windows scheduling as a restricted version of bin packing. In J. I. Munro, editor, *SODA*, pages 224–233. SIAM, 2004.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
4. J. W.-T. Chan, T. W. Lam, and P. W. H. Wong. Dynamic bin packing of unit fractions items. *Theoretical Computer Science*, 409(3):172–206, 2008.
5. J. W.-T. Chan, P. W. H. Wong, and F. C. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Algorithmica*, 53(2):172–206, 2009.
6. E. G. Coffman, Jr., G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Combinatorial analysis. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization.* Kluwer Academic Publishers, 1998.
7. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.
8. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Bin packing approximation algorithms: A survey. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS, 1996.
9. J. Csirik and G. J. Woeginger. On-line packing and covering problems. In A. Fiat and G. J. Woeginger, editors, *On-line Algorithms—The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 147–177. Springer, 1996.
10. L. Epstein and M. Levy. Dynamic multi-dimensional bin packing. *Journal of Discrete Algorithms*, 8:356–372, 2010.
11. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.
12. X. Han, C. Peng, D. Ye, D. Zhang, and Y. Lan. Dynamic bin packing with unit fraction items revisited. *Information Processing Letters*, 110:1049–1054, 2010.
13. Z. Ivkovic and E. L. Lloyd. A fundamental restriction on fully dynamic maintenance of bin packing. *Inf. Process. Lett.*, 59(4):229–232, 1996.
14. S. S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.
15. A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.
16. P. W. H. Wong and F. C. C. Yung. Competitive multi-dimensional dynamic bin packing via L-shape bin packing. In *Proc. of Workshop on Approximation and Online Algorithms (WAOA)*, pages 242–254, 2009.