Dynamic Bin Packing of Unit Fractions Items

Wun-Tat Chan[1,⋆], Tak-Wah Lam[1], and Prudence W.H. Wong[2,⋆⋆]

[1] Department of Computer Science,
University of Hong Kong, Hong Kong
{wtchan, twlam}@cs.hku.hk
[2] Department of Computer Science,
University of Liverpool, UK
pwong@csc.liv.ac.uk

Abstract. This paper studies the dynamic bin packing problem, in which items arrive and depart at arbitrary time. We want to pack a sequence of unit fractions items (i.e., items with sizes $1/w$ for some integer $w \geq 1$) into unit-size bins such that the maximum number of bins used over all time is minimized. Tight and almost-tight performance bounds are found for the family of any-fit algorithms, including first-fit, best-fit, and worst-fit. We show that the competitive ratio of best-fit and worst-fit is 3, which is tight, and the competitive ratio of first-fit lies between 2.45 and 2.4985. We also show that no on-line algorithm is better than 2.428-competitive. This result improves the lower bound of dynamic bin packing problem even for general items.

1 Introduction

Bin packing problem has been studied since the early 70's and different variants of the problem continue to attract researchers attentions (see the survey [6, 9, 10]). In the classical bin packing problem, we want to pack a sequence of items each with size in the range $(0, 1]$ into unit-size bins using the minimum number of bins. One of the generalizations of the problem is known as the *dynamic bin packing problem* [8], in which items arrive and depart at arbitrary time. The objective is to minimize the maximum number of bins used over all time. In this paper, we study dynamic bin packing of *unit fractions items*. A unit fraction item has size of the form $1/w$ for some integer $w \geq 1$. We analyze the performance of the family of any-fit algorithms, which includes first-fit, best-fit and worst-fit, and provide tight and almost-tight performance bounds. Our lower bound on dynamic bin packing of unit fractions items even improves the lower bound of Coffman et al. [8] on dynamic bin packing of general items.

There is a long history of results for the classical bin packing problem and its variants [6, 9, 10]. Most of the previous works considered the "static" bin packing

⋆ This research was supported in part by Hong Kong RGC Grant HKU-5172/03E.
⋆⋆ This research was supported in part by Nuffield Foundation Grant NAL/01004/G.

L. Caires et al. (Eds.): ICALP 2005, LNCS 3580, pp. 614–626, 2005.

where items will not depart. In this model, the off-line bin packing problem is NP-hard [11]. In the on-line setting, each item must be assigned to a bin, without knowledge of the subsequent items. Moreover, no migration of items are allowed, i.e., items are not allowed to move from one bin to another. The performance is measured in terms of competitive ratio (see [3] for a survey). The current best upper bound is due to Seiden [14], who proved that the algorithm HARMONIC++ has a competitive ratio at most 1.58889[1]. The current best lower bound is due to van Vliet [15] who showed that no on-line algorithm can achieve a competitive ratio less than 1.54014.

In many real applications, item sizes are often not arbitrary real numbers in $(0, 1]$. Bar-Noy et al. [2] initiated the study of the *unit fractions bin packing problem* (UFBP), a restricted version of the classical bin packing problem in which all sizes are of the form $1/w$ for some integer $w \geq 2$. In the on-line setting, they gave an on-line algorithm with a competitive ratio $1+O(1/\sqrt{H})$, where H denotes the sum of sizes of all items. Note that this algorithm is asymptotically optimal. Bin packing with other restricted form of item sizes includes divisible item sizes [7] (where each possible item size can be divided by the next smaller item size) and discrete item sizes [5] (where possible item sizes are $\{1/k, 2/k, \cdots, j/k\}$ for some $1 \leq j \leq k$).

Dynamic bin packing is a generalization of the classical bin packing problem introduced by Coffman et al. [8]. The problem assumes that items may depart at arbitrary time, and the objective is to minimize the maximum number of bins used over all time. It was shown in their paper that the on-line algorithm first-fit has a competitive ratio lies between 2.75 and 2.897, and no on-line algorithm can achieve a competitive ratio better than 2.5. Note that these results assume a very general optimal off-line algorithm, which can re-pack the items. Coffman et al. also gave an improved lower bound of 2.388 when the off-line algorithm is not allowed to re-pack the items. Ivkovic and Lloyd [12] studied an even more general problem called the *fully dynamic bin packing problem*, where migration of items are allowed, and gave a 1.25-competitive on-line algorithm for this problem.

This paper studies the problem of dynamic bin packing of unit fractions items, the main contribution are several very close upper and lower bounds (see Table 1). We show that any-fit algorithms, which include first-fit, best-fit and worst-fit are 3-competitive. We further show that the performance of best-fit and worst-fit are indeed tight, i.e., they cannot be better than 3-competitive. On the other hand, we show that first-fit has a better performance, its competitive ratio lies between 2.45 and 2.4985. In addition, we prove that no on-line algorithm can be better than 2.428-competitive. This result improves the lower bound of 2.388 of Coffman et al. [8] on dynamic bin packing for general items.

There is a problem related to UFBP, called the *windows scheduling problem* (WS) [1, 2, 4], as pointed out by Bar-Noy et al. [2]. Similar to UFBP, the input of WS is a sequence of items, each with a *window* represented by an inte-

[1] Seiden [14] pointed out that the previous best algorithm HARMONIC+1 by Richey [13] has competitive ratio at least 1.59217 rather than the claimed 1.58872.

Table 1. Summary of results

Algorithms	Upper bounds	Lower bounds
First-fit	2.4985	2.45
Best-fit	3	3
Worst-fit	3	3
Any-fit	3	2.428
Any on-line algorithms	–	2.428

ger. Each item represents a piece of information to be broadcast to all clients. It is assumed that all items are of the same length, which takes the same amount of time to broadcast. The objective of WS is to use the minimum number of broadcast channels to broadcast each item periodically such that the duration between two consecutive broadcasts of the same item must not exceed the window of that item. By letting the bins as broadcast channels and the reciprocal of item sizes as windows, UFBP can be considered as a special case of WS, and hence the lower bound result on UFBP applies to WS. (Note that the upper bound on UFBP does not carry over to WS.) Chan and Wong [4] considered the dynamic version of WS, in which items may also depart. They gave a 5-competitive algorithm and showed that no on-line algorithm can be better than 2-competitive. The lower bound of dynamic bin packing of unit fractions item in this paper improves the lower bound for the dynamic version of WS to 2.428.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 analyzes the performance of the family of any-fit algorithms. This includes upper and lower bounds for first-fit (Sections 3.1 and 3.2, respectively), and upper and lower bounds for best-fit and worst-fit (Section 3.3). Finally, Section 4 gives a lower bound for any on-line algorithm.

2 Preliminaries

In this section, we give a precise definition of the dynamic unit fractions bin packing problem and the necessary notations for further discussion. We are to pack a sequence of items into bins of unit-capacity. Items arrive and depart at arbitrary time. We denote the i-th item by m_i and its arrival time by a_i. Each item m_i comes with a size s_i which is a reciprocal of an integer, i.e., $s_i = 1/w_i$ for some integer $w_i \geq 1$. When item m_i arrives at a_i, it must be assigned to a bin immediately. At any time, the *load* of a bin is the total size of items that are currently assigned to that bin and have not yet departed, and this load must be at most 1 because of unit-capacity. 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.

As with previous work, we measure the performance of an on-line algorithm in terms of a competitive ratio. Given a sequence σ of items and an on-line bin packing algorithm $\mathcal{A}$, let $\mathcal{A}(\sigma, t)$ denote the number of bins used by $\mathcal{A}$ at time t.

We say that $\mathcal{A}$ is c-competitive if there exists a constant k such that for any input sequence σ, we have $\max_t \mathcal{A}(\sigma, t) \leq c \cdot \max_t \mathcal{O}(\sigma, t) + k$, where $\mathcal{O}$ is the optimal off-line algorithm.

In this paper, we 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 an occupied bin as long as there exists such a bin that can accommodate the item; a new bin is only used when no occupied bins can accommodate the item. The algorithms differ in the rule used to choose the occupied bin for the newly arrived item. To describe the rules of these algorithms, we first define a way to label the occupied bins at a specific time.

> At any time t, suppose that there are n occupied bins. For any bin X among these n bins, let $f(x) \leq t$ be the latest time X turns from empty to non-empty. At time t, we label these n non-empty bins using integers $1, 2, \ldots, n$ such that the label of bin X is less than that of bin Y if $f(X) \leq f(Y)$. Notice that the labels of bins change over time.

When a new item m_i arrives, if there is any occupied bin with load no more than $1-1/w_i$, the algorithms assign m_i to one of these bins as follows:

Any-fit (AF) assigns m_i to any of these bins arbitrarily.
First-fit (FF) assigns m_i to the one with the smallest label at a_i.
Best-fit (BF) assigns m_i to the heaviest load one; ties are broken arbitrarily.
Worst-fit (WF) assigns m_i to the lightest load one; ties are broken arbitrarily.

3 Performance of the Family of Any-Fit Algorithms

In this section we analyze the performance of any-fit algorithms. In Sections 3.1 and 3.2, we give an upper bound of 2.4985 and a lower bound of 2.45 for the competitive ratio of FF. Then in Section 3.3, we show that both BF and WF cannot be better than 3-competitive and then give the matching upper bounds.

3.1 Upper Bound for First-Fit

Before we analyze the upper bound of FF, let us have some definitions. Consider any positive integers x and y. Suppose that we pack a bin using items of sizes $1, 1/2, \ldots, 1/x$ only. We want to define the notion of the minimum load that such a bin must have in order that an additional item of size $1/y$ cannot be packed into the bin. We define a function $\alpha(x, y)$ to capture this notion. Formally,

$$\alpha(x, y) = \min_{1 \leq j \leq x \,\&\, n_j \geq 0} \{n_1 + n_2/2 + \ldots n_x/x \mid n_1 + n_2/2 + \ldots n_x/x > 1 - 1/y\}.$$

For example, when $x = 4$ and $y = 3$, we have $\alpha(4, 3) = 3/4$ and correspondingly $n_1 = 0$, $n_2 = 1$, $n_3 = 0$ and $n_4 = 1$.

With respect to a particular input σ, we define a sequence of integer pairs (b_i, r_i) as follows. Let b_1 denote the maximum number of bins used by FF over all time.

Suppose the smallest item that FF ever packs into a bin with label b_1 is of size $1/r_1$. We define b_i and r_i for $i \geq 2$ as follows. Let $b_i < b_{i-1}$ be the largest integer such that FF ever packs an item of size smaller than $1/r_{i-1}$ into a bin with label b_i. The size of the smallest item that FF ever packs into a bin with label b_i is denoted $1/r_i$. Let k be the largest value of i that b_i and r_i can be defined. Notice that $b_1 > b_2 > \ldots > b_k$ and $r_1 < r_2 < \ldots < r_k$.

Now we are ready for the analysis. Consider the time instance t_k when FF packs an item X of size $1/r_k$ into a bin B with label b_k. Since k is the largest index of b_i that can be defined, no item with size smaller than $1/r_k$ has ever been packed into any bin, in particular the bins with label from 1 to $b_k - 1$. Together with the fact that FF packs X into B but not bins with labels 1 to $b_k - 1$, we can conclude that at time t_k, each of the bins with labels from 1 to $b_k - 1$ must have a load at least $\alpha(r_k, r_k)$. Including item X, the total load of all bins at t_k is at least $1/r_k + (b_k - 1) \cdot \alpha(r_k, r_k)$. Let

$$\ell_k = 1/r_k + (b_k - 1) \cdot \alpha(r_k, r_k).$$

Next, for any integer $1 \leq i \leq k-1$, consider the time instance t_i when FF packs an item X_i of size $1/r_i$ into a bin with label b_i. By the definition of b_i and r_i, we can use a similar argument as before to show that: (1) each of the bins with labels from 1 to b_k must have load at least $\alpha(r_k, r_i)$; (2) for any integer p with $k > p \geq i+1$, each of the bins with labels from $b_{p+1}+1$ to b_p must have load at least $\alpha(r_p, r_i)$; and (3) each of the bins with labels from $b_{i+1}+1$ to $b_i - 1$ must have load at least $\alpha(r_i, r_i)$. Including item X_i, the total load of all bins at time t_i is at least $1/r_i + (b_i - b_{i+1} - 1) \cdot \alpha(r_i, r_i) + \sum_{p=i+1}^{k-1} (b_p - b_{p+1}) \cdot \alpha(r_p, r_i) + b_k \cdot \alpha(r_k, r_i)$. For $1 \leq i \leq k-1$, let

$$\ell_i = 1/r_i + (b_i - b_{i+1} - 1) \cdot \alpha(r_i, r_i) + \sum_{p=i+1}^{k-1} (b_p - b_{p+1}) \cdot \alpha(r_p, r_i) + b_k \cdot \alpha(r_k, r_i).$$

Let $\ell = \max_{1 \leq i \leq k} \ell_i$. The number of bins used by the optimal off-line algorithm is at least ℓ. On the other hand, the maximum number of bins used by FF is b_1. Below we show that $b_1 < 2.4985\ell + 1.337$, which implies the following theorem.

Theorem 1. *First-fit is* 2.4985*-competitive.*

To prove Theorem 1, we assume $k \geq 5$. The case for $k < 5$ can be proved similarly and will be given in the full paper. Depending on the values of r_i's, we consider the following six sub-cases: **Case 1**: $r_1 \geq 2$; **Case *i***, for $2 \leq i \leq 5$: $r_1 = 1$, $r_2 = 2$, $\cdots$, $r_{i-1} = i - 1$, and $r_i \geq i + 1$; and **Case 6**: $r_1 = 1$, $r_2 = 2$, $\cdots$, $r_5 = 5$. We analyze the relationship between b_1 and ℓ case by case in each of the following lemmas.

Lemma 1. *If* $r_1 \geq 2$*, then* $b_1 < 2\ell + 1$*.*

Proof. Since $\alpha(x, y) > 1 - 1/y$ for any integers x and y, we have $\ell_1 > 1/r_1 + (b_1 - b_2 - 1)(1 - 1/r_1) + \sum_{p=2}^{k-1} (b_j - b_{j+1})(1 - 1/r_1) + b_k(1 - 1/r_1) = 1/r_1 + (b_1 - 1)(1 - 1/r_1)$. By simple arithmetic, we have $b_1 < \ell_1 r_1/(r_1 - 1) + (r_1 - 2)/(r_1 - 1) < 2\ell + 1$; the latter inequality holds because $r_1 \geq 2$ and $\ell_1 \leq \ell$.

Lemma 2. *If $r_1 = 1$ and $r_2 \geq 3$, then $b_1 < 2.4445\ell + 1$.*

Proof. Notice that for any integer x, $\alpha(x, 1)$ is the minimum value in the form of $n_1 + n_2/2 + n_3/3 + \ldots + n_x/x$ that is greater than 0; therefore, $\alpha(x, 1)$ must be equal to $1/x$. Then, we have $\ell_1 = 1/1 + (b_1 - b_2 - 1) \cdot \alpha(1, 1) + \sum_{p=2}^{k-1}(b_p - b_{p+1}) \cdot \alpha(r_p, 1) + b_k \cdot \alpha(r_k, 1) > (b_1 - b_2) + (b_2 - b_3)/r_2$.

Next, by the definition that $\alpha(x, y) \geq 1 - 1/y$ for any integers x and y, we have $\ell_2 = 1/r_2 + (b_2 - b_3 - 1) \cdot \alpha(r_2, r_2) + \sum_{p=3}^{k-1}(b_p - b_{p+1}) \cdot \alpha(r_p, r_2) + b_k \cdot \alpha(r_k, r_2) > (b_2 - 1)(1 - 1/r_2)$. Similarly, we have $\ell_3 > (b_3 - 1)(1 - 1/r_3)$. By solving the three inequalities, we have $b_1 < 22\ell/9 + 1 < 2.4445\ell + 1$.

Lemma 3. *If $r_1 = 1$, $r_2 = 2$, and $r_3 \geq 4$, then $b_1 < 2.4792\ell + 1.25$.*

Proof. Recall that $\alpha(x, 1) > 1/x$ for any integer x. We have $\ell_1 > (b_1 - b_2) + (b_2 - b_3)/2 + (b_3 - b_4)/r_3$. By the fact that $\alpha(2, 2) = 1$ and $\alpha(x, 2) > 1/2$ for any integer x, we have $\ell_2 = 1/2 + (b_2 - b_3 - 1) \cdot \alpha(2, 2) + \sum_{p=3}^{k-1}(b_p - b_{p+1}) \cdot \alpha(r_p, 2) + b_k \cdot \alpha(r_k, 2) > (b_2 - b_3 - 1) + b_3/2$. We can also prove that $\ell_3 > (b_3 - 1)(1 - 1/r_3)$ and $\ell_4 > (b_4 - 1)(1 - 1/r_4)$. By solving the four inequalities, we have $b_1 < 119\ell/48 + 5/4 < 2.4792\ell + 1.25$, and the lemma follows.

Lemma 4. *If $r_1 = 1$, $r_2 = 2$, $r_3 = 3$, and $r_4 \geq 5$, then $b_1 < 2.4942\ell + 1.3167$.*

Proof. Using the same approach, we have

$$\begin{aligned}
&\ell_1 > (b_1 - b_2) + (b_2 - b_3)/2 + (b_3 - b_4)/3 + (b_4 - b_5)/r_4, \\
&\ell_2 > (b_2 - b_3 - 1) + (b_3 - b_4)(2/3) + b_4/2, && \{\because \alpha(3, 2) = 2/3\} \\
&\ell_3 > ((b_3 - b_4 - 1)(5/6) + 2b_4/3, && \{\because \alpha(3, 3) = 5/6\} \\
&\ell_4 > (b_4 - 1)(1 - 1/r_4), \text{ and} \\
&\ell_5 > (b_5 - 1)(1 - 1/r_5).
\end{aligned}$$

By solving the five inequalities, we have $b_1 < 2993\ell/1200 + 79/60 < 2.4942\ell + 1.3167$, and the lemma follows.

Lemma 5. *If $r_i = i$ for all $1 \leq i \leq 4$, and $r_5 \geq 6$, then $b_1 < 2.49345\ell + 1.3325$.*

Proof. Using the same approach, we have

$$\begin{aligned}
&\ell_1 > (b_1 - b_2) + (b_2 - b_3)/2 + (b_3 - b_4)/3 + (b_4 - b_5)/4 + (b_5 - b_6)/r_5, \\
&\ell_2 > (b_2 - b_3 - 1) + (b_3 - b_4)(2/3) + (b_4 - b_5)(7/12) + b_5/2, && \{\because \alpha(4, 2) = 7/12\} \\
&\ell_3 > (b_3 - b_4 - 1)(5/6) + (b_4 - b_5)(3/4) + 2b_5/3, && \{\because \alpha(4, 3) = 3/4\} \\
&\ell_4 > ((b_4 - b_5 - 1)(5/6) + b_5(3/4), && \{\because \alpha(4, 4) = 5/6\} \\
&\ell_5 > (b_5 - 1)(1 - 1/r_5), \text{ and} \\
&\ell_6 > (b_6 - 1)(1 - 1/r_6).
\end{aligned}$$

Solving these inequalities, we have $b_1 < 2.4935\ell + 1.3325$, and the lemma follows.

Lemma 6. *If $r_i = i$ for all $1 \leq i \leq 5$, then $b_1 < 2.4985\ell + 1.337$.*

The proof of Lemma 6 uses a similar approach as in Lemmas 1 to 5, and the details will be given in the full paper. By Lemmas 1 to 6, Theorem 1 follows.

In fact, we conjecture that the worst case happens when $r_i = i$ for all $1 \leq i \leq k$. In that case, the computed competitive ratio is approaching 2.48.

3.2 Lower Bound for First-Fit

In this section we give a lower bound for FF by constructing an adversary sequence of items such that the maximum number of bins used by FF is at least 2.45 times that used by the optimal off-line algorithm. For any positive integers x and y, define $\beta(x, y)$ to be the minimum number of items of size $1/x$ in a bin such that an additional item of size $1/y$ cannot be packed into the bin. Formally,

$$\beta(x, y) = \min_{z \in \mathcal{Z}^+} \{z \mid z/x > 1 - 1/y\},$$

i.e., $\beta(x, y) = 1 + x - \lceil x/y \rceil$. For example, if $x = 4$ and $y = 3$, then $\beta(x, y) = 3$.

Let n be an integer and let $D = n!$. The adversary sequence consists of n stages. In each stage, some items released in the previous stage depart and a number of new items of the same size are released. The choices of which items to depart depend on how FF packs the items in previous stages. In Stage 1, Dn items of size $1/n$ are released. FF packs all Dn items into D bins, and each bin is fully packed.

For subsequent stages, i.e., Stage i, for $2 \leq i \leq n$, the adversary targets to maintain an invariant on how FF packs the items: At the beginning of Stage i,

- each occupied bin contains only items of the same size; and
- a bin that contains items of size $1/x$ contains $\beta(x, n - i + 2)$ items.

The invariant holds at the beginning of Stage 2 because each occupied bin contains $\beta(n, n) = n$ items of size $1/n$. Stage i consists of two steps.

1. For each occupied bin, if it contains items of size $1/x$, we arbitrarily choose $\beta(x, n - i + 2) - \beta(x, n - i + 1)$ items and let them depart, in other words, there are $\beta(x, n-i+1)$ items remained. Let D_i be the sum of item size for all the departed items. We will prove later that D_i is an integer (see Lemma 7).
2. Next, $D_i(n-i+1)$ items of size $1/(n-i+1)$ are released. Since each bin with item of size $1/x$ contains $\beta(x, n - i + 1)$ items, none of the newly released items can be packed into any occupied bin. Therefore, FF will use D_i empty bins to pack all these items, each bin contains $n-i+1 = \beta(n-i+1, n-i+1)$ items. Thus, the invariant also holds at the beginning of Stage $i + 1$.

Define $D_1 = D$, which is the number of empty bins used in Stage 1. From the above discussion, we can see that the number of empty bins required for items of size $1/(n-i+1)$ in Stage i is D_i. Then, at the beginning of Stage i, there will be D_j bins each with items of size $1/(n - j + 1)$ for all $1 \leq j \leq i - 1$. Therefore, the sum of the size of all departed items in Stage i satisfies:

$$D_i = \sum_{j=1}^{i-1} \left\{ \frac{D_j(\beta(n - j + 1, n - i + 2) - \beta(n - j + 1, n - i + 1))}{n - j + 1} \right\}.$$

Lemma 7. *D_i is an integer multiple of $(n - i + 1)$ for $1 \leq i \leq n$.*

Proof. We prove by induction a stronger claim that D_i is an integer multiple of $(n-i+1)!$ for $1 \leq i \leq n$. It is clear that $D_1 = D$ is an integer multiple of $n!$. Suppose D_i is an integer multiple of $(n-i+1)!$ for $1 \leq i \leq k$. We have $D_{k+1} = \sum_{j=1}^{k}(D_j/(n-j+1))(\beta(n-j+1, n-k+1) - \beta(n-j+1, n-k))$. Since the function β gives an integer output and $D_j/(n-j+1)$ is an integer multiple of $(n-j)!$, the summation gives an integer multiple of $(n-k)!$, which completes the induction.

The following lemmas give the performance of FF on the adversary sequence.

Lemma 8. *There exists some integer n such that the maximum number of bins used by FF is at least $2.45D$.*

Proof. After Stage n, FF uses $\sum_{i=1}^{n} D_i$ bins. We carry out the analysis on the value of $\sum_{i=1}^{n} D_i$ by actually computing the value of $\sum_{i=1}^{n} D_i$ with different values of n. We find that the increase in n generally leads to an increase in $\sum_{i=1}^{n} D_i$, though not monotonically. By letting $n = 21421$, and computing the values of D_i, we have $\sum_{i=1}^{n} D_i > 2.45D$.

Lemma 9. *The optimal off-line algorithm uses at most D bins at any time.*

Proof. We give an algorithm $\mathcal{O}$ to pack the items in the adversary such that $\mathcal{O}$ uses at most D bins over all time. In this proof permanent items refer to the items remain after Stage n and temporary items refer to the items depart in Stage n or before.

The algorithm $\mathcal{O}$ runs as follows. In each stage, when there are new items released, $\mathcal{O}$ packs the new items using the minimum number of empty bins such that a bin contains only permanent items, or temporary items that will depart in the same stage. We claim that $\mathcal{O}$ uses exactly D bins after each stage. In the initial stage, $\mathcal{O}$ packs the $D = n!$ permanent items into $(n-1)!$ bins and the $(n-1)n!$ temporary items to another $(n-1)(n-1)!$ bins. Totally, there are $n! = D$ occupied bins.

We prove that in each subsequent Stage i, for $2 \leq i \leq n$, the departed items produce D_i empty bin and $\mathcal{O}$ uses the D_i empty bins to pack the $D_i(n-i+1)$ items of size $1/(n-i+1)$ released. First, the number of empty bins produced in Stage i equals $\sum_{j=1}^{i-1} \lfloor D_j(\beta(n-j+1, n-i+2) - \beta(n-j+1, n-i+1))/(n-j+1) \rfloor$, which is equal to D_i because by Lemma 7, the term $D_j/(n-j+1)$ is an integer. Second, among the $D_i(n-i+1)$ items of size $1/(n-i+1)$ released in Stage i, the total size of those items that will depart in Stage p, for $i+1 \leq p \leq n$, is $D_i(\beta(n-j+1, n-p+2) - \beta(n-j+1, n-p+1))/(n-i+1)$, which is an integer because by Lemma 7, $D_i/(n-i+1)$ is an integer. Thus, we show that $\mathcal{O}$ can use D_i empty bins to pack all $D_i(n-i+1)$ items of size $1/(n-i+1)$. In other words, $\mathcal{O}$, and thus the optimal off-line algorithm, uses at most D bins at any time, and the lemma follows.

By Lemmas 8 and 9, the following theorem holds.

Theorem 2. *FF is at least 2.45-competitive.*

3.3 Performance of Other Any-Fit Algorithms

We show that BF and WF have a worse performance than FF, precisely, we show that BF and WF cannot be better than 3-competitive. On the other hand, we give the matching upper bounds. We prove in the Appendix that AF, including BF and WF, is 3-competitive.

Theorem 3. *Any-fit is 3-competitive.*

We give an adversary for WF as follows. Let k be an arbitrarily large integer constant and let $w = 2k$. The sequence contains $5k$ items, $m_1, m_2, \ldots, m_{5k}$, with m_i arriving at time i. There are three different sizes of the items: (1) $s_i = 1/2$ for $i = 1, 3, \ldots, 4k-1$; (2) $s_i = 1/w$ for $i = 2, 4, \ldots, 4k$; and (3) $s_i = 1$ for $i = 4k+1, 4k+2, \ldots, 5k$. All items of size $1/2$ depart at time $4k$ while items of size $1/w$ and 1 never depart. In the Appendix we show that the maximum number of bins used by WF is at least $3k$ and by the optimal off-line algorithm is at most $k+1$. For any $0 < \epsilon \leq 3/2$, setting $k = 3/\epsilon - 1$ results in the competitive ratio $3k/(k+1) > (3-\epsilon)$. Hence, we have the following theorem.

Theorem 4. *Worst-fit is no better than* 3*-competitive.*

Next, we give an adversary for BF. Let k be an arbitrarily large integer constant and let $w = 2k$. The adversary sequence consists of $2k$ stages, each lasts for 4 time units. Precisely, Stage i spans from time $4i+1$ to $4i+4$. There are three different sizes of items: $1/w$, $1/2$ and 1, all items of size $1/2$ will depart at some time while items of size $1/w$ and 1 never depart. Before Stage 0, two items are released, one with size $1/2$, and the other with size $1/w$. The stages proceed as follows. **Stage i**, for $0 \leq i \leq 2k-2$: At time $4i+1$, i items of size $1/2$ are released. At time $4i+2$, one more item of size $1/2$ is released. At time $4i+3$, all items of size $1/2$ released before time $4i+2$ depart, including those released at time $4i+1$ and the one released in Stage $i-1$. At time $4i+4$, a single item with size $1/w$ is released. **Stage $2k-1$**: At time $4(2k-1)+3$, the item with size $1/2$ released in Stage $(2k-2)$ departs. At time $4(2k-1)+4$, k items of size 1 are released. In the Appendix, we show that the maximum number of bins used by BF is at least $3k$ and that by the optimal off-line algorithm is at most $k+1$. For any $0 < \epsilon \leq 3/2$, setting $k = 3/\epsilon - 1$ results in the competitive ratio $3k/(k+1) > (3-\epsilon)$. Hence, we have the following theorem.

Theorem 5. *Best-fit is no better than* 3*-competitive.*

4 General Lower Bound

We give an adversary sequence of items such that the maximum number of bins used by any on-line algorithm is at least 2.428 times that used by the optimal off-line algorithm. First, we need the following notion. For any positive integers x and y, define $\lambda(x, y)$ to be the maximum number of items of size $1/x$ that can be packed into a bin containing an item of size $1/y$. Formally,

$$\lambda(x, y) = \max_{z \in \mathcal{Z}^+} \{z \mid z/x \leq 1 - 1/y\},$$

i.e., $\lambda(x, y) = x - \lceil x/y \rceil$. For example, if $x = 4$ and $y = 3$, then $\lambda(x, y) = 2$.

Consider any on-line algorithm $\mathcal{A}$. Let n be an integer and let $F = n!(n-1)!$. The adversary sequence consists of n stages and has the following properties: In each stage, some items released in the previous stage depart and a number of items of the same size are released. The arrival of items in the adversary sequence ensures that at the end of each stage, $\mathcal{A}$ has to use some additional bins to pack the items released in that stage. We are going to define a sequence of numbers F_i for $1 \leq i \leq n$, which is related the number of additional bins required in Stage i.

In Stage 1, Fn items of size $1/n$ are released. The algorithm $\mathcal{A}$ uses at least F bins to pack the Fn items. If $\mathcal{A}$ uses more than F bins, all items in bins other than the first F bins depart. We define F_1 to be F.

In each of the subsequent stages, i.e., Stage i, for $2 \leq i \leq n$, there are three steps. (1) For each occupied bin, all its items except the smallest one depart. (2) Let R_i be the total size of the items remained. (We will prove later, in the proof of Lemma 11, that R_i is indeed an integer.) The adversary then releases $(F - R_i)(n - i + 1)$ items of size $1/(n - i + 1)$. At this point, the total size of all items not yet departed, including those released in previous stages, is F. (3) Define

$$F_i = F - \sum_{j=1}^{i-1} F_j \left(\frac{1}{n - j + 1} + \frac{\lambda(n - j + 1, n - i + 1)}{n - i + 1} \right).$$

If $\mathcal{A}$ uses more than F_i additional bins in Stage i, all items packed into the additional bins other than the first F_i additional bins depart. Roughly speaking, F_i is the minimum number of additional bins required in Stage i; the term $\lambda(n - j + 1, n - i + 1)$ reflects the maximum number of items released in Stage i that can be packed into an occupied bin which was an additional bin in Stage j. We will prove this formally in Lemma 11. We first prove a property of F_i.

Lemma 10. *F_i is an integer multiple of $(n - i + 1)!(n - i)!$ for $1 \leq i \leq h$.*

Proof. We prove the lemma by induction. It is clear that $F_1 = F$ is an integer multiple of $n!(n-1)!$. Suppose F_i is an integer multiple of $(n-i+1)!(n-i)!$ for $1 \leq i \leq k$. We have $F_{k+1} = F - \sum_{j=1}^{k} F_j(1/(n-j+1) + \lambda(n-j+1, n-i+1)/(n-k))$. Since the function λ gives an integer output and $F_j/((n - j + 1)(n - k))$ is an integer multiple of $(n - k)!(n - k - 1)!$ as $k \geq j$, the summation gives an integer multiple of $(n - k)!(n - k - 1)!$, which completes the induction.

Lemma 11. *For $1 \leq i \leq n$, $\mathcal{A}$ uses no less than F_i additional bins in Stage i.*

Proof. In Stage 1, it is clear that $\mathcal{A}$ uses at least $F_1 = F$ additional bins. We show by induction that in Step (2) of Stage i for $2 \leq i \leq n$, $\mathcal{A}$ also uses at least F_i additional bins. Assume that it is true for $i = k$. Before Step (2) of Stage $k+1$, $\mathcal{A}$ already has F_j bins containing a single item of size $1/(n-j+1)$ for $1 \leq j \leq k$. Therefore, $R_{k+1} = \sum_{j=1}^{k} F_j/(n - j + 1)$. We can see that R_{k+1} is an integer as

F_j is an integer multiple of $(n-j+1)!(n-j)!$. The number of items of size $1/(n-k)$ released is $(F - R_{k+1})(n-k) = (F - \sum_{j=1}^{k} F_j/(n-j+1))(n-k)$. The number of items that can be packed into the occupied bins is equal to $\sum_{j=1}^{k} F_j \cdot \lambda(n-j+1, n-k)$. Therefore, the number of additional bins needed in Stage $k+1$ is at least

$$F - \sum_{j=1}^{k} \frac{F_j}{n-j+1} - \sum_{j=1}^{k} \frac{F_j \cdot \lambda(n-j+1, n-k)}{n-k}$$
$$= F - \sum_{j=1}^{k} F_j \left(\frac{1}{n-j+1} + \frac{\lambda(n-j+1, n-k)}{n-k} \right) = F_{k+1}.$$

This induction is completed.

Lemma 12. *There exists some integer n such that the maximum number of bins used by $\mathcal{A}$ is at least $2.428F$.*

Proof. After Stage n, $\mathcal{A}$ uses $\sum_{i=1}^{n} F_i$ bins. We carry out the analysis on the value of $\sum_{i=1}^{n} F_i$ by actually computing the value of $\sum_{i=1}^{n} F_i$ with different values of n. We find that the increase in n results an increase in $\sum_{i=1}^{n} F_i$ monotonically. In particular, by letting $n = 12794$, and computing the values of F_i, we have $\sum_{i=1}^{n} F_i > 2.428F$.

Theorem 6. *Any on-line algorithm is at least 2.428-competitive.*

Proof. Similar to Lemma 9, we can show that the optimal off-line algorithm uses at most F bins. Together with Lemma 12, the theorem follows.

References

1. A. Bar-Noy and R. E. Ladner. Windows scheduling problems for broadcast systems. *SIAM J. Comput.*, 32(4):1091–1113, 2003.
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. W.-T. Chan and P. W. H. Wong. On-line windows scheduling of temporary items. In R. Fleischer and G. Trippen, editors, *ISAAC*, volume 3341 of *Lecture Notes in Computer Science*, pages 259–270. Springer, 2004.
5. 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.
6. E. G. Coffman, Jr., G. Galambos, S. Martello, and D. Vigo. Bin pakcing approximation algorithms: Combinatorial analysis. In *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 1998.
7. E. G. Coffman, Jr., M. Garey, and D. Johnson. Bin packing with divisible item sizes. *Journal of Complexity*, 3:405–428, 1987.

8. E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.
9. 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.
10. 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.
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. Z. Ivkovic and E. L. Lloyd. Fully dynamic algorithms for bin packing: Being (mostly) myopic helps. *SIAM J. Comput.*, 28(2):574–611, 1998.
13. M. B. Richey. Improved bounds for harmonic-based bin packing algorithms. *Discrete Applied Mathematics*, 34:203–227, 1991.
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. *Inf. Process. Lett.*, 43(5):277–284, 1992.

Appendix

Proof of Theorem 3: Consider any input seqeunce σ. Suppose AF uses at most n bins. The proof is based on two notions. (1) Let t_1 be a time instance such that AF uses n occupied bins, and n_1 be the number of occupied bins with item of size 1 at time t_1. Notice that the optimal off-line algorithm uses at least n_1 occupied bins at time t_1. (2) Let n_2 be the largest integer such that AF packs an item m of size $1/2$ or less into an empty bin with label n_2 and suppose this happens at t_2. At time t_2, all bins with labels smaller than n_2 have load greater than $1/2$; otherwise, AF can pack m to one of these bins. In that case, the optimal off-line algorithm uses at least $\lceil n_2/2 \rceil$ bins at time t_2. By the definition of n_2, at time t_1, every bin with label greater than n_2 contains an item of size greater than $1/2$, i.e., 1. Hence, we have $n \leq n_1 + n_2$. On the other hand, the optimal off-line algorithm uses at least $\max\{n_1, \lceil n_2/2 \rceil\}$ bins. By simple arithmetic, we have $n \leq 3 \max\{n_1, \lceil n_2/2 \rceil\}$ (the worst case happens when $n_1 = \lceil n_2/2 \rceil$).

Proof of Theorem 4: We first describe how WF packs the $5k$ items in the adversary. WF packs the item m_{2j-1} of size $1/2$ and m_{2j} of size $1/w$ to the same bin with label j, for $1 \leq j \leq 2k$. After all items of size $1/2$ depart, there are $2k$ occupied bins; k more bins are needed for the items of size 1. Therefore, WF uses $3k$ bins.

On the other hand, the optimal off-line algorithm can use a single bin to pack all the items of size $1/w$ and k bins to pack the items of size $1/2$. This packing uses $k+1$ bins. After all the items of size $1/2$ depart, the k bins can be used to pack the items of size 1. Thus, the optimal off-line algorithm uses at most $k+1$ bins, and the competitive ratio of WF is at least $3k/(k+1)$. For any $0 < \epsilon \leq 3/2$, picking k to be $3/\epsilon - 1$ implies that WF is no better than $(3-\epsilon)$-competitive.

Proof of Theorem 5: We first describe how BF packs the items in the adversary. We claim that for $1 \leq i \leq 2k-1$, at the beginning of Stage i, BF uses $i+1$

bins, one of them has load $1/2 + 1/w$, and the other i bins each has load $1/w$. The base case for Stage 1 can be verified easily. Suppose the claim is true for some $i \geq 1$. Consider what happens in Stage i. At time $4i + 1$, BF packs each of the i new items into the i bins with load $1/w$. All the $i + 1$ occupied bins now have load $1/2 + 1/w$. The item of size $1/2$ released at time $4i + 2$ must then be packed into a new bin. After the departure of items of size $1/2$ at time $4i + 3$, we are left with a bin with load $1/2$ and $i + 1$ bins each with load $1/w$. When the item of size $1/w$ is released at time $4i + 4$, BF packs it into the bin with load $1/2$. Then, at the beginning of Stage $i+1$, BF uses $i+2$ bins where $i+1$ of them have load $1/w$ and one has load $1/2 + 1/w$, and the claim follows. Finally, in Stage $2k - 1$, BF needs k more bins, and thus uses at least $2k + k = 3k$ bins.

On the other hand, the optimal off-line algorithm can reserve a single bin for the $2k$ items of size $1/w$. At any time, there are at most $2k$ items of size $1/2$ which can be packed into k bins. In the final stage, all these items depart and the k bins can be used for the items of size 1. Hence, the optimal off-line algorithm uses at most $k + 1$ bins, and the competitive ratio of BF is at least $3k/(k + 1)$. For any $0 < \epsilon \leq 3/2$, picking k to be $3/\epsilon - 1$ implies that BF is no better than $(3 - \epsilon)$-competitive.