

# Dynamic Bin Packing of Unit Fractions Items <sup>\*</sup>

Joseph Wun-Tat Chan<sup>†</sup>    Tak-Wah Lam<sup>‡</sup>    Prudence W.H. Wong<sup>§</sup>

## 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. In particular, 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 2.4942. We also show that no on-line algorithm is better than 2.428-competitive.

## 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 [7, 10, 11]). 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* [9], 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.

There is a long history of results for the classical bin packing problem and its variants [7, 10, 11]. Most of the previous works considered the “static” bin packing in the sense that items will not depart. In this “static” model, the off-line bin packing problem is NP-hard [12]. For the on-line version of the problem, each item must be assigned to

---

<sup>\*</sup>A preliminary version of this paper appeared in “The 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)”

<sup>†</sup>Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK, jchan@dcs.kcl.ac.uk. This research was supported in part by Hong Kong RGC Grant HKU-5172/03E when the author was with the Department of Computer Science, University of Hong Kong, Hong Kong.

<sup>‡</sup>Department of Computer Science, University of Hong Kong, Hong Kong, twlam@cs.hku.hk.

<sup>§</sup>Department of Computer Science, University of Liverpool, UK, pwong@csc.liv.ac.uk. This research was supported in part by Nuffield Foundation Grant NAL/01004/G.

a bin, without the knowledge of subsequent items. Moreover, no migration of items is allowed, i.e., items are not allowed to move from one bin to another. The performance of an on-line algorithm 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. 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 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 [8] (where each possible item size can be divided by the next smaller item size) and discrete item sizes [6] (where possible item sizes are  $\{1/k, 2/k, \dots, j/k\}$  for some  $1 \leq j \leq k$ ).

Dynamic bin packing is a generalization of the classical bin packing problem introduced by Coffman et al. [9]. This generalization assumes that items may depart at arbitrary time. 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 lying 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. [9] also gave a lower bound of 2.388 when the off-line algorithm is not allowed to re-pack the items. Ivkovic and Lloyd [13] 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 dynamic bin packing of unit fractions items, the main contributions 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 2.4942. 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 by Coffman et al. [9] 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 integer. Each item represents a piece of information to be broadcast to all clients. Assume that all items are of the same length, which take 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

Algorithms	Upper bounds	Lower bounds
First-fit	2.4942	2.45
Best-fit	3	3
Worst-fit	3	3
Any-fit	3	2.428
Any on-line algorithms	–	2.428

Table 1: Summary of results

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 the definitions of the problem and the family of any-fit algorithms. Section 3 analyzes the performance of the family of any-fit algorithms. This includes the upper and lower bounds for first-fit (Sections 3.1 and 3.2, respectively), and the upper and lower bounds for best-fit and worst-fit (Section 3.3). Section 4 gives a lower bound for any on-line algorithm. Finally, some concluding remarks are given in Section 5.

## 2 Preliminaries

In this section we give the definition of the dynamic bin packing problem with unit fractions items and the necessary notations for further discussion. There is a sequence of items to be packed into bins of unit-capacity. The 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  $m_i$  is associated 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 refers to the total size of items that are currently assigned to the bin and have not yet departed, and this load must be at most 1 due to unit bin capacity. Migration is not allowed in the sense that once an item is assigned to a bin, it cannot be re-assigned 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 competitive ratio. Given a sequence  $\sigma$  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  $\sigma$ , 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.

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 if otherwise. The algorithms differ in the rule of choosing the occupied bin for the newly arrived item.

When a new item  $m_i$  of size  $1/w_i$  arrives, if there are occupied bins 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 bin which has been occupied for the longest time.

**Best-fit (BF)** assigns  $m_i$  to the heaviest loaded bin; ties are broken arbitrarily.

**Worst-fit (WF)** assigns  $m_i$  to the lightest loaded bin; ties are broken arbitrarily.

As pointed out by Coffman et al. [9], for analyzing the performance of the first-fit, it suffices to consider the input sequences with the following two properties.

1. FF uses the maximum number of bins when the last item is packed but not before.
2. No occupied bin ever becomes empty during the execution of FF on input sequences satisfying the first property. Otherwise, if there is an occupied bin that becomes empty, we can consider the same sequence of items without all the items that are packed to that bin before the bin becomes empty. First-fit will work out the same final packing and the maximum number of bins used will remain unchanged.

By the second property, we can label the occupied bins by the order they become occupied such that bin  $i$  refers to the  $i$ -th bin used by first-fit. It is obvious that the labels never change. Moreover, among a set of occupied bins, the bin that has been occupied for the longest time is the bin with the smallest bin label.

### 3 Performance of the family of any-fit algorithms

In this section we analyze the performance of the family of any-fit algorithms. In Sections 3.1 and 3.2 we give an upper bound of 2.4942 and a lower bound of 2.45 for the competitive ratio of FF, respectively. 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 of first-fit

The upper bound of first-fit is proved in a case analysis. We compute the performance ratio of first-fit against the optimal algorithm in each case, thus obtain a worst case bound of 2.4942. The building blocks of the general case, where individual cases are identified, requires some new notations that are defined in the following paragraphs.

Let  $x$  and  $y$  be any positive integers. Suppose that a bin is already packed with some items whose sizes are chosen from the set  $\{1, 1/2, \dots, 1/x\}$ . We define a notion of the minimum load of such a bin that an additional item of size  $1/y$  cannot be fitted into the bin. This minimum load is denoted precisely by a function,

$$\alpha\langle x, y \rangle = \min_{1 \leq j \leq x \text{ and } n_j \geq 0} \{n_1 + n_2/2 + \dots + n_x/x \mid n_1 + n_2/2 + \dots + n_x/x > 1 - 1/y\}.$$

$\alpha\langle x, y \rangle$	$y = 1$	2	3	4	5	6
$x = 1$	1	1	1	1	1	1
2	1/2	1	1	1	1	1
3	1/3	2/3	5/6	5/6	5/6	5/6
4	1/4	7/12	3/4	5/6	5/6	5/6
5	1/5	8/15	7/10	47/60	5/6	5/6
6	1/6	8/15	7/10	23/30	49/60	17/20

Table 2: Values of  $\alpha\langle x, y \rangle$  for  $1 \leq x, y \leq 6$

For example, when  $x = 4$  and  $y = 2$ , we have  $\alpha\langle 4, 2 \rangle = 7/12$  and correspondingly  $n_1 = 0$ ,  $n_2 = 0$ ,  $n_3 = 1$  and  $n_4 = 1$ . We have the following two trivial facts about  $\alpha\langle x, y \rangle$ :

$$\alpha\langle x, y \rangle > 1 - 1/y \quad \text{and} \quad \alpha\langle x, 1 \rangle = 1/x. \quad (1)$$

The values of  $\alpha\langle x, y \rangle$  for  $1 \leq x, y \leq 6$  are given in Table 2; some of these values are required for subsequent analysis.

With respect to an input sequence  $\sigma$ , 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 bin  $b_1$  is of size  $1/r_1$ . We define  $b_i$  and  $r_i$  iteratively 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 bin  $b_i$ . The size of the smallest item that FF ever packs into bin  $b_i$  is denoted as  $1/r_i$ . Let  $k$  be the largest value of  $i$  that  $b_i$  and  $r_i$  can be defined. Notice that  $b_1 > b_2 > \dots > b_k$  and  $r_1 < r_2 < \dots < r_k$ .

Now we are ready to describe the general case and the lower bound of the number of bins used by the optimal algorithm in the general case, which is the total load of all bins at some particular time instance. Consider the time instance  $t_k$  when FF packs an item  $X$  of size  $1/r_k$  into bin  $b_k$ . Since  $k$  is the largest index of  $b_i$  that can be defined, no item of size smaller than  $1/r_k$  has ever appeared nor been packed into any bin, in particular bins 1 to  $b_k - 1$ . Together with the fact that FF packs  $X$  into bin  $b_k$  but not bins 1 to  $b_k - 1$ , we can conclude that at time  $t_k$ , each of bins 1 to  $b_k - 1$  must have a load at least  $\alpha\langle r_k, r_k \rangle$ . Including the item  $X$ , the total load of all the occupied bins at time  $t_k$  is at least

$$\ell_k = 1/r_k + (b_k - 1) \cdot \alpha\langle r_k, r_k \rangle. \quad (2)$$

Next, for any integer  $1 \leq i \leq k - 1$ , consider at time  $t_i$  when FF packs an item  $X_i$  of size  $1/r_i$  into bin  $b_i$ . 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 1 to  $b_k$  must have load at least  $\alpha\langle r_k, r_i \rangle$ ; (2) for any integer  $p$  with  $i + 1 \leq p < k$ , each of the bins  $b_{p+1} + 1$  to  $b_p$  must have a load at least  $\alpha\langle r_p, r_i \rangle$ ; and (3) each of the bins  $b_{i+1} + 1$  to  $b_i - 1$  must have a load at least  $\alpha\langle r_i, r_i \rangle$ . Including the item  $X_i$ , the total load of all occupied bins at time  $t_i$  is at least

$$\ell_i = 1/r_i + (b_i - b_{i+1} - 1) \cdot \alpha\langle r_i, r_i \rangle + \sum_{p=i+1}^{k-1} (b_p - b_{p+1}) \cdot \alpha\langle r_p, r_i \rangle + b_k \cdot \alpha\langle r_k, r_i \rangle. \quad (3)$$

Let  $\ell = \max_{1 \leq i \leq k} \ell_i$ . The number of bins used by the optimal off-line algorithm is at least  $\ell$ . On the other hand, the maximum number of bins used by FF is  $b_1$ . By the

case analysis given later, we prove that  $b_1 \leq 2.4942\ell + 1$ , which implies that first-fit is 2.4942-competitive.

The case analysis studies all possible values of  $k$  and  $r_i$  for  $1 \leq i \leq k$ , which are partitioned into a number of cases. The idea for the case division is as follows. We can first partition all cases into two:  $\{k = 1\}$  and  $\{k \geq 2\}$ . Then the case of  $\{k \geq 2\}$  can be further divided into three subcases  $\{k = 2, r_1 = 1\}$ ,  $\{k \geq 2, r_1 \geq 2\}$ , and  $\{k \geq 3, r_1 = 1\}$ . Similarly, case of  $\{k \geq 3, r_1 = 1\}$  can be further divided into three subcases  $\{k = 3, r_1 = 1, r_2 = 2\}$ ,  $\{k \geq 3, r_1 = 1, r_2 \geq 3\}$ , and  $\{k \geq 4, r_1 = 1, r_2 = 2\}$ . Repeating this method of case division, we can always partition, for any integer  $c \geq 1$ , all cases into three types of cases. Note that there is only one case instance for the Type (3).

**Type (1):**  $\{k = i, r_1 = 1, r_2 = 2, \dots, r_{i-1} = i - 1\}$  for  $1 \leq i \leq c$ .

**Type (2):**  $\{k \geq i, r_1 = 1, r_2 = 2, \dots, r_{i-2} = i - 2, r_{i-1} \geq i\}$  for  $2 \leq i \leq c$ .

**Type (3):**  $\{k \geq c + 1, r_1 = 1, r_2 = 2, \dots, r_{c-1} = c - 1\}$ .

We analyze for each case the corresponding relation between  $b_1$  and  $\ell$ . Precisely, in each case we obtain an inequality in the form of  $b_1 \leq x_1\ell + x_2$  for some constants  $x_1$  and  $x_2$ . The competitive ratio of first-fit can be upper bounded by the maximum value of  $x_1$  over all cases. Since this method of case division consists of a variable  $c$ , which can be any positive integer, we first describe how to determine the value of  $c$  so as to achieve the best possible competitive ratio by this method. For cases of Types (1) and (2), the increase of value of  $c$  implies more cases to consider, which also means a possibly larger maximum value of  $x_1$  that is obtained. On the contrary, the increase of value of  $c$  for Type (3) case means a more specific case, thus a possibly smaller value of  $x_1$  is obtained. Since we want to minimize the overall maximum value of  $x_1$ , we find a value of  $c$  that balances the two sides. By trying the different values of  $c$  starting from  $c = 1, 2, \dots$ , it happens that assigning  $c = 8$  gives the smallest maximum value of  $x_1$ . With  $c = 8$ , the exact cases we considered and their corresponding values of  $x_1$  obtained, i.e., the competitive ratios, are summarized in Table 3.

In the following, we show how to obtain the relation  $b_1 \leq x_1\ell + x_2$  for the cases in Table 3. We pick the Type (2) case  $k \geq 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 \geq 5$  as an example. The other cases can be analyzed similarly using the same approach. By using Equations (1), (2) and (3), we can obtain an inequality for each of  $\ell_i$  for  $1 \leq i \leq 5$ , as follows.

$$\begin{aligned}
\ell_1 &= \frac{1}{r_1} + (b_1 - b_2 - 1)\alpha\langle r_1, r_1 \rangle + \sum_{p=2}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, r_1 \rangle + b_k\alpha\langle r_k, r_1 \rangle \\
&\geq 1 + (b_1 - b_2 - 1)\alpha\langle 1, 1 \rangle + (b_2 - b_3)\alpha\langle 2, 1 \rangle + (b_3 - b_4)\alpha\langle 3, 1 \rangle + (b_4 - b_5)\alpha\langle r_4, 1 \rangle \\
&\geq 1 + (b_1 - b_2 - 1) + \frac{1}{2}(b_2 - b_3) + \frac{1}{3}(b_3 - b_4) + \frac{1}{r_4}(b_4 - b_5) \\
&\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \left(\frac{1}{3} - \frac{1}{r_4}\right)b_4 - \frac{1}{r_4}b_5
\end{aligned}$$

$$\begin{aligned}
\ell_2 &= \frac{1}{r_2} + (b_2 - b_3 - 1)\alpha\langle r_2, r_2 \rangle + \sum_{p=3}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, r_2 \rangle + b_k\alpha\langle r_k, r_2 \rangle \\
&= \frac{1}{2} + (b_2 - b_3 - 1)\alpha\langle 2, 2 \rangle + (b_3 - b_4)\alpha\langle 3, 2 \rangle + \sum_{p=4}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, 2 \rangle + b_k\alpha\langle r_k, 2 \rangle \\
&\geq \frac{1}{2} + (b_2 - b_3 - 1) + \frac{2}{3}(b_3 - b_4) + \frac{1}{2}b_4 \\
&\geq b_2 - \frac{1}{3}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\
\ell_3 &= \frac{1}{r_3} + (b_3 - b_4 - 1)\alpha\langle r_3, r_3 \rangle + \sum_{p=4}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, r_3 \rangle + b_k\alpha\langle r_k, r_3 \rangle \\
&\geq \frac{1}{3} + \frac{5}{6}(b_3 - b_4 - 1) + \frac{2}{3}b_4 \\
&\geq \frac{5}{6}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\
\ell_4 &= \frac{1}{r_4} + (b_4 - b_5 - 1)\alpha\langle r_4, r_4 \rangle + \sum_{p=5}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, r_4 \rangle + b_k\alpha\langle r_k, r_4 \rangle \\
&\geq \frac{1}{r_4} + (b_4 - 1)(1 - \frac{1}{r_4}) \\
&\geq (1 - \frac{1}{r_4})b_4 - (1 - \frac{2}{r_4})
\end{aligned}$$

If  $k = 5$ , by Equations (1) and (2), we have

$$\ell_5 = \frac{1}{r_5} + (b_5 - 1)\alpha\langle r_5, r_5 \rangle \geq (b_5 - 1)(1 - \frac{1}{r_5}) \geq \frac{5}{6}b_5 - \frac{5}{6}$$

Otherwise, by Equations (1) and (3), we can obtain the same inequality.

$$\begin{aligned}
\ell_5 &= \frac{1}{r_5} + (b_5 - b_6 - 1)\alpha\langle r_5, r_5 \rangle + \sum_{p=6}^{k-1} (b_p - b_{p+1})\alpha\langle r_p, r_5 \rangle + b_k\alpha\langle r_k, r_5 \rangle \\
&\geq (b_5 - 1)(1 - \frac{1}{r_5}) \geq \frac{5}{6}b_5 - \frac{5}{6}
\end{aligned}$$

Since  $\ell \geq \max_{1 \leq i \leq 5} \{\ell_i\}$ , we can substitute  $\ell$  for  $\ell_i$  in the above five inequalities. Solving the inequalities, we have  $b_1 \leq \frac{143(r_4)^2 - 102r_4 - 72}{60r_4(r_4 - 1)}\ell + \frac{1035}{1200} \leq \frac{2993}{1200}\ell + \frac{1035}{1200} \leq 2.4942\ell + 0.8625$  because  $\frac{143(r_4)^2 - 102r_4 - 72}{60r_4(r_4 - 1)} \leq \frac{2993}{1200} < 2.4942$  for  $r_4 \geq 5$ .

The sets of inequalities for all the other cases are shown in the Appendix. After computing the relation  $b_1 \leq x_1\ell + x_2$  for each of the cases, as shown in Table 3, we have the maximum value for  $x_1$  be 2.4942 and  $x_2$  be 1. Thus, we have the relation  $b_1 \leq 2.4942\ell + 1$  for all cases, and that implies the following theorem.

**Theorem 1.** *First-fit is 2.4942-competitive.*

### 3.2 Lower bound of first-fit

We derive a lower bound for FF by constructing an adversary sequence so that the maximum number of bins used by FF is at least 2.45 times that used by the optimal off-line algorithm. First, consider some definitions with positive integers  $x$  and  $y$ , which are used when describing the adversary. Suppose a bin contains only items of size  $1/x$ . Define  $\beta\langle x, y \rangle$  to be the minimum number of items the bin must contain so that an additional item of size  $1/y$  cannot be fitted into the bin. Precisely,

$$\beta\langle x, y \rangle = \min_{\text{Integer } z \geq 1} \{z \mid z/x > 1 - 1/y\},$$

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

Type (1) cases	$x_1$	$x_2$
$k = 1$	1	1
$k = 2, r_1 = 1$	2	1
$k = 3, r_1 = 1, r_2 = 2$	2.25	1
$k = 4, r_1 = 1, r_2 = 2, r_3 = 3$	2.3834	0.9334
$k = 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4$	2.4309	0.9017
$k = 6, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 = 5$	2.456	0.8808
$k = 7, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 = 5, r_6 = 6$	2.4635	0.874
Type (2) cases	$x_1$	$x_2$
$k \geq 2, r_1 \geq 2$	2	1
$k \geq 3, r_1 = 1, r_2 \geq 3$	2.4445	0.6667
$k \geq 4, r_1 = 1, r_2 = 2, r_3 \geq 4$	2.4792	0.8334
$k \geq 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 \geq 5$	2.4942	0.8625
$k \geq 6, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 \geq 6$	2.4935	0.8669
$k \geq 7, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 = 5, r_6 \geq 7$	2.4926	0.8646
Type (3) cases	$x_1$	$x_2$
$k \geq 8, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 = 5, r_6 = 6$	2.4939	0.864

Table 3: Division of cases and the parameters  $x_1$  and  $x_2$  in the relation  $b_1 \leq x_1 \ell + x_2$  for each of the cases.

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 total 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 force 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  $\beta\langle x, n - i + 2 \rangle$  such items.

The invariant holds at the beginning of Stage 2 because each occupied bin contains  $\beta\langle n, n \rangle = n$  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  $\beta\langle x, n - i + 2 \rangle - \beta\langle x, n - i + 1 \rangle$  items and let them depart. In other words, there are  $\beta\langle x, n - i + 1 \rangle$  items remained. Let  $D_i$  be the total size of all the departed items in Stage  $i$ . Note that  $D_i$  is an integer, as will be proved in Lemma 2.
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  $\beta\langle x, n - i + 1 \rangle$  such items, none of the newly released items can be packed into any occupied bin. Therefore, in Stage  $i$ , FF will use  $D_i$  new bins to pack all these items, where each new bin contains  $n - i + 1 = \beta\langle n - i + 1, n - i + 1 \rangle$  items. Thus, the invariant also holds at the beginning of Stage  $i + 1$ .

To analyze the adversary on FF, let  $D_1 = D$ , which is the number of new bins used in Stage 1. From the above discussion, we can see that the number of new bins required for items of size  $1/(n - i + 1)$  in Stage  $i$  is  $D_i$ . It is clear that no occupied bin will become empty during the adversary. Thus at the beginning of Stage  $i$ , there are  $D_j$  occupied bins each with items of size  $1/(n - j + 1)$  for all  $1 \leq j \leq i - 1$ . The total size of all the items departed in Stage  $i$  is

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

**Lemma 2.**  $D_i$  is an integer multiple of  $(n - i + 1)!$  for  $1 \leq i \leq n$ .

*Proof.* We prove the lemma by induction. 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  $D_{k+1} = \sum_{j=1}^k (D_j/(n - j + 1))(\beta\langle n - j + 1, n - k + 1 \rangle - \beta\langle n - j + 1, n - k \rangle)$ . Since the function  $\beta$  gives an integer output and  $D_j/(n - j + 1)$  is an integer multiple of  $(n - j)!$ , the summation gives an integer multiple of  $(n - k)!$ . The induction is completed.  $\square$

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

**Lemma 3.** *There exists some integer  $n$  for the adversary such that the maximum number of bins used by FF is at least  $2.45D$ .*

*Proof.* The adversary takes on a parameter  $n$ . After Stage  $n$ , FF uses  $\sum_{i=1}^n D_i$  bins. We compute the ratio  $(\sum_{i=1}^n D_i)/D$  for different values of  $n$ . It is found that the increase in  $n$  generally leads to an increase in the ratio, though not monotonically. The ratio quickly converges to a value around 2.45. See Figure 1 for the ratio when the value of  $n$  is at most 200. In further computation, we found that when  $n = 21421$ , we have  $\sum_{i=1}^n D_i > 2.45D$ .  $\square$

**Lemma 4.** *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 new items are released,  $\mathcal{O}$  packs the new items using the minimum number of empty bins such that a bin contains solely 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 into 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 turn  $D_i$  occupied bins empty and  $\mathcal{O}$  uses these  $D_i$  empty bins to pack the  $D_i(n - i + 1)$  newly released items of size  $1/(n - i + 1)$ . First, the number of occupied bins become empty in Stage  $i$  equals  $\sum_{j=1}^{i-1} \lfloor D_j(\beta\langle n - j + 1, n - i + 2 \rangle - \beta\langle n - j + 1, n - i + 1 \rangle)/(n - j + 1) \rfloor$ , which is equal to  $D_i$  because by Lemma 2 the term  $D_j/(n - j + 1)$  is an integer. Second,

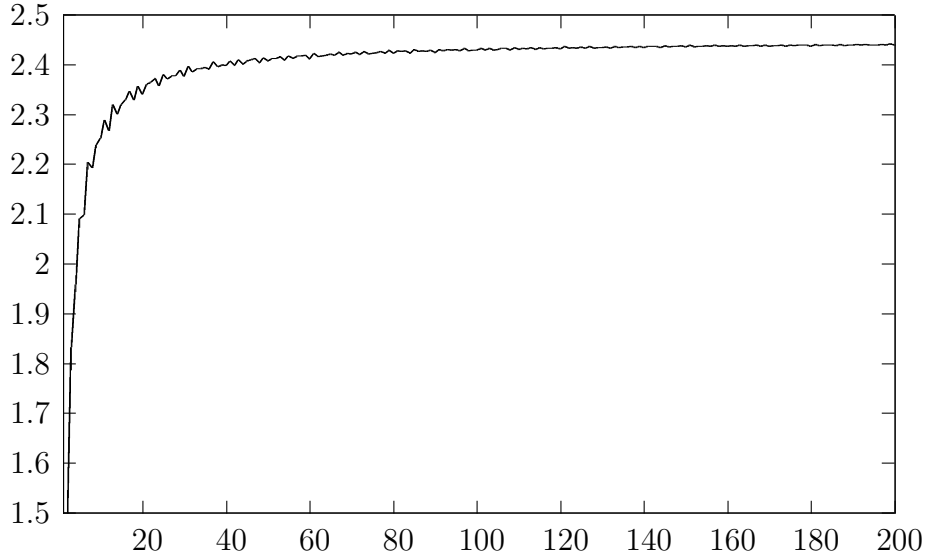


Figure 1: The ratio between  $\sum_{i=1}^n D_i$  and  $D$  when the value of  $n$  increases

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  $D_i(\beta\langle n - j + 1, n - p + 2 \rangle - \beta\langle n - j + 1, n - p + 1 \rangle)/(n - i + 1)$ , which is an integer because by Lemma 2  $D_i/(n - i + 1)$  is an integer. Thus, we have shown that  $\mathcal{O}$  can use the  $D_i$  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.  $\square$

By Lemmas 3 and 4, the following theorem holds.

**Theorem 5.** *First-fit 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 that AF, including BF and WF, is 3-competitive.

**Theorem 6.** *Any-fit is 3-competitive.*

*Proof.* Consider any input sequence  $\sigma$ . Suppose that AF uses at most  $n$  bins. The proof is based on two notions.

1. Let  $t_1$  be the time when AF uses  $n$  occupied bins, and  $m$  be the number of occupied bins containing item of size 1 at time  $t_1$ . Note that the optimal off-line algorithm uses at least  $m$  occupied bins at time  $t_1$ .
2. Let  $k$  be the largest integer such that AF packs an item  $X$  of size  $s$  with  $s \leq 1/2$  into a new bin and there are already  $k - 1$  occupied bins. Suppose this happens at time  $t_2$ . At time  $t_2$ , each of the  $k - 1$  occupied bins has load greater than  $1 - s$ ;

otherwise, AF can pack  $X$  into one of these bins, rather than a new bin. Thus, the total load is at least  $(k-1)(1-s) + s \geq k/2$ , and the optimal off-line algorithm uses at least  $\lceil k/2 \rceil$  bins.

We claim that  $k \geq n - m$ . At time  $t_1$ , there must be at most  $k$  occupied bins containing an item of size  $1/2$  or less; otherwise, there is an item of size  $1/2$  or less packed into a new bin when there are already  $k$  or more occupied bins, which contradicts to the definition of Notion (2). By simple arithmetic, we have  $n \leq m + k \leq 3 \cdot \max\{m, \lfloor k/2 \rfloor\}$  (the worst case happens when  $m = \lfloor k/2 \rfloor$ ). Since the optimal off-line algorithm uses at least  $\max\{m, \lfloor k/2 \rfloor\}$  bins, AF is 3-competitive.  $\square$

We give an adversary for WF as follows. Let  $k$  be an arbitrarily large integer and let  $w = 2k$ . The sequence consists of  $5k$  items,  $X_1, X_2, \dots, X_{5k}$ , with  $X_i$  arriving at time  $i$ . There are three different sizes of items: (1)  $s_i = 1/2$  for  $i = 1, 3, \dots, 4k - 1$ ; (2)  $s_i = 1/w$  for  $i = 2, 4, \dots, 4k$ ; and (3)  $s_i = 1$  for  $i = 4k + 1, 4k + 2, \dots, 5k$ . All items of size  $1/2$  depart at time  $4k$  while items of size  $1/w$  and  $1$  never depart. 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 7.** *Worst-fit is no better than 3-competitive.*

*Proof.* We first describe how WF packs the  $5k$  items in the adversary. WF packs the item  $X_{2j-1}$  of size  $1/2$  and  $X_{2j}$  of size  $1/w$  to the same bin  $j$ , 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.  $\square$

Next, we give an adversary for BF. Let  $k$  be an arbitrarily large integer 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 of 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.

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 8.** *Best-fit is no better than 3-competitive.*

*Proof.* 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 have load  $1/w$ . The base case for Stage 0 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 new 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.  $\square$

## 4 General lower bound

We give an adversary sequence 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 definition in order to describe the adversary. For any positive integers  $x$  and  $y$ , define  $\lambda(x, y)$  to be the maximum number of items of size  $1/y$  that can be packed into a bin containing only an item of size  $1/x$ . Precisely,

$$\lambda(x, y) = \max_{\text{Integer } z \geq 0} \{z \mid z/y \leq 1 - 1/x\},$$

i.e.,  $\lambda(x, y) = y - \lceil y/x \rceil$ . For example, if  $x = 3$  and  $y = 4$ , then  $\lambda(x, y) = 2$ . For instance, if  $1/y \geq 1/x$ , i.e.,  $y \leq x$ , then  $\lambda(x, y) = y - 1$ .

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. For any on-line algorithm  $\mathcal{A}$ , the new items in each stage ensure that  $\mathcal{A}$  has to use some new bins to pack the items. In Lemma 10 we give a lower bound on the number of new bins used in each stage.

In Stage 1,  $Fn$  items of size  $1/n$  are released. Any 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. Let  $F_1 = F$ . 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 item size of all items remained. (We will prove later, in the proof of Lemma 10, that  $R_i$  is indeed an integer.) The adversary then releases  $(F - R_i)(n - i + 1)$  items of size  $1/(n - i + 1)$ . Note that at this point the total item size of existing items becomes  $F$  again.
3. Define

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

If  $\mathcal{A}$  uses more than  $F_i$  new bins in Stage  $i$ , all items packed into the new bins other than the first  $F_i$  new bins depart. Roughly speaking,  $F_i$  is the minimum number of new bins required in Stage  $i$ ; for an occupied bin which was a new bin in Stage  $j$  for  $j < i$ , before the new items released there remains an item of size  $1/(n - j + 1)$  and the bin can accommodate at most  $\lambda(n - j + 1, n - i + 1) = n - i$  new items. We will prove this formally in Lemma 10.

The following lemma implies that  $F_i$  is an integer.

**Lemma 9.**  $F_i$  is an integer multiple of  $(n - i + 1)!(n - i)!$  for  $1 \leq i \leq n$ .

*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  $1 \leq i \leq k$ . Consider the definition  $F_{k+1} = F - \sum_{j=1}^k F_j(1/(n - j + 1) + (n - k - 1)/(n - k))$ . As  $F_j/((n - j + 1)(n - k))$  is an integer multiple of  $(n - k)!(n - k - 1)!$  for  $j \leq k$ , the summation gives an integer multiple of  $(n - k)!(n - k - 1)!$ , which completes the induction.  $\square$

**Lemma 10.** For  $1 \leq i \leq n$ , any on-line algorithm  $\mathcal{A}$  uses no less than  $F_i$  new bins in Stage  $i$ .

*Proof.* In Stage 1, it is clear that  $\mathcal{A}$  uses at least  $F_1 = F$  new bins. We show by induction that in Step (2) of Stage  $i$  for  $2 \leq i \leq n$ ,  $\mathcal{A}$  also uses at least  $F_i$  new 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  $\sum_{j=1}^k F_j \cdot \lambda(n - j + 1, n - k) = \sum_{j=1}^k F_j(n - k - 1)$ . Therefore, the number of new bins required in Stage  $k + 1$  is at least

$$\begin{aligned}
&F - \sum_{j=1}^k \frac{F_j}{n - j + 1} - \sum_{j=1}^k \frac{F_j(n - k - 1)}{n - k} \\
&= F - \sum_{j=1}^k F_j \left( \frac{1}{n - j + 1} + \frac{n - k - 1}{n - k} \right) = F_{k+1}.
\end{aligned}$$

This completes the induction.  $\square$

**Lemma 11.** *There exists some integer  $n$  such that the maximum number of bins used by any on-line algorithm  $\mathcal{A}$  is at least  $2.428F$ .*

*Proof.* The adversary takes on a parameter  $n$ . After Stage  $n$ , any on-line algorithm  $\mathcal{A}$  uses at least  $\sum_{i=1}^n F_i$  bins. We compute the ratio  $(\sum_{i=1}^n F_i)/F$  for different values of  $n$ . It is found that the increase in  $n$  leads to an increase in the ratio monotonically. The ratio quickly converges to a value around 2.428. See Figure 2 for the ratio when the value of  $n$  is at most 200. In further computation, we found that when  $n = 12794$ , we have  $\sum_{i=1}^n F_i > 2.428F$ .  $\square$

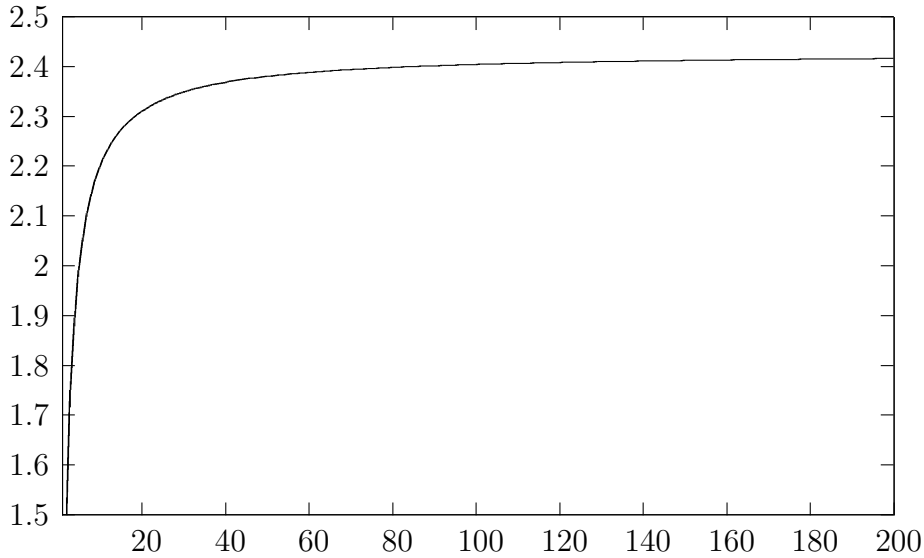


Figure 2: The ratio between  $\sum_{i=1}^n F_i$  and  $F$  when the value of  $n$  increases

**Lemma 12.** *The optimal off-line algorithm uses at most  $F$  bins at any time.*

*Proof.* Denote an item which remains after Stage  $n$  as a permanent item. We can see that there are  $F_i$  permanent items released in Stage  $i$  of size  $1/(n - i + 1)$  for  $1 \leq i \leq n$ . The optimal off-line algorithm can pack the  $F_i$  permanent items to exactly  $F_i/(n - i + 1)$  bins and no empty space remains in each of the bins because  $F_i$  is an integer multiple of  $(n - i + 1)!(n - i)!$ . Since the adversary maintains at any time the total item size is at most  $F$ , the optimal off-line algorithm uses at most  $F$  bins at any time.  $\square$

By Lemmas 11 and 12, the following theorem holds.

**Theorem 13.** *Any on-line algorithm is at least 2.428-competitive.*

## 5 Concluding Remarks

In this paper we have analyzed the performance of the family of any-fit algorithms, including first-fit, best-fit and worst-fit, on the dynamic bin packing problem with unit fractions items. We find that all the any-fit algorithms are at most 3-competitive. In further analysis, we show that first-fit can perform better and its competitive ratio lies between 2.45 and 2.4942, while the competitive ratio for best-fit and worst-fit are tight. We also prove that no on-line algorithm can be better than 2.428-competitive on dynamic bin packing of unit fractions items. In fact, this lower bound improves the lower bound of 2.388 by Coffman et al. [9] on dynamic bin packing of general items. Note that recently, the lower bound for packing general items has further been improved to 2.5 [5].

An immediate open question for the dynamic bin packing of unit fractions items is whether we can close the gap between the 2.4942 upper bound and the 2.428 lower bound. Recall that in this paper we assume items cannot be repacked. A further work is to consider when repacking of items is allowed. Another direction is to consider resource augmentation in which the on-line algorithm can use bins of larger size than the optimal off-line algorithm. To our best knowledge, the study of dynamic bin packing problem has been bounded to the on-line version only. No prior work has been done for the off-line version of the problem. Note that the off-line dynamic bin packing problem is NP-hard as it is a general case of the off-line bin packing problem. It is interesting to see if there are constant approximation algorithms for the problem.

## 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 *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 224–233, 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 *Proc. 15th Annual International Symposium on Algorithms and Computation (ISAAC)*, pages 259–270, 2004.
- [5] W.-T. Chan, P. W. H. Wong, and F. C. C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. In *Proc. 12th Annual International Computing and Combinatorics Conference (COCOON)*, pages 309–319, 2006.
- [6] 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.
- [7] 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.
- [8] E. G. Coffman, Jr., M. Garey, and D. Johnson. Bin packing with divisible item sizes. *J. Complexity*, 3:405–428, 1987.
- [9] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.
- [10] 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 Publishing, 1996.
- [11] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *On-line Algorithms—The State of the Art*, LNCS 1442, pages 147–177, 1996.
- [12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [13] Z. Ivkovic and E. L. Lloyd. Fully dynamic algorithms for bin packing: Being (mostly) myopic helps. *SIAM J. Comput.*, 28(2):574–611, 1998.
- [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: Sets of Inequalities for the Cases Considered in Section 3.1

For  $k = 1$ :

$$\ell_1 \leq b_1 - 1$$

For  $k = 2, r_1 = 1$ :

$$\begin{aligned}\ell_1 &\geq b_1 - \left(1 - \frac{1}{r_2}\right)b_2 \\ \ell_2 &\geq \left(1 - \frac{1}{r_2}\right)b_2 - \left(1 - \frac{2}{r_2}\right)\end{aligned}$$

For  $k = 3, r_1 = 1, r_2 = 2$ :

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \left(\frac{1}{2} - \frac{1}{r_3}\right)b_3 \\ \ell_2 &\geq b_2 - \frac{1}{2}b_3 - \frac{1}{2} \\ \ell_3 &\geq \left(1 - \frac{1}{r_3}\right)b_3 - \left(1 - \frac{2}{r_3}\right)\end{aligned}$$

**For  $k = 4, r_1 = 1, r_2 = 2, r_3 = 3$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \left(\frac{1}{3} - \frac{1}{r_4}\right)b_4 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\ \ell_4 &\geq \left(1 - \frac{1}{r_4}\right)b_4 - \left(1 - \frac{2}{r_4}\right)\end{aligned}$$

**For  $k = 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \left(\frac{1}{4} - \frac{1}{r_5}\right)b_5 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{12}b_5 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{12}b_5 - \frac{1}{2} \\ \ell_4 &\geq \frac{5}{6}b_4 - \frac{1}{12}b_5 - \frac{7}{12} \\ \ell_5 &\geq \left(1 - \frac{1}{r_5}\right)b_5 - \left(1 - \frac{2}{r_5}\right)\end{aligned}$$

**For  $k = 6, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 = 5$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \left(\frac{1}{5} - \frac{1}{r_6}\right)b_6 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_4 &\geq \frac{5}{6}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{7}{12} \\ \ell_5 &\geq \frac{5}{6}b_5 - \frac{1}{30}b_6 - \frac{19}{30} \\ \ell_6 &\geq \left(1 - \frac{1}{r_6}\right)b_6 - \left(1 - \frac{2}{r_6}\right)\end{aligned}$$

**For  $k = 7, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5, r_6 = 6$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \left(\frac{1}{6} - \frac{1}{r_7}\right)b_7 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_4 &\geq \frac{5}{6}b_4 - \frac{1}{20}b_5 - \frac{1}{60}b_6 - \frac{1}{60}b_7 - \frac{7}{12} \\ \ell_5 &\geq \frac{5}{6}b_5 - \frac{1}{60}b_6 - \frac{1}{60}b_7 - \frac{19}{30} \\ \ell_6 &\geq \frac{17}{20}b_6 - \frac{1}{60}b_7 - \frac{41}{60} \\ \ell_7 &\geq \left(1 - \frac{1}{r_7}\right)b_7 - \left(1 - \frac{2}{r_7}\right)\end{aligned}$$

**For  $k \geq 2, r_1 \geq 2$ :**

$$\ell_1 \geq \left(1 - \frac{1}{r_1}\right)b_1 - \left(1 - \frac{2}{r_2}\right)$$

**For  $k \geq 3, r_1 = 1, r_2 \geq 3$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \left(1 - \frac{1}{r_2}\right)b_2 - \frac{1}{r_2}b_3 \\ \ell_2 &\geq \left(1 - \frac{1}{r_2}\right)b_2 - \left(1 - \frac{2}{r_2}\right) \\ \ell_3 &\geq \frac{3}{4}b_3 - \frac{3}{4}\end{aligned}$$

**For  $k \geq 4, r_1 = 1, r_2 = 2, r_3 \geq 4$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \left(\frac{1}{2} - \frac{1}{r_3}\right)b_3 - \frac{1}{r_3}b_4 \\ \ell_2 &\geq b_2 - \frac{1}{2}b_3 - \frac{1}{2} \\ \ell_3 &\geq \left(1 - \frac{1}{r_3}\right)b_3 - \left(1 - \frac{2}{r_3}\right) \\ \ell_4 &\geq \frac{4}{5}b_4 - \frac{4}{5}\end{aligned}$$

**For  $k \geq 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 \geq 5$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \left(\frac{1}{3} - \frac{1}{r_4}\right)b_4 - \frac{1}{r_4}b_5 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{6}b_4 - \frac{1}{2} \\ \ell_4 &\geq \left(1 - \frac{1}{r_4}\right)b_4 - \left(1 - \frac{2}{r_4}\right) \\ \ell_5 &\geq \frac{5}{6}b_5 - \frac{5}{6}\end{aligned}$$

**For  $k \geq 6, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5 \geq 6$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \left(\frac{1}{4} - \frac{1}{r_5}\right)b_5 - \frac{1}{r_5}b_6 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{12}b_5 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{12}b_5 - \frac{1}{2} \\ \ell_4 &\geq \frac{5}{6}b_4 - \frac{1}{12}b_5 - \frac{7}{12} \\ \ell_5 &\geq \left(1 - \frac{1}{r_5}\right)b_5 - \left(1 - \frac{2}{r_5}\right) \\ \ell_6 &\geq \frac{6}{7}b_6 - \frac{6}{7}\end{aligned}$$

**For  $k \geq 7, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5, r_6 \geq 7$ :**

$$\begin{aligned}\ell_1 &\geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \left(\frac{1}{5} - \frac{1}{r_6}\right)b_6 - \frac{1}{r_6}b_7 \\ \ell_2 &\geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_3 &\geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{1}{2} \\ \ell_4 &\geq \frac{5}{6}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \frac{7}{12} \\ \ell_5 &\geq \frac{5}{6}b_5 - \frac{1}{30}b_6 - \frac{19}{30} \\ \ell_6 &\geq \left(1 - \frac{1}{r_6}\right)b_6 - \left(1 - \frac{2}{r_6}\right) \\ \ell_7 &\geq \frac{7}{8}b_7 - \frac{7}{8}\end{aligned}$$

**For**  $k \geq 8, r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4, r_5, r_6 = 6$ :

$$l_1 \geq b_1 - \frac{1}{2}b_2 - \frac{1}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_6 - \left(\frac{1}{6} - \frac{1}{r_7}\right)b_7 - \frac{1}{r_7}b_8$$

$$l_2 \geq b_2 - \frac{1}{3}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_7 - \frac{1}{2}$$

$$l_3 \geq \frac{5}{6}b_3 - \frac{1}{12}b_4 - \frac{1}{20}b_5 - \frac{1}{30}b_7 - \frac{1}{2}$$

$$l_4 \geq \frac{5}{6}b_4 - \frac{1}{20}b_5 - \frac{1}{60}b_6 - \frac{1}{60}b_7 - \frac{7}{12}$$

$$l_5 \geq \frac{5}{6}b_5 - \frac{1}{60}b_6 - \frac{1}{60}b_6 - \frac{19}{30}$$

$$l_6 \geq \frac{17}{20}b_6 - \frac{1}{60}b_7 - \frac{41}{60}$$

$$l_7 \geq \left(1 - \frac{1}{r_7}\right)b_7 - \left(1 - \frac{2}{r_7}\right)$$

$$l_8 \geq \frac{7}{8}b_8 - \frac{7}{8}$$