Dynamic Bin Packing of Unit Fractions Items [*]

Joseph Wun-Tat Chan[†] Tak-Wah Lam[‡] Prudence W.H. Wong[§]

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 ever 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' attention (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 a minimum number of unit-size 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 ever 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 "static" bin packing in the sense that items do 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 a bin,

[*]A preliminary version of this paper appeared in "The 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)"

[†]Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK, joseph.chan@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.

[‡]Department of Computer Science, University of Hong Kong, Hong Kong, twlam@cs.hku.hk.

[§]Department of Computer Science, University of Liverpool, UK, pwong@liverpool.ac.uk. This research was supported in part by Nuffield Foundation Grant NAL/01004/G.

without the knowledge of subsequent items. Moreover, no migration of items is allowed, i.e., items cannot be moved from one bin to another. The performance of an on-line algorithm is measured by its 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 $\{1/k, 2/k, \cdots, j/k\}$ for some $1 \leq j \leq k$).

Dynamic bin packing is a generalization of the classical bin packing problem introduced by Coffman et al. [9]. This generalization assumes that items may depart at arbitrary time. The objective is to minimize the maximum number of bins ever 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. This lower bound was recently improved to 2.5 [5]. Ivkovic and Lloyd [13] studied an even more general problem called the *fully dynamic bin packing problem*, where migration of items is 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. By contrast to the lower bound 2.5 [5] for packing general items, it can be said that packing unit fractions items is indeed "easier". In addition, we prove that no on-line algorithm can be better than 2.428-competitive for packing unit fractions 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 does 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

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

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 items 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 ever used over all time.

As with previous work, we measure the performance of an on-line algorithm by its 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.

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 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, respectively, 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 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, require 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 $\{1, 1/2, \ldots, 1/x\}$. We define a notion of the minimum load of such a bin that an additional item of size $1/y$ cannot fit into the bin. This minimum load is denoted precisely by a function,

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

$\alpha\langle x,y\rangle$	$y=1$	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 =1/2+1/3	5/6	5/6	1
4	1/4	7/12 =1/3+1/4	3/4	5/6	5/6	11/12 =2/3+1/4
5	1/5	8/15 =1/3+1/5	7/10 =1/2+1/5	47/60 =1/3+1/4+1/5	5/6	17/20 =1/4+3/5
6	1/6	8/15	7/10	23/30 =3/5+1/6	49/60 =1/4+2/5+1/6	17/20

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

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 $\alpha\langle x,y\rangle$:

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

The values of $\alpha\langle x,y\rangle$ for $1 \le x, y \le 6$ are given in Table 2 (which are computed by brute force exhaustion); some of these values are required for subsequent analysis.

With respect to an input sequence 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 bin b_1 is of size $1/r_1$. We define b_i and r_i iteratively for $i \ge 2$ 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 $b_1 > b_2 > \ldots > b_k$ and $r_1 < r_2 < \ldots < r_k$.

Now we are ready to describe the general case and the lower bound on the number of bins ever used by the optimal algorithm in the general case. The lower bound is in fact the maximum among the total loads of all bins at the time instances, t_i for $1 \le i \le k$, denoting the time when FF packs an item of size $1/r_i$ into bin b_i. 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 or 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 occupied bins at time t_k is at least

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

Next, for any integer $1 \le i \le k-1$, consider a 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 $i+1 \le p < k$, 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 $\ell = \max_{1\le i\le k}\ell_i$. 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. By the case analysis given later, we prove that $b_1 \le 2.4942\ell + 1$, which implies that first-fit is 2.4942-competitive.

The case analysis studies all possible values of k and r_i for $1 \le i \le k$, which are divided 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 $\{k \ge 2\}$. Then the case of $\{k \ge 2\}$ can be further divided into three subcases $\{k = 2, r_1 = 1\}$, $\{k \ge 2, r_1 \ge 2\}$, and $\{k \ge 3, r_1 = 1\}$. Similarly, the case of $\{k \ge 3, r_1 = 1\}$ can be further divided into three subcases $\{k = 3, r_1 = 1, r_2 = 2\}$, $\{k \ge 3, r_1 = 1, r_2 \ge 3\}$, and $\{k \ge 4, r_1 = 1, r_2 = 2\}$. Repeating this method of case division, we can always partition, for any integer $c \ge 1$, all cases into three types of cases.

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

Type (2): $\{k \ge i, r_1 = 1, r_2 = 2, \ldots, r_{i-2} = i-2, r_{i-1} \ge i\}$ for $2 \le i \le c$.

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

Note that there is only one case instance for the Type (3). We analyze for each case the corresponding relation between b_1 and ℓ. Precisely, in each case we obtain an inequality in the form of $b_1 \le x_1\ell + x_2$ 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 $c = 1, 2, \ldots$, 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 $b_1 \le x_1\ell + x_2$ for the cases in Table 3. We pick the Type (2) case $k \ge 5, r_1 = 1, r_2 = 2, r_3 = 3, r_4 \ge 5$ 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 ℓ_i for $1 \le i \le 5$, as follows.

$$\begin{aligned}
\ell_1 &= \tfrac{1}{r_1} + (b_1 - b_2 - 1)\alpha\langle r_1, r_1\rangle + \textstyle\sum_{p=2}^{k-1}(b_p - b_{p+1})\alpha\langle r_p, r_1\rangle + b_k\alpha\langle r_k, r_1\rangle \\
&\geq 1 + (b_1 - b_2 - 1)\alpha\langle 1, 1\rangle + (b_2 - b_3)\alpha\langle 2, 1\rangle + (b_3 - b_4)\alpha\langle 3, 1\rangle + (b_4 - b_5)\alpha\langle r_4, 1\rangle \\
&\geq 1 + (b_1 - b_2 - 1) + \tfrac{1}{2}(b_2 - b_3) + \tfrac{1}{3}(b_3 - b_4) + \tfrac{1}{r_4}(b_4 - b_5) \\
&\geq b_1 - \tfrac{1}{2}b_2 - \tfrac{1}{6}b_3 - (\tfrac{1}{3} - \tfrac{1}{r_4})b_4 - \tfrac{1}{r_4}b_5 \qquad (4) \\
\ell_2 &= \tfrac{1}{r_2} + (b_2 - b_3 - 1)\alpha\langle r_2, r_2\rangle + \textstyle\sum_{p=3}^{k-1}(b_p - b_{p+1})\alpha\langle r_p, r_2\rangle + b_k\alpha\langle r_k, r_2\rangle \\
&= \tfrac{1}{2} + (b_2 - b_3 - 1)\alpha\langle 2, 2\rangle + (b_3 - b_4)\alpha\langle 3, 2\rangle + \textstyle\sum_{p=4}^{k-1}(b_p - b_{p+1})\alpha\langle r_p, 2\rangle + b_k\alpha\langle r_k, 2\rangle \\
&\geq \tfrac{1}{2} + (b_2 - b_3 - 1) + \tfrac{2}{3}(b_3 - b_4) + \tfrac{1}{2}b_4 \\
&\geq b_2 - \tfrac{1}{3}b_3 - \tfrac{1}{6}b_4 - \tfrac{1}{2} \qquad (5) \\
\ell_3 &= \tfrac{1}{r_3} + (b_3 - b_4 - 1)\alpha\langle r_3, r_3\rangle + \textstyle\sum_{p=4}^{k-1}(b_p - b_{p+1})\alpha\langle r_p, r_3\rangle + b_k\alpha\langle r_k, r_3\rangle \\
&\geq \tfrac{1}{3} + \tfrac{5}{6}(b_3 - b_4 - 1) + \tfrac{2}{3}b_4 \\
&\geq \tfrac{5}{6}b_3 - \tfrac{1}{6}b_4 - \tfrac{1}{2} \qquad (6) \\
\ell_4 &= \tfrac{1}{r_4} + (b_4 - b_5 - 1)\alpha\langle r_4, r_4\rangle + \textstyle\sum_{p=5}^{k-1}(b_p - b_{p+1})\alpha\langle r_p, r_4\rangle + b_k\alpha\langle r_k, r_4\rangle \\
&\geq \tfrac{1}{r_4} + (b_4 - 1)(1 - \tfrac{1}{r_4}) \\
&\geq (1 - \tfrac{1}{r_4})b_4 - (1 - \tfrac{2}{r_4}) \qquad (7)
\end{aligned}$$

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

$$\ell_5 = \tfrac{1}{r_5} + (b_5 - 1)\alpha\langle r_5, r_5\rangle \geq (b_5 - 1)(1 - \tfrac{1}{r_5}) \geq \tfrac{5}{6}b_5 - \tfrac{5}{6} \qquad (8)$$

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

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

Since $\ell \geq \max_{1\leq i\leq 5}\{\ell_i\}$, we can substitute ℓ for ℓ_i in the above five inequalities. The aim is to solve the system of inequalities and get a bound on b_1 with a function of ℓ. This can be done by using the bound on b_4 from Equation (7) to get the bound on b_3 from Equation (6). Then, we can use the bounds on b_4 and b_3 to get the bounds on b_2 from Equation (5), and hence the bounds on b_1 from Equation (4) using also the bound on b_5 from Equation (8). As a result, 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 $r_4 \geq 5$.

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 the lower bound of FF by constructing an adversary sequence so that the maximum number of bins ever used by FF is at least 2.45 times that used by the optimal

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. The maximum among the x_1's is 2.4942.

off-line algorithm. First, we introduce a notation which is used in describing the adversary. Consider two positive integers x and y and a bin that contains only items of size $1/x$. Let $\beta\langle x, y\rangle$ denote the minimum number of items the bin must contain so that an additional item of size $1/y$ cannot fit 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 $\beta\langle x, y\rangle = 3$.

The adversary consists of n stages. Generally speaker, the adversary is characterized by a sequence of integers $k_1, k_2, \ldots, k_n$ in descending order, i.e., $k_i > k_{i+1}$ for $1 \leq i \leq n-1$. (The values of the integers will be determined later in the analysis.) In Stage i, for each $1 \leq i \leq n$, the adversary releases a number of items of size $1/k_i$ for some integer k_i and then let some of them depart. In particular, in Stage 1, D_1k_1 items of size $1/k_1$ are released, for some large integer D_1. FF packs all D_1k_1 items into D_1 bins, and each bin is fully packed.

At the beginning of each subsequent stage, i.e., Stage i, for $2 \leq i \leq n$, the adversary targets to force the following invariant on the packing that FF creates.

- The total size of the items in all bins is D_1;
- Each occupied bin contains only items of the same size; and
- A bin that contains items of size $1/x$ contains $\beta\langle x, k_{i-1}\rangle$ such items.

The invariant holds at the beginning of Stage 2 since each of the D_1 occupied bins contains $\beta\langle k_1, k_1\rangle = k_1$ items of size $1/k_1$.

Stage i consists of the following two steps.

1. For each occupied bin containing items of size $1/x$, $\beta\langle x, k_{i-1}\rangle - \beta\langle x, k_i\rangle$ items depart, i.e., $\beta\langle x, k_i\rangle$ items remained.

2. Let D_i be the total size of all the departed items in Step 1 of this stage. (With a sufficiently large D_1, D_i is an integer, as proved in Lemma 3). $D_i k_i$ items of size $1/k_i$ are released. Since each bin with items of size $1/x$ contains $\beta\langle x, k_i\rangle$ such items, none of the newly released items can be packed into any occupied bin. Therefore, FF uses D_i new bins to pack these items, where each new bin contains $k_i = \beta\langle k_i, k_i\rangle$ items. Thus, the invariant also holds at the beginning of Stage $i + 1$.

We analyze the performance of FF on this input sequence. By the adversary, FF uses $\sum_{i=1}^{n} D_i$ bins after Stage n. On the other hand, the optimal off-line algorithm uses D_1 bins over all time (Lemma 4). The competitive ratio of FF is at least $\sum_{i=1}^{n} D_i/D_1$. By the adversary, D_i for $2 \le i \le n$ can be computed as

$$D_i = \sum_{j=1}^{i-1} \left\{ \frac{D_j}{k_j} (\beta\langle k_j, k_{i-1}\rangle - \beta\langle k_j, k_i\rangle) \right\}.$$

Note that the competitive ratio is independent of the value of D_1. Let $r_i = D_i/D_1$. We have

$$r_i = \sum_{j=1}^{i-1} \left\{ \frac{r_j}{k_j} (\beta\langle k_j, k_{i-1}\rangle - \beta\langle k_j, k_i\rangle) \right\}$$

and the competitive ratio is equal to $\sum_{i=1}^{n} r_i$, in which its value depends on the values of $k_1, k_2, \ldots, k_n$.

The following lemma shows that there is an actual instance of the adversary such that the competitive ratio of FF, i.e., $\sum_{i=1}^{n} r_i$, is at least 2.45. Therefore, FF is at least 2.45-competitive. (Theorem 5).

Lemma 2. *There exits a sequence of n integers $k_1, k_2, \ldots, k_n$ in descending order such that $\sum_{i=1}^{n} r_i > 2.45$.*

Proof. If we let $n = 16$ and the values $100000, 97, 37, 23,\ 19,\ 13, 11, 10, 9, 7, 6, 5, 4, 3, 2, 1$ for $k_1, \ldots, k_{16}$, respectively, we have $\sum_{i=1}^{n} r_i > 2.45$. The detailed mathematics are omitted. $\square$

Lemma 3. *If $D_1 = \prod_{j=1}^{n} k_j$, for $1 \le i \le n$, D_i is an integer, and in particular, D_i is a multiple of k_i.*

Proof. We prove by induction on i that D_i is a multiple of $\prod_{x=i}^{n} k_x$, which is a multiple of k_i. By the adversary, $D_i = \sum_{j=1}^{i-1} (D_j/k_j)(\beta\langle k_j, k_{i-1}\rangle - \beta\langle k_j, k_i\rangle)$. Since the function β gives an integer output and D_j/k_j for $1 \le j \le i-1$ is an integer multiple of $\prod_{x=j+1}^{n} k_x$, the summation gives an integer multiple of $\prod_{x=i}^{n} k_x$. $\square$

Lemma 4. *The optimal off-line algorithm uses D_1 bins over all time.*

Proof. We give an algorithm $\mathcal{O}$ that packs the items in the adversary using D_1 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 some stage. 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 solely temporary items that will depart in the same stage. What we need to prove is that, for all $1 \leq i < j \leq n$, the total size of the temporary items of size $1/k_i$ that depart in Stage j is an integer. In that case, we can guarantee that in each stage algorithm $\mathcal{O}$ can fully pack each bin so that in total exactly D_1 bins are used. By the adversary, the total size of the temporary items of size $1/k_i$ that depart in Stage j is $(D_i/k_i)(\beta\langle k_i, k_{j-1}\rangle - \beta\langle k_i, k_j\rangle)$, which is an integer because by Lemma 3 D_i/k_i is an integer. Therefore, algorithm $\mathcal{O}$, and hence the optimal off-line algorithm, uses D_1 bins over all time. □

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 σ. 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 an 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 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. □

We give an idea on how an adversary for WF can be constructed. The adversary can force WF, at some stage, to use n bins each with only one very small item for any integer $n > 0$. This is done by alternating releasing item of size $1/2$ and a very small item. All items of size $1/2$ then depart leaving one very small item in each bin. Finally, $n/2$ items of size 1 are released and none of them can be packed into existing bins and thus require $n/2$ new bin. Therefore, WF uses $3n/2$ bins. On the other hand, the optimal off-line algorithm uses only $n/2 + 1$ where one bin is used for packing all very small items and $n/2$ bins for all items of sizes $1/2$ and 1. This can be done because all the $n/2$ items of size 1 are released after all the n items of size $1/2$ depart. Hence we have the following theorem.

Theorem 7. *Worst-fit is no better than* 3*-competitive.*

Next, we give an adversary for BF. This adversary has a similar target as the one for WF: to force BF to use n bins each with only one very small item. Due to the difference between BF and WF, it takes more careful design of item release and departure to achieve the same aim. We give an inductive argument on how this can be done. Suppose there are already k bins each with only one very small item. $k+1$ items of size $1/2$ are released where k of them are packed into the k occupied bins but the remaining one must be packed into a new bin. Then the first k items of size $1/2$ depart leaving the bin with an item of size $1/2$ the heaviest bin. A very small item is then released and BF will pack the item into the bin with an item of size $1/2$. When this item of size $1/2$ departs, we have $k + 1$ bins each with only one very small item. Similar to the case of WF, we can also force BF to use $3n/2$ bins, while the optimal off-line algorithm uses only $n/2 + 1$ bins. Again, the optimal off-line algorithm uses one bin for packing all very small items and $n/2$ bins for all items of sizes $1/2$ and 1. It is clear that at any time instance there are at most n items of size $1/2$ or at most $n/2$ items of size 1. Similar to the adversary for WF, all the $n/2$ items of size 1 are released after all the items of size $1/2$ depart. Therefore, the optimal off-line algorithm uses only $n/2+1$ bins. The following theorem then follows.

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

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. The adversary has the same framework as the adversary presented in Section 3.2. It consists of n stages and is characterized by a sequence of integers $k_1, k_2, \ldots, k_n$ in descending order. The integer k_i represents the size of the items released in the i-th stage. In Stage 1, F_1k_1 items of size $1/k_1$ are released, for some large integer F_1. Any algorithm $\mathcal{A}$ uses at least F_1 bins to pack the F_1k_1 items. If more than F_1 bins are used, except the first F_1 bins, all other bins are made empty with all their items departed. Each of the subsequent stages, i.e., Stage i, for $2 \leq i \leq n$, consists of the following three steps.

1. For each occupied bin, an item with the smallest size remains and all other items depart.

2. Let R_i be the total size of all the items that remain. The adversary releases $(F_1 - R_i)k_i$ items of size $1/k_i$. (With a sufficiently large F_1, $F_1 - R_i$ is an integer, as shown in Lemma 10.) Note that at this point the total size of the items in all bins becomes F_1 again.

3. Suppose that in Stage j for $1 \le j < i$, F_j new bins are used. Since $1/k_j < 1/k_i$, each of these bins (with a single item of size $1/k_j$) can accommodate at most $k_i - 1$ items of size $1/k_i$. The number of new bins used in this stage is at least

$$F_i = F_1 - \sum_{j=1}^{i-1} F_j \left(\frac{1}{k_j} + \frac{k_i - 1}{k_i} \right).$$

   If more than F_i new bins are used, except the first F_i new bins, all other new bins are made empty with all their items departed.

We analyze the performance of any algorithm on this input sequence. In the n stages, any algorithm uses at least $\sum_{i=1}^{n} F_i$ bins. On the other hand, the optimal off-line algorithm uses F_1 bins (Lemma 11). The competitive ratio of any algorithm is at least $\sum_{i=1}^{n} F_i/F_1$, where $F_i = F_1 - \sum_{j=1}^{i-1} F_j(1/k_j + (k_i - 1)/k_i)$. Note that the competitive ratio is independent of the value of F_1. Let $s_i = F_i/F_1$. We have

$$s_i = 1 - \sum_{j=1}^{i-1} s_j \left(\frac{1}{k_j} + \frac{k_i - 1}{k_i} \right)$$

and the competitive ratio is equal to $\sum_{i=1}^{n} s_i$, in which its value depends on the values of $k_1, k_2, \ldots, k_n$.

The following lemma shows that there is an actual instance of the adversary such that the competitive ratio of any algorithm, i.e., $\sum_{i=1}^{n} s_i$, is at least 2.428. Therefore, the competitive ratio for any algorithm is at least 2.428 (Theorem 12).

Lemma 9. *There exists a sequence of n integers $k_1, k_2, \ldots, k_n$ in descending order such that $\sum_{i=1}^{n} s_i > 2.428$.*

Proof. If we let $n = 81$ and the value 1000000 for k_1 and the values $80, 79, \ldots, 1$ for $k_2, k_3, \ldots, k_{81}$, respectively, we have $\sum_{i=1}^{n} s_i > 2.428$. The detailed mathematics are omitted. □

Lemma 10. *If $F_1 = k_1 \prod_{j=2}^{n} k_j^2$, for $1 \le i \le n$, F_i and R_i are integers, and in particular, F_i is a multiple of k_i.*

Proof. We prove by induction on i that F_i is a multiple of $k_i \prod_{j=i+1}^{n} k_j^2$, which is a multiple of k_i. By the adversary, $F_i = F_1 - \sum_{j=1}^{i-1} F_j(1/k_j + (k_i - 1)/k_i)$. As F_j is a multiple of $k_j \prod_{x=j+1}^{n} k_x^2$ for $1 \le j \le i-1$, and hence $F_j/(k_j k_i)$ is a multiple of $k_i \prod_{x=i+1}^{n} k_x^2$, F_i is a multiple of $k_i \prod_{x=i+1}^{n} k_x^2$. By the adversary, $R_i = \sum_{j=1}^{i-1} F_j/k_j$. Since F_j is a multiple of k_j, R_i is an integer. □

Lemma 11. *The optimal off-line algorithm uses F_1 bins over all time.*

Proof. Denote an item that remains after Stage n as a permanent item. We can see that F_i permanent items of size $1/k_i$ are released in Stage i for each $1 \leq i \leq n$. The optimal off-line algorithm can pack the F_i permanent items to F_i/k_i bins and the other $(F_1 - R_i)k_i - F_i$ items to $(F_1 - R_i) - F_i/k_i$ bins and every bin is fully packed as F_i is a multiple of k_i by Lemma 10. Therefore, the optimal off-line algorithm can fully pack each bin in each stage and hence it uses F_1 bins over all time. □

Theorem 12. *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.

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. While the family of any-fit algorithms give constant approximation ratio for the problem, it is interesting to see if there are approximation algorithms with better approximation ratios.

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. *ACM Transactions on Algorithms*, 3(3):28, 2007.

[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. *Algorithmica* (To appear). http://dx.doi.org/10.1007/s00453-008-9185-z.

[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 pakcing 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.