On Dynamic Bin Packing: An Improved Lower Bound and Resource Augmentation Analysis

Wun-Tat Chan[*] Prudence W.H. Wong[†] Fencol C.C. Yung[‡]

Abstract

We study the dynamic bin packing problem introduced by Coffman, Garey and Johnson [7]. This problem is a generalization of the bin packing problem in which items may arrive and depart from the packing dynamically. The main result in this paper is a lower bound of 2.5 on the achievable competitive ratio, improving the best known 2.428 lower bound [3], and revealing that packing items of restricted form like unit fractions (i.e., of size $1/k$ for some integer k), which can guarantee a competitive ratio 2.4985 [3], is indeed easier.

We also investigate the resource augmentation analysis on the problem where the on-line algorithm can use bins of size b (> 1) times that of the optimal off-line algorithm. An interesting result is that we prove $b = 2$ is both necessary and sufficient for the on-line algorithm to match the performance of the optimal off-line algorithm, i.e., achieve 1-competitiveness. Further analysis is made to give a trade-off between the bin size multiplier b and the achievable competitive ratio.

1 Introduction

Bin packing is a classical combinatorial optimization problem (see the surveys [5, 8, 11]). The objective is to pack a sequence of items into a minimum number of bins such that the total size of the items in a bin does not exceed the bin capacity. The on-line version of the problem assumes that items may arrive at arbitrary time and no advance information is known about the items not yet arrived. *Dynamic bin packing* was introduced as a generalization of the on-line bin packing by Coffman, Garey and Johnson [7]. In this generalization, items may also depart at arbitrary time and both on-line and off-line algorithms are not allowed to move items from one bin to another. The goal is to minimize the maximum number of bins used over all time.

The performance of an on-line algorithm $\mathcal{A}$ is generally measured by its *competitive ratio* [2]. For our problem where a sequence of item arrivals and departures is given, the competitive ratio c is the worst case ratio between the maximum number of bins used by $\mathcal{A}$ over all time and the maximum number of bins used by the optimal off-line algorithm (which knows the whole sequence in advance) over all time. Algorithm $\mathcal{A}$ is said to be *c-competitive*.

Coffman, Garey and Johnson [7] proved that the lower bound on the competitive ratio of any on-line algorithm on dynamic bin packing is 2.388[1]. They also showed that a modified version of first-fit is 2.788-competitive [7]. Chan, Lam and Wong [3] improved the lower bound to 2.428 by considering only *unit fraction items*, where a unit fraction item is an item of size $1/k$ for some integer k. They also showed that for packing unit fraction items only, first-fit is 2.4985-competitive [3]. A natural question arises: Is packing items of restricted form, such as unit fraction items, as difficult as packing items of general form? In another aspect, *resource augmentation analysis* [15] has been studied in the context of on-line bin packing [12, 13], in

[*]Department of Computer Science, University of Hong Kong, Hong Kong, wtchan@cs.hku.hk

[†]Department of Computer Science, University of Liverpool, UK, pwong@csc.liv.ac.uk

[‡]Department of Computer Science, University of Hong Kong, Hong Kong, ccyung@graduate.hku.hk

[1]A variant of the problem is to assume a stronger off-line algorithm that can repack the current set of items into the minimum possible number of bins each time a new item arrives, in which case a stronger lower bound 2.5 is achieved [7].

which an on-line algorithm can use bins of size b (> 1) times that of the optimal off-line algorithm. To our knowledge, there is no previous work on resource augmentation analysis for dynamic bin packing. We address the above questions in this paper.

Our contributions. This paper presents the following results on dynamic bin packing.

1. We push up the lower bound on competitive ratio from 2.428 [3] to 2.5[2], giving a negative answer to the question that packing unit fraction items is as difficult as packing general items because packing unit fraction items attains a competitive ratio 2.4985 [3] (< 2.5).
2. We investigate on resource augmentation analysis, showing an interesting result that doubling the bin size for the on-line algorithm is both necessary and sufficient to match the performance of the optimal off-line algorithm, i.e., to attain 1-competitiveness. Further analysis is made to give a trade-off between the bin size multiplier b (for $1 < b \leq 2$) and the achievable competitive ratio.

Related work. There is a long history of results for the classical bin packing problem and its variants [5, 8, 11]. The best upper bound and lower bound on the competitive ratio for on-line bin packing to date are 1.58889 [16] and 1.54014 [17], respectively. The upper bound reveals that dynamic bin packing is more difficult than on-line bin packing. For both dynamic and on-line bin packing, items of various restricted forms have been studied, which include unit fraction items [1, 3], items of divisible sizes [6] (where each possible item size can be divided by the next smaller item size), and items of discrete sizes [4, 9, 10] (where possible item sizes are $\{1/k, 2/k, \cdots, j/k\}$ for some $1 \leq j \leq k$). Resource augmentation analysis for on-line bin packing has been studied [12, 13]; matching upper and lower bounds (up to an additive constant) are given for bounded space bin packing [12] in which there is a limit on the opened bins that can be used at any time; better upper bound has been derived for (unbounded space) on-line bin packing [13]. Ivkovic and Lloyd studied the *fully dynamic bin packing problem* [14], which is a variant of dynamic bin packing that allows repacking of items for each item arrival or departure. They gave a 1.25-competitive on-line algorithm for the problem [14].

Organization of the paper. Section 2 gives some preliminaries necessary for discussion. In Section 3, we present the 2.5 lower bound. In Section 4, we show that doubling the bin size is both necessary and sufficient to achieve 1-competitiveness. In Section 5, we study the trade-off between bin size and competitive ratio. Finally, we give some concluding remarks in Section 6.

2 Preliminaries

In this section, we give a precise definition of the dynamic bin packing problem and the necessary notations for further discussion. 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 bin immediately without exceeding the capacity of the assigned bin. 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 in the sense that once an item is assigned to a bin, it cannot be moved to another bin. The objective is to minimize the maximum number of bins used over all time.

In the resource augmentation analysis (Sections 4 and 5), an on-line algorithm $\mathcal{A}$ is given size-b bins with $1 \leq b \leq 2$, while the optimal off-line algorithm uses size-1 bins. Consider any input sequence σ. Let $\mathcal{A}_b(\sigma, t)$ denote the number of size-b bins used at time t by $\mathcal{A}$, similarly,

[2] As we mentioned before, there was a 2.5 lower bound [7] when the off-line algorithm can repack, our result implies we can now achieve the same bound even when the off-line algorithm does not repack.

we have $\mathcal{O}_1(\sigma, t)$ for the optimal off-line algorithm. $\mathcal{A}$ is said to be c-competitive if there exists a constant k such that for any input sequence σ, $\max_t \mathcal{A}_b(\sigma, t) \leq c \cdot \max_t \mathcal{O}_1(\sigma, t) + k$. We will consider several on-line algorithms: any-fit, first-fit, best-fit, and worst-fit. When an item arrives, all these algorithms pack the item into existing bins as long as there exists a bin that can accommodate the item; a new bin is only opened when no bins can accommodate the item. The algorithms differ in the rule to choose an existing bin for the newly arrived item. When a new item X with size s arrives, if there is any bin with load no more than $b-s$, the algorithms assign X to one of these bins as follows:

Any-fit (AF) assigns X to any of these bins arbitrarily.

First-fit (FF) assigns X to the one that is opened the earliest.

Best-fit (BF) assigns X to the heaviest load one; ties are broken arbitrarily.

Worst-fit (WF) assigns X to the lightest load one; ties are broken arbitrarily.

3 A 2.5 lower bound

In this section, we prove that no on-line algorithm can be better than 2.5-competitive. Consider any on-line algorithm $\mathcal{A}$. The adversary works in stages using items of various sizes including ϵ, $\frac{1}{6}$, $\frac{1}{3}$, $\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}$, $\frac{1}{2}+\frac{\epsilon}{4}$, $\frac{2}{3}$ and 1, where ϵ is a small constant to be defined later. Roughly speaking, the adversary first releases some items of a particular size. Depending on how $\mathcal{A}$ packs the items, the adversary lets some items depart and further releases some other items such that the total size of items present at any time is always upper bounded by a particular value. Yet the space released from some departed items cannot be reused for newly arrived items because of mismatch of item size and size of empty space, thus, enforcing $\mathcal{A}$ to use more new bins. We first describe the adversary and we will show in the appendix how the optimal off-line algorithm packs the items (without repacking) released by the adversary.

Let $\epsilon = \frac{1}{18k}$ for some large positive integer k. Recall that for any $s > 0$, $\ell > 0$, we denote by *s-item* an item of size s, and by *ℓ-bin* a bin of load ℓ. When we discuss how items are packed into bins, we denote by $x{:}\ell$ that there are x bins of load ℓ, and by $\{x_1{:}\ell_1, x_2{:}\ell_2, ...\}$ a packing configuration. The adversary releases items in stages such that $\mathcal{A}$ uses $45k$ bins while the total size of items at any time is no more than $18k + 2$. The item sizes are chosen carefully to allow the optimal off-line algorithm to use at most $18k+2$ bins. Then we have the following theorem.

Theorem 1. *No on-line algorithm is better than* 2.5*-competitive for dynamic bin packing.*

Consider any on-line algorithm $\mathcal{A}$. Let n_i be the number of new bins used by $\mathcal{A}$ in Stage i. In Stage 1, $\frac{18k}{\epsilon}$ items of size ϵ are released, thus, $n_1 \geq 18k$. We distinguish three cases: $n_1 \geq 24k$, $24k > n_1 \geq 21k$, and $21k > n_1 \geq 18k$.

Case 1: $n_1 \geq 24k$. In Stage 2, we let items depart until $\mathcal{A}$ is left with $\{24k{:}\epsilon\}$. Then $36k$ items of size $\frac{1}{2}$ are released. Note that the total size of items becomes $24k\epsilon + 18k < 18k + 2$. At most one new item can be packed into an existing bin, thus, $n_2 \geq (36k - 24k)/2 = 6k$. In Stage 3, items depart until $\{24k{:}\epsilon, 6k{:}\frac{1}{2}\}$. We then release $15k$ items of size 1, thus, $n_3 = 15k$. The total number of bins used by $\mathcal{A}$ equals $24k + 6k + 15k = 45k$.

Case 2: $24k > n_1 \geq 21k$. We make the following observations on the load of bins used by $\mathcal{A}$.

Observation 2. *If $24k > n_1 \geq 21k$, then $\mathcal{A}$ uses at least $6k$ bins of load at least $\frac{2}{3}+\epsilon$, $12k$ bins at least $\frac{1}{2}+\epsilon$, and $15k$ bins at least $\frac{1}{3}+\epsilon$ at the end of Stage* 1*.*

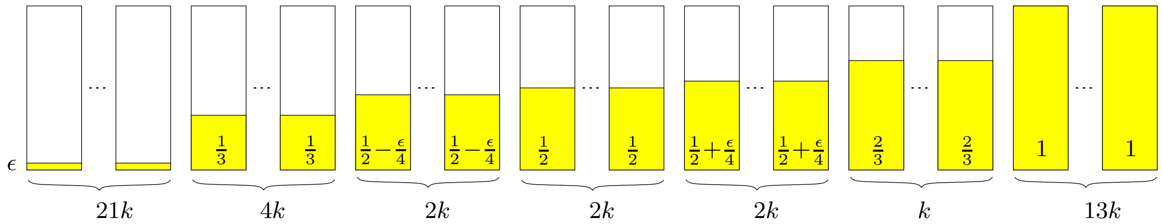

Figure 1: The final configuration achieved by the adversary in Case 2.2.

Proof. Assume that there are less than $6k$ bins of load at least $\frac{2}{3}+\epsilon$. The remaining bins has a maximum load of $\frac{2}{3}$. Then the maximum load that has been packed is $< 6k+(24k-6k)(\frac{2}{3}) = 18k$, contradicting that a total load of $18k$ has been released. The other cases are similar. $\square$

In Stage 2, we let items depart until $\{6k{:}(\frac{2}{3}+\epsilon), 6k{:}(\frac{1}{2}+\epsilon), 3k{:}(\frac{1}{3}+\epsilon), 6k{:}\epsilon\}$. We then release $30k$ items of size $\frac{1}{3}$. If $\mathcal{A}$ packs a $\frac{1}{3}$-item into some bin B of load $\frac{1}{2}+\epsilon$, we depart a total size of $\frac{1}{6}$ of ϵ-item from B, making its load become $\frac{2}{3}+\epsilon$. For every two such bins, we further release one $\frac{1}{3}$-item. Repeat departing ϵ-items of size $\frac{1}{6}$ and releasing items of size $\frac{1}{3}$ as long as $\mathcal{A}$ packs $\frac{1}{3}$-item into bin of load $\frac{1}{2}+\epsilon$. This process must terminate because once $\mathcal{A}$ packs a $\frac{1}{3}$-item into a bin, its load becomes $\frac{2}{3}+\epsilon$, meaning that it cannot accommodate another $\frac{1}{3}$-item. We assume that there are even number of $(\frac{1}{2}+\epsilon)$-bins that are packed with $\frac{1}{3}$-items; the other case is similar and we skip the details.

Let x and y be the number of $(\frac{1}{2}+\epsilon)$- and $(\frac{1}{3}+\epsilon)$-bins, respectively, that have been packed a $\frac{1}{3}$-item at the end of Stage 2. Let a_1 and a_2 be the number of new bins (used in Stage 2) that have been packed exactly one $\frac{1}{3}$-item and at least two $\frac{1}{3}$-items, respectively, i.e., $n_2 = a_1 + a_2$.

Observation 3. *The number of new bins used by $\mathcal{A}$ in Stage 2, i.e., $a_1 + a_2$, is at least $4k$.*

Proof. The total load of all bins is at most $(6k+x)(\frac{2}{3}+\epsilon)+(6k-x)(\frac{1}{2}+\epsilon)+y(\frac{2}{3}+\epsilon)+(3k-y)(\frac{1}{3}+\epsilon)+6k(\frac{2}{3}+\epsilon)+a_2+\frac{a_1}{3}$. This quantity must be at least $18k+21k\epsilon$. Using the property that $x \le 6k$ and $y \le 3k$, we can derive that $a_1 + a_2 \ge 4k$. $\square$

We further consider two sub-cases: $a_1 + a_2 \ge 9k$ and $9k > a_1 + a_2 \ge 4k$.

Case 2.1: $a_1 + a_2 \ge 9k$. In Stage 3, we depart items until $\{21k{:}\epsilon, 9k{:}\frac{1}{3}\}$. Finally, release $15k$ of 1-items, thus, $n_3 = 15k$. The total number of bins used by $\mathcal{A}$ becomes $21k + 9k + 15k = 45k$.

Case 2.2: $9k > a_1 + a_2 \ge 4k$. Figure 1 shows the target configuration the adversary aims to achieve. We observe a further bound on a_2, the proof is similar to Observation 3.

Observation 4. *For Case 2.2, we have $a_2 > k$.*

Proof. The maximum load that all bins can accommodate is at most $(6k+x)(\frac{2}{3}+\epsilon)+(6k-x)(\frac{1}{2}+\epsilon)+3k(\frac{2}{3}+\epsilon)+6k(\frac{2}{3}+\epsilon)+a_2+\frac{a_1}{3}$, which must be $\ge 18k+21k\epsilon$, Together with the inequalities $x \le 6k$ and $a_1 + a_2 < 9k$, we can derive that $a_2 > k$. $\square$

The remaining stages run as follows.

3. Depart items until
$$\{6k{:}(\tfrac{1}{2}+\epsilon),\ x{:}(\tfrac{1}{3}+\tfrac{1}{6}+\epsilon),\ (6k-x){:}(\tfrac{1}{2}+\epsilon),\ 3k{:}\epsilon,\ 6k{:}\epsilon,\ k{:}(\tfrac{1}{3}+\tfrac{1}{3}),\ 3k{:}\tfrac{1}{3}\ \}.$$
This is possible because $a_2 > k$ and $a_1 + a_2 \ge 4k$. We then release $20k$ items of size $\frac{1}{2}-\frac{\epsilon}{4}$. Since only bins of load ϵ and $\frac{1}{3}$ can accommodate one such item, $n_3 \ge (20k-12k)/2 = 4k$.
4. Depart items until
$$\{\ 6k{:}(\tfrac{1}{2}+\epsilon),\ x{:}(\tfrac{1}{3}+\tfrac{1}{6}+\epsilon),\ (6k-x){:}(\tfrac{1}{2}+\epsilon),\ 3k{:}\epsilon,\ 6k{:}\epsilon,\ k{:}(\tfrac{1}{3}+\tfrac{1}{3}),\ 3k{:}\tfrac{1}{3},\ 2k{:}(\tfrac{1}{2}-\tfrac{\epsilon}{4})\ \}.$$
Next we release $18k$ items of size $\frac{1}{2}$, making $n_4 \ge (18k-14k)/2 = 2k$.

5. Depart items until
   $\{\ 6k{:}(\frac{1}{2}+\epsilon),\ x{:}(\frac{1}{3}+\frac{1}{6}+\epsilon),\ (6k-x){:}(\frac{1}{2}+\epsilon),\ 3k{:}\epsilon,\ 6k{:}\epsilon,\ k{:}(\frac{1}{3}+\frac{1}{3}),\ 3k{:}\frac{1}{3},\ 2k{:}(\frac{1}{2}-\frac{\epsilon}{4}),\ 2k{:}\frac{1}{2}\ \}$.
   Releasing $16k$ items of size $\frac{1}{2}+\frac{\epsilon}{4}$ makes $n_5 \geq 16k - 14k = 2k$.
6. Depart items until
   $\{\ 6k{:}(\frac{1}{3}+\epsilon),\ x{:}(\frac{1}{3}+\epsilon),\ (6k-x){:}(\frac{1}{3}+\epsilon),\ 3k{:}\epsilon,\ 6k{:}\epsilon,\ k{:}(\frac{1}{3}+\frac{1}{3}),\ 3k{:}\frac{1}{3},\ 2k{:}(\frac{1}{2}-\frac{\epsilon}{4}),\ 2k{:}\frac{1}{2},\ 2k{:}(\frac{1}{2}+\frac{\epsilon}{4})\ \}$.
   We then release $13k$ items of size $\frac{2}{3}$, making $n_6 \geq 13k - 12k = k$.
7. Depart items until
   $$\{6k{:}\epsilon,\ x{:}\epsilon,\ (6k-x){:}\epsilon,\ 3k{:}\epsilon,\ 6k{:}\epsilon,\ k{:}\tfrac{1}{3},\ 3k{:}\tfrac{1}{3},\ 2k{:}(\tfrac{1}{2}-\tfrac{\epsilon}{4}),\ 2k{:}\tfrac{1}{2},\ 2k{:}(\tfrac{1}{2}+\tfrac{\epsilon}{4}),\ k{:}\tfrac{2}{3}\}.$$
   A final of $13k$ 1-items are released, making $n_7 = 13k$. Totally, $\mathcal{A}$ uses $21k + 4k + 2k + 2k + 2k + k + 13k = 45k$ bins.

Case 3: $21k > n_1 \geq 18k$. For the sake of simplicity, we let $m = 3k/2$. In other words, $14m > n_1 \geq 12m$. The following observation can be proved by contradiction, similar to Observation 2.

Observation 5. *If $14m > n_1 \geq 12m$, then $\mathcal{A}$ uses at least $8m$ bins with load at least $\frac{2}{3}+\epsilon$, $10m$ bins with at least $\frac{1}{2}+\epsilon$, and $11m$ bins with at least $\frac{1}{3}+\epsilon$ at the end of Stage 1.*

In Stage 2, we let items depart until $\{8m{:}(\frac{2}{3}+\epsilon), 2m{:}(\frac{1}{2}+\epsilon), m{:}(\frac{1}{3}+\epsilon), m{:}\epsilon\}$. Total size departed is $\frac{16m}{3} - 12m\epsilon$. Then release $32m$ items of size $\frac{1}{6}$. A $(\frac{2}{3}+\epsilon)$-bin can accommodate one item, $(\frac{1}{2}+\epsilon)$-bin two, $(\frac{1}{3}+\epsilon)$-bin three, and ϵ-bin five. Number of new bins required is at least $(32m - 8m - 4m - 3m - 5m)/6 = 2m$.

In Stage 3, depart items until: $\{8m{:}(\frac{2}{3}+\epsilon), 2m{:}(\frac{1}{2}+\epsilon), m{:}(\frac{1}{3}+\epsilon), m{:}\epsilon, 2m{:}\frac{1}{6}\}$. Total size departed is $5m$. Next release $15m$ items of size $\frac{1}{3}$. Similar to Case 2, if $\mathcal{A}$ packs a $\frac{1}{3}$-item into a bin of load $\frac{1}{2}+\epsilon$, we depart a total size of $\frac{1}{6}$ of ϵ-items from that bin. For every two such bins, we further release one $\frac{1}{3}$-item. Repeat departing ϵ-items of size $\frac{1}{6}$ and releasing items of size $\frac{1}{3}$ as long as $\mathcal{A}$ packs $\frac{1}{3}$-item into bin of load $\frac{1}{2}+\epsilon$.

Let x and y be the number of bins with load $\frac{1}{2}+\epsilon$ and $\frac{1}{6}$, respectively, that has been packed at least one $\frac{1}{3}$-item at the end of Stage 3. Let a_1 and a_2 be the number of new bins (used in Stage 3) that have been packed exactly one $\frac{1}{3}$-item and at least two $\frac{1}{3}$-items, respectively, i.e., $n_3 = a_1 + a_2$. Using the inequalities $(8m+x)(\frac{2}{3}+\epsilon) + (2m-x)(\frac{1}{2}+\epsilon) + m(\frac{2}{3}+\epsilon) + m(\frac{2}{3}+\epsilon) + y(\frac{2}{3}+\frac{1}{6}) + (2m-y)(\frac{1}{6}) + a_2 + \frac{a_1}{3} \geq 12m + 12m\epsilon$, $x \leq 2m$, and $y \leq 2m$, we have

Observation 6. *The number of new bins used by $\mathcal{A}$ in Stage 3, i.e., $a_1 + a_2$, is at least $2m$.*

Case 3.1: $a_1 + a_2 \geq 7m$. In Stage 4, depart items until $\{8m{:}\epsilon, 2m{:}\epsilon, m{:}\epsilon, m{:}\epsilon, 2m{:}\frac{1}{6}, 7m{:}\frac{1}{3}\}$. Total size departed equals $(9+\frac{1}{3})m > 9m$. Then release $9m$ items of size 1, requiring $9m$ new bins. Therefore, the total number of bins used by $\mathcal{A}$ equals $12m + 2m + 7m + 9m = 30m = 45k$.
Case 3.2: $7m > a_1 + a_2 \geq 2m$. Figure 2 shows the target configuration the adversary aims to achieve. Before moving on, we have one more observation.

Observation 7. *For Case 3.2, we have $a_2 + y \geq 2m$, i.e., $a_2 \geq 2m - y$.*

Proof. Using the inequalities $(8m+x)(\frac{2}{3}+\epsilon) + (2m-x)(\frac{1}{2}+\epsilon) + m(\frac{2}{3}+\epsilon) + m(\frac{2}{3}+\epsilon) + y(\frac{2}{3}+\frac{1}{6}) + (2m-y)(\frac{1}{6}) + a_2 + \frac{a_1}{3} \geq 12m + 12m\epsilon$, $x \leq 2m$ and $a_1 + a_2 < 7m$, we have $a_2 + y \geq 2m$. ☐

The remaining stages run as follows.

4. Depart items until
   $$\{\ 8m{:}(\tfrac{1}{2}+\epsilon),\ x{:}(\tfrac{1}{3}+\tfrac{1}{6}+\epsilon),\ (2m-x){:}(\tfrac{1}{2}+\epsilon),\ m{:}(\tfrac{1}{3}+\epsilon),\ m{:}\epsilon,\ y{:}(\tfrac{1}{6}+\tfrac{1}{3}),$$
   $$(2m-y){:}\tfrac{1}{6},\ (2m-y){:}(\tfrac{1}{3}+\tfrac{1}{3}),\ y{:}\tfrac{1}{3}\ \}.$$
   Next release $10m$ items of size $\frac{1}{2}-\frac{\epsilon}{4}$, making $n_4 \geq (10m - 4m - y)/2 \geq 2m$ because $y \leq 2m$.

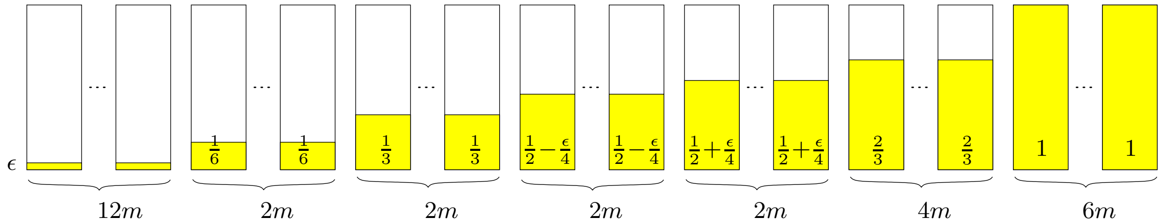

Figure 2: The final configuration achieved by the adversary in Case 3.2.

5. Depart items until
$$\{\ 8m{:}(\tfrac{1}{2}+\epsilon),\ x{:}(\tfrac{1}{3}+\tfrac{1}{6}+\epsilon),\ (2m-x){:}(\tfrac{1}{2}+\epsilon),\ m{:}(\tfrac{1}{3}+\epsilon),\ m{:}\epsilon,\ y{:}(\tfrac{1}{6}+\tfrac{1}{3}),\\ (2m-y){:}\tfrac{1}{6},\ (2m-y){:}(\tfrac{1}{3}+\tfrac{1}{3}),\ y{:}\tfrac{1}{3},\ 2m{:}(\tfrac{1}{2}-\tfrac{\epsilon}{4})\ \}.$$
We then release $8m$ items of size $\frac{1}{2}+\frac{\epsilon}{4}$, making $n_5 \geq 8m-m-m-(2m-y)-y-2m = 2m$.
6. Depart items until
$$\{\ 8m{:}(\tfrac{1}{3}+\epsilon),\ x{:}(\tfrac{1}{3}+\epsilon),\ (2m-x){:}(\tfrac{1}{3}+\epsilon),\ m{:}(\tfrac{1}{3}+\epsilon),\ m{:}\epsilon,\ y{:}(\tfrac{1}{6}+\tfrac{1}{3}),\\ (2m-y){:}\tfrac{1}{6},\ (2m-y){:}(\tfrac{1}{3}+\tfrac{1}{3}),\ y{:}\tfrac{1}{3},\ 2m{:}(\tfrac{1}{2}-\tfrac{\epsilon}{4}),\ 2m{:}(\tfrac{1}{2}+\tfrac{\epsilon}{4})\ \}.$$
Releasing $7m$ items of size $\frac{2}{3}$ makes $n_6 \geq 7m - m - (2m-y) - y = 4m$.
7. Depart items until
$$\{\ 12m{:}\epsilon,\ 2m{:}\tfrac{1}{6},\ 2m{:}\tfrac{1}{3},\ 2m{:}(\tfrac{1}{2}-\tfrac{\epsilon}{4}),\ 2m{:}(\tfrac{1}{2}+\tfrac{\epsilon}{4}),\ 4m{:}\tfrac{2}{3}\ \}.$$
Releasing another $6m$ items of size 1 makes $n_7 = 6m$.

Therefore, the total number of bins used by $\mathcal{A}$ is $12m+2m+2m+2m+2m+4m+6m=30m=45k$. We will show in the appendix that in all cases there is a schedule that uses at most $18k+2$ bins and thus Theorem 1 follows.

4 1-competitive if and only if size-2 bins are used

In this section, we show that using size-2 bins is necessary (Theorem 9) and sufficient (Theorem 8) to achieve 1-competitiveness.

Theorem 8. *Any fit algorithm with size-*2 *bins is* 1*-competitive.*

Proof. Suppose AF uses n size-2 bins for a sequence of items. When AF first uses n bins due to the arrival of a new item X, all the existing $n-1$ bins must have a load greater than 1, otherwise, X can be packed into one of these bins and AF does not need to open a new bin. In other words, the total load of items is at least $n-1+s$ where s is the size of X. Any algorithm using unit-size bins needs at least n bins to pack all these items. Therefore, the maximum number of size-2 bins used by AF is at most that used by the optimal off-line algorithm. □

Theorem 9. *No on-line algorithm can be* 1*-competitive by using size-x bins, for any $x < 2$.*

Proof. Suppose $x = 2-\epsilon$, for some small $\epsilon > 0$. Let k be a positive integer such that $1/2^{k-1} \geq \epsilon > 1/2^k$. Notice the size satisfies the property $2-1/2^{k-1} \leq 2-\epsilon = x < 2-1/2^k$. The adversary works in two phases.

In the first phase, release 2^{3k-1} items of size $1/2^k$. The total load of the items is 2^{2k-1} and all items can be packed into 2^{2k-1} unit-size bins. If the on-line algorithm uses more than 2^{2k-1} bins, we are done. So we only need to consider the case in which the on-line algorithm uses at most 2^{2k-1} bins. We are going to prove that the on-line algorithm uses at least 2^k bins with load at least $1-1/2^k$. Let Y be the number of such bins. Then the total load accommodated by the on-line algorithm is at most $Y(2-1/2^{k-1})+(2^{2k-1}-Y)(1-1/2^{k-1}) = 2^{2k-1}(1-1/2^{k-1})+Y$.

This load cannot be smaller than the total load of items, i.e, $2^{2k-1}(1-1/2^{k-1})+Y \geq 2^{2k-1}$. In other words, $Y \geq 2^{2k-1}/2^{k-1} = 2^k$.

In the second phase, we retain a load of $1-1/2^k$ in 2^k bins and let all other items depart. Then release $2^{2k-1}-(2^k-1)$ items of size 1. Notice that none of these items can be packed into an existing bin because $1 + 1-1/2^k$ is greater than x, the size of the bin. Therefore, the total number of bins used by the on-line algorithm is $2^{2k-1}+1$.

On the other hand, the optimal off-line algorithm can pack the items released in the first phase in a way that those retained in the second phase are packed into 2^k-1 bins and the departing items into the other $2^{2k-1}-(2^k-1)$ bins. The latter bins can be reused for the size-1 items released in the second phase. Hence, the maximum number of unit-size bins used by the optimal off-line algorithm is 2^{2k-1}, which is strictly smaller than the maximum number of size-x bins used by the on-line algorithm, i.e., the on-line algorithm is not 1-competitive. □

5 Trade-off between bin size and competitive ratio

In this section, we discuss results in which the on-line algorithm uses bins of size $1 < b < 2$. We first give a general lower bound for any on-line algorithm in terms of b. Then we analyze the performance of the first-fit algorithm giving its upper bound.

5.1 General lower bound for $1 < b < 2$

In this section, we give a general lower bound of dynamic bin packing for $1 < b < 2$. We describe two adversaries, one gives better lower bound for $1 < b < 1.5$ and the other for $1.5 \leq b < 2$. The first adversary attempts to obtain the following lemma.

Lemma 10. *No on-line algorithm using size-b bins can be better than $\frac{2}{b}$-competitive.*

Proof. Consider any on-line algorithm $\mathcal{A}$. Let ϵ be a small constant and $k = \lfloor \frac{1}{\epsilon} \rfloor - 2$. The adversary runs in 3 stages.

1. Release ϵ-items of total size k. If $\mathcal{A}$ uses more than $\frac{2k}{b}$ bins, we are done. Suppose $\mathcal{A}$ uses at most $\frac{2k}{b}$ bins. We claim that there must be at least $(\frac{2}{b}-1)k$ bins with load $\geq b-1+\epsilon$, otherwise, the total possible load accommodated by $\mathcal{A}$ is less than $((\frac{2}{b}-1)k-1)b+(k+1)(b-1+\epsilon) = k-1+(k+1)\epsilon < k$, contradiction.
2. Depart items until $\{\, k(\frac{2}{b}-1){:}(b-1+\epsilon) \,\}$. Then release k items of size $\frac{b}{2}+\epsilon$. At most one such item can be packed into an existing bin or an empty bin. So, at least $k-(\frac{2}{b}-1)k = 2k(1-\frac{1}{b})$ new bins are opened.
3. Depart items until $\{\, k(\frac{2}{b}-1){:}(b-1+\epsilon), 2k(1-\frac{1}{b}){:}(\frac{b}{2}+\epsilon) \,\}$. Finally, release $(\frac{2}{b}-1)k$ items of size 1. None of the items can be packed into existing bins, thus, another $(\frac{2}{b}-1)k$ new bins are opened. Total number of bins used becomes $\frac{2k}{b}$.

Now, we show that there is a schedule that uses at most $k+O(1)$ unit-size bins at any time. To achieve this, at the end of Stage 3, we require that the ϵ-items and $(\frac{b}{2}+\epsilon)$-items to be packed into at most $k-(\frac{2}{b}-1)k+O(1)$ bins, i.e., $2k(1-\frac{1}{b})+O(1)$ bins. Since ϵ is a small constant, it is possible to pack ϵ-items with a $(\frac{b}{2}+\epsilon)$-item to make a load of 1. It can be verified that the total size of ϵ-items and $(\frac{b}{2}+\epsilon)$-items, i.e., $k(\frac{2}{b}-1)(b-1+\epsilon)+2k(1-\frac{1}{b})(\frac{b}{2}+\epsilon)$, equals $2k(1-\frac{1}{b})+O(1)$; implying that they can be packed into $2k(1-\frac{1}{b})+O(1)$ bins. □

The second adversary makes use of unit fraction items, i.e., in the form $1/w$, for some integer w. Let m be the largest integer such that $b - \frac{1}{m} < 1$ and let $k = m!(m-1)!$. To ease discussion, we define the functions $\alpha(i)$ and $\beta(i)$ for any positive integer $1 \leq i \leq m$ as follows.

$$\text{Let } \alpha(1) = k, \quad \beta(i) = \sum_{j=1}^{i} \alpha(j), \quad \alpha(i) = \beta(i-1)\left(\frac{m+1-i}{m+2-i} - \frac{m-i}{m+1-i}\right).$$

In other words, $\alpha(i) = \frac{\beta(i-1)}{(m+2-i)(m+1-i)}$. E.g., $\alpha(2) = \frac{k}{m(m-1)}$, $\alpha(3) = \frac{k+\frac{k}{m(m-1)}}{(m-1)(m-2)}$. The following property is used in the design of the adversary and the analysis.

Fact 11. *Both $\alpha(i)$ and $\beta(i)$ are integer multiples of $(m-i+1)!(m-i)!$*

Proof. By definition, $\alpha(1) = \beta(1) = m!(m-1)!$, the fact is true for $i = 1$. Suppose the fact is true for all $j \leq p$ for some p. By definition, $\alpha(p+1) = \frac{\beta(p)}{(m+1-p)(m-p)}$ which is a multiple of $(m-p)!(m-p-1)!$, by the induction hypothesis. Also notice that for $j \leq p$, $\alpha(j)$ being a multiple of $(m-j+1)!(m-j)!$ implies that $\alpha(j)$ is also a multiple of $(m-p)!(m-p-1)$. Since $\beta(p) = \sum_{1 \leq j \leq p} \alpha(j)$, $\beta(p)$ is also a multiple of $(m-p)!(m-p-1)!$, and the fact follows. $\square$

Consider any on-line algorithm $\mathcal{A}$. The adversary runs in m stages. In Stage 1, release $\frac{1}{m}$-items up to a total size of k, i.e., km such items. For each stage $2 \leq i \leq m$, we depart some items released in previous stage and then release some $\frac{1}{m+1-i}$-items, such that in Stage i, the following invariants are maintained: (1) items of a total size $\beta(i-1)\frac{m+1-i}{m+2-i}$ depart and the same size of $\frac{1}{m+1-i}$-items are released, keeping the total size of items being k; and (2) $\mathcal{A}$ uses at least $\alpha(i)$ more new bins at the end of the stage. Stage i proceeds as follows.

a. Retain one $\frac{1}{m+2-i}$-item from each of the $\alpha(i-1)$ new bins used in Stage $i-1$ and let all other $\frac{1}{m+2-i}$-items depart, i.e., only retain $\alpha(i-1)$ such items.
b. Release items of size $\frac{1}{m+1-i}$ until the total size of all items is k.

We are going to prove by induction that the invariants are maintained. Notice that the total size of items that we have considered is always an integer, which can be showed to be a consequence of Fact 11. We first show that the invariants hold for Stage 2. The total size of $\frac{1}{m}$-items departed equals $\beta(1) - \alpha(1)\frac{1}{m} = \beta(1)\frac{m-1}{m}$. A total size of $\beta(1)\frac{m-1}{m}$ items of size $\frac{1}{m-1}$ are then released. Notice that at most $m-2$ such items can be packed into an existing bins, otherwise, the load of a bin is at least $\frac{1}{m} + 1 > b$, by the definition of m. In other words, a total size of at least $\beta(1)(\frac{m-1}{m} - \frac{m-2}{m-1})$ items cannot be packed into existing bins, implying that $\mathcal{A}$ needs at least $\alpha(2)$ new bins. Therefore, the invariants hold for Stage 2. Suppose the invariants hold for all stages up to Stage p for some $p \geq 2$. Consider Stage $p+1$. The total size of items departed equals $\beta(p-1)\frac{m+1-p}{m+2-p} - \alpha(p-1)\frac{1}{m+1-p}$. By some arithmetic, we can show that this equals $\beta(p)\frac{m-p}{m+1-p}$. After the departure, every existing bin has a load of at least $\frac{1}{m}$. Using a similar argument as the base case, at most $m-p-1$ items of size $\frac{1}{m-p}$ released in this stage can be packed into existing bins, with a total size of $\beta(p)\frac{m-p}{m+1-p} - \beta(p)\frac{m-p-1}{m-p}$ remained, thus requiring at least $\alpha(p+1)$ new bins. Therefore, the invariants also hold for Stage $p+1$. At the end of Stage m, $\mathcal{A}$ uses a total of at least $\beta(m)$ bins.

Consider the optimal off-line algorithm. Note that the total size of items at any time is kept at k. Furthermore, the total size of items of the same type of item is always an integer because of Fact 11, and the item size is unit fraction, we can always pack the same type of item fully into the same unit-size bin. Therefore, the optimal off-line algorithm only needs k bins. We have the following theorem.

Lemma 12. *No on-line algorithm using size-b bins can be better than $\frac{\beta(m)}{m!(m-1)!}$-competitive, where m is the largest integer such that $b - \frac{1}{m} < 1$.*

Combining Lemmas 10 and 12, we have the following lower bound (see Figure 3 for the trend).

Theorem 13. *No on-line algorithm using size-b bins is better than* $\max\{\frac{\beta(m)}{m!(m-1)!}, \frac{2}{b}\}$*-competitive, where m is the largest integer such that $b - \frac{1}{m} < 1$.*

5.2 Performance of first-fit for $1 < b < 2$

In this section, we analyze the upper bound of the first-fit algorithm using size-b bins. To simplify the discussion, we refer to two properties pointed out by Coffman et al. [7]. (1) We can focus on the input sequences such that the maximum number of bins used by FF when the last item is packed and not before. (2) No occupied bin ever becomes empty during the execution of FF on input sequences satisfying the first property. It can be shown [7] that the two properties are true because FF will work out the same packing for the modified input sequence, e.g., a modified sequence in which the items packed to that occupied bin are removed. By the second property, we can label the occupied bins by the order they became occupied, i.e., bin i refers to the i-th bin used by FF, and the labels never change.

Let k denote the maximum number of bins used by the optimal off-line algorithm using unit-size bins. Suppose x is the last (maximum) bin that FF ever packs an item of size $\leq 1/2$. Let B be the last bin that FF opens with an item of size $\leq b/2$ and y be the number of bins (including B) with label $> x$ whose smallest item has a size in the range $(1/2, b/2]$ when FF opens B. Let $z = n - x - y$, where n is the maximum number of bins used by FF. We claim that the following inequalities hold.

$$\begin{aligned} (x-1)(b-1/2) &\leq k && (1)\\ xb/2 + (y-1) &\leq k && (2)\\ (x+y)(b-1) + zb/2 &\leq k && (3)\\ x(b-1) + y/2 + zb/2 &\leq k && (4)\\ y + z &\leq k && (5) \end{aligned}$$

When FF packs an item of size at most $1/2$ into the x-th bin, all existing bins must have a load at least $b - 1/2$, otherwise, FF would pack the item into those bins instead. The total item size is $\geq (x-1)(b-1/2)$, which must be $\leq k$, the total size that can be accommodated by the optimal off-line algorithm using unit-size bins; Inequality 1 follows. When FF opens the bin B with an item of size $\leq b/2$, the first x bins must have load at least $b/2$; and there must be at least $y - 1$ bins with label $> x$ containing two items of size at least $1/2$, otherwise, there will be a bin with load $\leq b/2$ and FF does not need to open B. Thus, Inequality 2 follows. For Inequality 3, when FF packs the first item into bin-n, all existing bins must have a load at least $b - 1$ (otherwise, FF can pack the item in those bins instead of opening a new bin) and there must be z bins containing an item of size $\geq b/2$ (the latter can be argued by using Coffman's two properties about input sequence but we skip details here). A similar argument gives Inequality 4. Finally, Inequality 5 is due to the fact that items packed to the last $y + z$ bins must have size greater than $1/2$, each of them must be packed into a different bin in the optimal off-line algorithm that uses unit-size bins.

Using Inequalities 1 and 5, $x+y+z \leq \frac{2k}{2b-1} + 1 + k = \frac{2b+1}{2b-1}k + 1$. Using Inequalities 1, 2 and 3, we can show that $x+y+z \leq \frac{5-b}{(2b-1)}k + O(1)$. Inequalities 1, 2 and 4 give $x+y+z \leq \frac{b^2+3}{b(2b-1)}k + O(1)$.

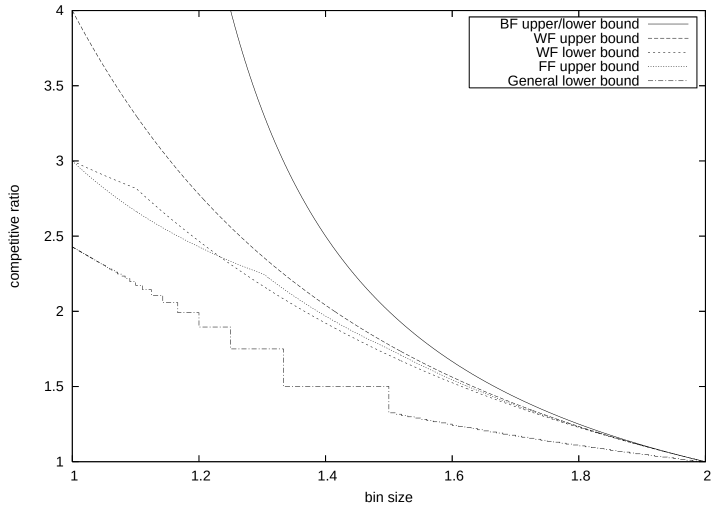

Figure 3: Trade-off between bin size b and competitive ratio.

Theorem 14. *The competitive ratio of FF using size-b bins is at most* $\min\{\frac{2b+1}{2b-1}, \frac{5-b}{(2b-1)}, \frac{b^2+3}{b(2b-1)}\}$.

Figure 3 shows how the competitive ratio of FF varies with b. Notice that our formula in Theorem 14 reaches the value 1 when $b = 2$ matching Theorem 8; yet when $b = 1$, the value is 3, not matching the existing best upper bound of 2.788 [7]. We leave it as an open question to close the gap between the upper and lower bounds. We now state the performance of BF and WF (see the upper three curves in Figure 3 and Table 1). The proof is given in the Appendix.

Theorem 15. *(1) BF using size-b bins is* $\frac{1}{b-1}$*-competitive, this bound is tight. (2) WF using size-b bins is* $\frac{4}{b^2}$*-competitive; on the other hand, its competitive ratio is no better than* $\min\{\frac{2+b}{b}, \frac{b^2-8b+20}{4b}\}$.

Algorithm	upper bound	lower bound
BF	$\frac{1}{b-1}$	$\frac{1}{b-1}$
WF	$\frac{4}{b^2}$	$\min\{\frac{2+b}{b}, \frac{b^2-8b+20}{4b}\}$
FF	$\min\{\frac{2b+1}{2b-1}, \frac{5-b}{(2b-1)}, \frac{b^2+3}{b(2b-1)}\}$	$\max\{\frac{\beta(m)}{m!(m-1)!}, \frac{2}{b}\}$

Table 1: Summary of results for bin size $1 \leq b < 2$.

6 Concluding remarks

In this paper, we have shown a 2.5 lower bound for dynamic bin packing, revealing that dynamic bin packing of general items is more difficult than unit fraction items. An open question is to close the gap between this 2.5 lower bound and the 2.788 upper bound [7]. We believe it is possible to push down the upper bound by analyzing some modified version of FF. One can also analyze other algorithms, like the class of Harmonic algorithms [17], yet our preliminary study showed that some versions of Harmonic algorithm have a non-constant lower bound; further investigation on other variants of Harmonic algorithms is desirable. We also give the first resource augmentation analysis for dynamic bin packing, showing that doubling bin size is necessary and sufficient to achieve 1-competitiveness. Trade-off of bin size and competitive ratio is also studied. Note that the formula derived for the upper bound of FF does not yet match the general lower bound. We are attempting to give tighter bounds to close the gap.

References

[1] 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.

[2] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[3] W. T. Chan, T. W. Lam, and P. W. Wong. Dynamic bin packing of unit fractions items. In *ICALP*, pages 614–626. Springer, 2005.

[4] E. G. Coffman, Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis. Bin packing with discrete item sizes, Part I: Perfect packing theorems and the average case behavior of optimal packings. *SIAM J. Discrete Math.*, 13:38–402, 2000.

[5] 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.

[6] E. G. Coffman, Jr., M. Garey, and D. Johnson. Bin packing with divisible item sizes. *Journal of Complexity*, 3:405–428, 1987.

[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] E. G. Coffman, Jr., D. S. Johnson, L. A. McGeoch, P. W. Shor, and R. R. Weber. Bin packing with discrete item sizes, Part III: Average case behavior of FFD and BFD. in preparation.

[10] E. G. Coffman, Jr., D. S. Johnson, P. W. Shor, and R. R. Weber. Bin packing with discrete item sizes, Part II: Tight bounds on first fit. *Random Structures and Algorithms*, 10:69–101, 1997.

[11] 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.

[12] J. Csirik and G. J. Woeginger. Resource augmentation for online bounded space bin packing. *J. Algorithms*, 44(2):308–320, 2002.

[13] L. Epstein and R. van Stee. Online bin packing with resource augmentation. In G. Persiano and R. Solis-Oba, editors, *WAOA*, volume 3351 of *Lecture Notes in Computer Science*, pages 23–35. Springer, 2004.

[14] Z. Ivkovic and E. L. Lloyd. Fully dynamic algorithms for bin packing: Being (mostly) myopic helps. *SIAM J. Comput.*, 28(2):574–611, 1998.

[15] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science*, pages 214–221, 1995.

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

[17] A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Inf. Process. Lett.*, 43(5):277–284, 1992.

# bins	$\mathbf{2}$	$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{2k}$	$\boldsymbol{k}$	$\boldsymbol{k+\frac{x}{2}}$	$\boldsymbol{\frac{x}{2}}$	$\boldsymbol{11k-x}$	$\boldsymbol{k}$
Stage 1	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 2	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}_{:\epsilon}$	$\frac{1}{3}, \frac{2}{3}_{:\epsilon}$	$\frac{2}{3}_{:\epsilon}, \frac{1}{3}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{3}, \frac{1}{3}$
Stage 3	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$(\frac{1}{2}-\epsilon)_{:\epsilon}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{3}_{:\epsilon}$, $\frac{1}{2}-\frac{\epsilon}{4}$
Stage 4	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{2}$	$\frac{1}{2}-\frac{\epsilon}{4}, \frac{1}{2}$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}$	$(\frac{1}{2}-\epsilon)_{:\epsilon}$, $\frac{1}{2}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{2}$
Stage 5	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{3}, (\frac{1}{6}-\epsilon)_{:\epsilon}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$(\frac{1}{2}-\epsilon)_{:\epsilon}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{3}_{:\epsilon}$, $\frac{1}{2}+\frac{\epsilon}{4}$
Stage 6	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{3}_{:\epsilon}, \frac{2}{3}$	$\frac{1}{3}_{:\epsilon}$
Stage 7	$1_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{2}, \frac{1}{2}$	1	1	1	1

Table 2: The optimal schedule for Case 2.2.

Appendix

Optimal schedule for the adversary in Section 3 (the 2.5 lower bound)

Now, we show how all the items can be packed into $18k+2$ bins (without repacking) in an off-line manner. We overload the notation a bit and use the notation $y_{:z}$ to describe packing a bin with a load y of z-items.

Case 1: In Stage 1, all ϵ-items can be packed into $18k$ bins, each with $1_{:\epsilon}$. In Stage 2, we pack 2 bins with $1_{:\epsilon}$ and $18k$ bins with $\frac{1}{2}+\frac{1}{2}$. This means that the $24k$ items that do not depart are packed together into 2 bins. Finally, in Stage 3, we pack 2 bins with $1_{:\epsilon}$, $3k$ bins with $\frac{1}{2}+\frac{1}{2}$, $15k$ bins with 1-item. This is possible because we can pack the $6k$ $\frac{1}{2}$-items that do not depart together into $3k$ bins. At any time, at most $18k+2$ bins are used.

Case 2.1: In Stage 1, the packing is the same as in Case 1, non-departing items are packed together. Then in Stage 2, pack 2 bins with $1_{:\epsilon}$, $(8k-\frac{x}{6})$ bins with $1_{:\epsilon}$, $(10k+\frac{x}{6})$ bins with $3\times\frac{1}{3}$. Finally in Stage 3, we pack 2 bins with $1_{:\epsilon}$, $3k$ bins with $3\times\frac{1}{3}$ and $15k$ bins with 1-item. At any time, at most $18k+2$ bins are used.

The remaining cases are presented in Tables 2, 3, and 4. Note that we always pack non-departing items together into the same bin.

# bins	$\mathbf{1}$	$\boldsymbol{m}$	$\boldsymbol{m}$	$\boldsymbol{m}$	$\boldsymbol{2m}$	$\boldsymbol{m+\frac{x}{2}}$	$\boldsymbol{2m-\frac{x}{2}}$	$\boldsymbol{4m}$
Stage 1	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 2	$1_{:\epsilon}$	$6\times\frac{1}{6}$	$6\times\frac{1}{6}$	$6\times\frac{1}{6}$	$6\times\frac{1}{6}$	$\frac{2}{3}_{:\epsilon}, \frac{1}{6}, \frac{1}{6}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 3	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{2}{3}_{:\epsilon}, \frac{1}{3}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 4	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{3}$	1	1	1	1

Table 3: The optimal schedule for Case 3.1.

# bins	$\boldsymbol{2}$	$\boldsymbol{m}$	$\boldsymbol{2m}$	$\boldsymbol{m}$	$\boldsymbol{m}$	$\boldsymbol{2m}$	$\boldsymbol{x}$	$\boldsymbol{m-\frac{x}{2}}$	$\boldsymbol{m-\frac{x}{2}}$	$\boldsymbol{m}$	$\boldsymbol{2m}$
Stage 1	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 2	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}_{:\epsilon}$	$4 \times \frac{1}{6}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$4 \times \frac{1}{6}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$4 \times \frac{1}{6}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$6 \times \frac{1}{6}$	$\frac{1}{6}, \frac{1}{2}_{:\epsilon}$ $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}, \frac{1}{2}_{:\epsilon}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{5}{6}_{:\epsilon}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 3	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}$, $(\frac{1}{3}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$\frac{1}{3}, \frac{1}{2}_{:\epsilon}$ $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}_{:\epsilon}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{5}{6}_{:\epsilon}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 4	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}-\frac{\epsilon}{4}$	$\frac{1}{3}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{2}-\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 5	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{3}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$\frac{1}{3}_{:\epsilon}, \frac{1}{2}+\frac{\epsilon}{4}$, $(\frac{1}{6}-\epsilon)_{:\epsilon}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 6	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{1}{3}_{:\epsilon}, \frac{2}{3}$	$\frac{1}{3}_{:\epsilon}, \frac{2}{3}$	$1_{:\epsilon}$	$1_{:\epsilon}$
Stage 7	$1_{:\epsilon}$	$\frac{1}{6}, \frac{1}{6}$ $\frac{2}{3}$	$\frac{1}{3}, \frac{2}{3}$	$\frac{2}{3}$	1	$\frac{1}{2}-\frac{\epsilon}{4}$, $\frac{1}{2}+\frac{\epsilon}{4}$	1	1	1	1	1

Table 4: The optimal schedule for Case 3.2.

Proof of Theorem 15

We denote by OPT the optimal off-line algorithm using unit-size bins.

Upper bound of BF

Suppose BF uses a maximum of n bins. When BF first packs an item into bin n, the load of bin i for $i < n$ is at least $b-1$, otherwise, BF can pack the item into those bins instead of opening a new bin. Therefore, OPT needs $n^* \geq (n-1)(b-1)$ bins, and hence the competitive ratio of BF is at most $(\frac{n^*}{(b-1)})$ since $n \leq \frac{n^*}{(b-1)} + 1$.

Lower bound of BF

Let k be an arbitrarily large integer. We define a sequence of small positive constants ϵ_i, for $1 \leq i \leq k$, such that (i) $\epsilon_1 < \min\{(2-b)/2, b-1\}$, and (ii) $\epsilon_{i+1} = \epsilon_i/2$. To simplify the proof, we assume that we can pick an ϵ_k satisfying the above properties such that $b-1$ is divisible by ϵ_k (the other case can be proved similarly). The adversary runs in k phases. In phase 1, $(b-1+\epsilon_1)/\epsilon_k$ items of size ϵ_k are released. In each subsequent phase, phase i for $2 \leq i \leq k$, an item of size $1-\epsilon_{i-1}+\epsilon_i$ is first released, followed by $(b-1+\epsilon_i)/\epsilon_k$ items of size ϵ_k, and then the $1-\epsilon_{i-1}+\epsilon_i$ item departs. We will show that BF uses $\geq k$ bins while OPT uses $\leq k(b-1)+2$ bins, implying that the competitive ratio of BF is no better than $\frac{1}{b-1}$.

Let us look at how BF packs the above items. We claim that in each subsequent phase, phase i for $2 \leq i \leq k$, BF packs the newly arrived items into a new bin making its load become $b-1+\epsilon_i$ at the end of the phase. The claim holds for phase 1 because BF packs all those items into a single bin whose load becomes $b-1+\epsilon_1$. Assume the claim holds up to phase i.

In phase $i+1$, if the new item of size $1-\epsilon_i+\epsilon_{i+1}$ is packed into an existing bin $1 \leq j \leq i$, the load of this bin becomes $b-1+\epsilon_j+1-\epsilon_i+\epsilon_{i+1} = b+\epsilon_j-\epsilon_{i+1} > b$ because $\epsilon_{i+1} = \epsilon_i/2 < \epsilon_j$. Therefore, BF must pack the new item into a new bin. Note that the load of any existing i bins is at most $b-1+\epsilon_1$ and the load of the new bin is $1-\epsilon_i+\epsilon_{i+1}$. By the definition of ϵ_1, we have $b-1+\epsilon_1 < b-1+(2-b)/2 = b/2$; on the other hand, $1-\epsilon_i+\epsilon_{i+1} = 1-\epsilon_{i+1}/2 \geq 1-\epsilon_1/2 > 1-(2-b)/2 = b/2$. In other words, the load of the new bin is the highest among all bins. Therefore, BF packs the further arriving items of size ϵ_{i+1} into the new bin instead of the existing bins. Note that all these items can be packed into the new bin because $1-\epsilon_i+\epsilon_{i+1}+\epsilon_k(b-1+\epsilon_{i+1})/\epsilon_k = b-\epsilon_i+2\epsilon_{i+1} = b$. The departure of the $1-\epsilon_i+\epsilon_{i+1}$

item leaves the new bin to have a load of $b-1+\epsilon_{i+1}$ and the claim holds for phase $i+1$. Therefore, BF uses k bins.

On the other hand, OPT can use a dedicate bin to store the $1-\epsilon_i+\epsilon_{i+1}$ items (at any time, there is at most one such item), and $k(b-1)+1$ bins to store the ϵ_k items because the total load of such items is at most $k(b-1)+\sum \epsilon_i \leq k(b-1)+2\epsilon_1$. Altogether OPT only uses $k(b-1)+2$ bins. Therefore, BF using size-b bins, $b>1$, is no better than $\frac{1}{b-1}$-competitive.

Upper bound of WF

Suppose WF uses a maximum of n bins. Let t_1 be the time instance when WF packs the first item into bin n, and n_1 be the largest integer such that there is no item of size $> \frac{b}{2}$ in bin n_1 at t_1. Let $n_2 = n - n_1$. Consider the time instance t_2 when WF packs the first item into bin n_1. There are two cases: the size of the item is less than or equal to $\frac{b}{2}$, and the size is greater than $\frac{b}{2}$. Suppose we can show that at t_1, OPT uses m_1 bins and for the two cases regarding time t_2, OPT uses m_2 and m_3 bins, respectively. Then we can conclude that OPT uses $\geq \max\{m_1, \min\{m_2, m_3\}\}$ bins.

At t_1, WF has packed an item of size greater than $\frac{b}{2}$, into each of the bins n_1+1 to n. Furthermore, at t_1, WF packs an item into bin n, thus, each of the first $n-1$ bins must have at least a load of $b-1$. Therefore, the total load of items at t_1 is at least $n_2 b/2 + n_1(b-1)$. For the two cases regarding t_2, if the first item WF packs into bin n_1 is of size at most $b/2$, the total load of items at t_2 is at least $(n_1-1)(b-b/2) = (n_1-1)b/2$. In the latter case, bin n_1 has an item of size greater than $b/2$ at t_2 but no item of such size at t_1. Therefore, there must be a time instance that WF packs an item to bin n_1 while its load is at least $b/2$. By the property of WF, the load of all the other bins is also at least $b/2$ at that time instance, i.e., the total load is at least $(n_1-1)b/2$; otherwise, WF should have packed the item into those bins instead. In both cases, the total load at t_2 is at least $(n_1-1)b/2$. Therefore, OPT uses $\geq \max\{n_2 b/2 + n_1(b-1), (n_1-1)b/2\}$ bins. The maximum is attained when the two expressions are equal, by arithmetic, we can show that $n \leq \frac{4}{b^2}n^* + 1$. Hence, WF using size-$b$ bins is $\frac{4}{b^2}$-competitive.

Lower bound of WF

Let k be an arbitrarily large integer constant. To simply the proof, we assume that b is an integer multiple of $1/k$ (the other case can be proved similarly). The adversary attempts to first force WF to use $2k$ bins each with a load just more than $b/2$. Then some items depart from the first $2k$ bins and items with increasing size are released, this repeats until the load of each of the first $2k$ bins is just more than $b-1$. Finally, more items of size 1 are released, which have to be packed into some new bins.

Let $x=(b-1)k+1$, and $y=(b/2)k+1$. Intuitively, x and y is the minimum number of items of size $1/k$ required to make the load of a bin greater than $b-1$ and $b/2$, respectively. The adversary works in 3 phases. In phase 1, the following steps are repeated for $2k$ times: (i) an item of size $b/2$ is released, (ii) followed by an item with size $1/k$, (iii) the item with size $b/2$ departs, and (iv) finally $y-1$ items of size $1/k$ is released. Phase 2 is divided into $y-x$ stages. In stage j, for $1 \leq j \leq y-x$, an item of size $1/k$ from each of the bins used in phase 1 departs, then 2 items of size $(y+j)/k$ are released. In phase 3, $\min\{2(b-1)k, (b^2-4b+4)k/4\}$ of items with size 1 is released (note that we can pick k in such a way that it is divisible by 4). We are going to show that WF uses $\geq \min\{(2+b)k-3, (b^2-8b+20)k/4-3\}$ bins while OPT uses $\leq kb+1$ bins.

Consider how WF packs the above item into bins of size b. In phase 1, we observe that after every round of the 4 steps, WF uses one more bin and each of the existing bins contains y items of size $1/k$, i.e. has a load of more than $b/2$. The base case is easy. Assume the observation holds after some round. Then, the newly released item of size $1/2$ cannot be packed into the first bin, so both items of $b/2$ and $1/k$ are packed into a new bin. After the departure of the $b/2$ item, the new bin used has the lightest load among the others, and so all the remaining $y-1$ items are packed into this bin as well, making its load $y/k > b/2$.

In phase 2, we claim that at the end of stage j, for any $1 < j < y-x$, WF uses $2k+2j$ bin and each of them has a load $\geq (y-j)/k$. In stage 1, the newly arrived size $(y+1)/k$ items have to be packed into a new bin because each of the first $2k$ bins have a load of $(y-1)/k$ after items departed and $(y+1)/k+(y-1)/k > b$. The 2 newly arrived size $(y+1)/k$ items are also packed into 2 different bins because their size $> b/2$. Two new bins are used and thus the claim holds for stage 1. Assume that the claim holds for stage j. In stage $j+1$, all of the existing bins have a load of $\geq (y-j-1)/k$ after items departed. The newly arrived size $(y+j+1)/k$ items have to be packed into a new bin because $(y-j-1)/k+(y+j+1)/k > b$. The two newly arrived size $(y+j+1)/k$ items also have to be packed into 2 different bins because their size $> b/2$. WF uses $2k+j+2$ bin and each of them has a load $\geq (y-j-1)/k$, and the claim holds. At the end of phase 2, WF uses $2k+2(y-x)$ bins and each of them has a load $\geq x/k$. Note that by the definition of x and y, $y-x \geq (2-b)k/2$.

In phase 3, all the $\min\{2(b-1)k, (b^2-4b+4)k/4\}$ items of size 1 have to be packed into different new bins because $x/k+1 > b$ and $b < 2$. Combining the 3 phases, WF use a total of at least $\min\{(2+b)k-3, (b^2-8b+20)k/4-3\}$. We leave the proof of (2) in the appendix.

On the other hand, during phase 1, OPT can use 1 bin to hold the item with size $b/2$, and pack the other items with 1/k into kb bins. The $1/k$ items are arranged such that the departing items in phase 2 comes from 2 groups, the group $G1$ consists of $2((2-b)k/2)$ bins, and the group $G2$ consists of $\min\{2(b-1)k, (b^2-4b+4)k/4\}$ bins. Note that in each stage of phase 2, total size of items depart is 2, and total size of items release ≤ 2. The items are arranged such that in stage j of phase 2, $(y+j)$ items depart from 2 of the bins in $G1$, and remaining items depart from bins in $G2$. The space in the 2 bins of $G1$ can be reused to pack the 2 newly released size $(y+j)/k$ items, and the space in $G2$ can be later reused to pack the size 1 items in phase 3. Hence, the number of bins used by OPT is at most $kb+1$. As a result, WF using size-b bins is no better than $\min\{(2+b)/b, (b^2-8b+20)/4b\}$-competitive.