1 Introduction

Bin packing is one of the fundamental problems in theory of computer science. Analysis of bin packing algorithms has has contributed a lot to theories behind approximation algorithms (in the offline setting) and online algorithms. The first ‘competitive’ results for classic bin packing algorithms appeared a decade before the term competitive analysis was introduced in the context of paging and list update. We start this section with a review of classic results for bin packing and later we explore some recent results with respect to application of bin packing in the cloud as well as advice complexity of bin packing problem.

1.1 Offline vs online bin packing

Bin packing asks for packing of a collection of input items into a minimum number of bins of uniform capacity. Items can represent files that should be stored in servers, merchandise that should be packed into trucks (with a given weight capacity) in order to be sent from one location to another, or cakes students who request different portions of cakes. In these examples, servers, trucks, and cakes represent bins of uniform sizes that should be shared between items of different sizes (files of different volume, merchandise of different weights, and cake-pieces of different sizes). The size of an item is assumed to be smaller than the capacity of bins. We often assume bins have capacity 1 and items have sizes in $(0, 1]$. Our understanding of an ‘item’ is in the bin packing problem is solely based on its size; hence, we often use an item and its size interchangeably.

In the offline setting, items form a ‘multiset’ and an algorithm can process this multiset before placing any item into a bin, e.g., it can sort items in decreasing order (which is quite useful in practice). It is not hard to see the problem is NP-hard:

**Theorem 1.** Offline bin packing problem is NP-hard.

*Proof.* We use a reduction from 3-partition problem, which is an NP-hard problem. An instance of 3-partition is formed by a multiset $S$ of positive integers, and the goal is to decide whether it is possible

---

1 Multiset is an extension of a set where a given member can appear more than once.
to partition $S$ into two subsets of two subsets $S_1$ and $S_2$ s.t. sum of the numbers in $S_1$ equals to the sum of the numbers in $S_2$. For example, the answer to the instance $S = \{3, 1, 3, 2, 3, 2, 3, 4, 1\}$ is ‘yes’ because $S$ can be partitioned to $S_1 = \{3, 2, 3\}$ and $S_2 = \{1, 3, 4, 1\}$ whose total sum is equal to 11.

Given an instance of partition problem, defined by multiset $S$, we form an instance of bin packing in polynomial time. Let $X$ denote the total sum of items in $S$. We create an instance of bin packing defined by multiset $S'$ which is the same as $S$ with its items divided by $X/2$. In the above example, we have $X = 22$, and the bin packing multiset is $S' = \{3/11, 1/11, 3/11, 2/11, 3/11, 2/11, 3/11, 4/11, 1/11\}$. It is not hard to see there is a packing of $S'$ into two bins if and only if the answer to the partition instance is ‘yes’. Figure 1 provides an illustration.

The above theorem not only shows that bin packing is NP-hard, but it also shows that it is hard to find a packing that opens less than 1.5 times the number of bins used by $\text{Opt}$. However, this is only the case when $\text{Opt}$ opens 2 bins; in fact there are polynomial algorithms that open $1(1+\epsilon)\text{Opt}+1$ bins for any input sequence. That implies that, we can get arbitrary close to $\text{Opt}$ for sequences for which the number of bins opened by $\text{Opt}$ goes to infinity. We call the maximum ratio between the cost of an offline algorithm and that of $\text{Opt}$ for sequences where the cost of $\text{Opt}$ goes to infinity as the asymptotic approximation ratio of the offline algorithm. So, there are algorithms with asymptotic approximation ratio of $1 + \epsilon$. There are more straightforward offline algorithms with better ratios. For example, a simple algorithm which sorts items in decreasing order of their sizes and then places them one by one using First-Fit rule (which will be described later) has an asymptotic approximation ratio of 11/9. As another remark, if there is a constant number of different item-sizes, there is a polynomial algorithm which solves the problem optimally.

2 Online Algorithms of Bin Packing

In online bin packing, the input is a sequence of items, revealed one by one. An online algorithm places each item into a bin without a priori knowledge about future items. The decisions of the algorithm are irrevocable, i.e., after placing an item into a bin, the algorithm cannot change the bin associated with the item.

Next Fit is a simple bin packing algorithm which maintains one bin open at each time. If an incoming item fits in the open bin, the algorithm places it there. Otherwise, it closes the bin and opens a new bin. A closed bin is never referred to again (i.e., no future item is placed in a closed
Figure 2: Packing of Next Fit for sequence \( \sigma = (0.9, 0.3, 0.8, 0.5, 0.1, 0.1, 0.3, 0.2, 0.4, 0.2, 0.4, 0.5, 0.5, 0.8, 0.6, 0.4, 0.5) \).

First Fit is a more practical algorithm which never closes a bin. It keeps all bins opened in the order in which they are opened and places an input item in the first bin which has enough space. If none of the existing items has enough space, First Fit opens a new bin for the item. Best Fit is a similar algorithm that places an item in the bin that best ‘fits’ it, i.e., the fullest bin which has enough space. We define the level of a bin as the total size of items placed in the bin. So, Best Fit places an item in the bin with maximum level which has enough space. Figure 3 show the packings of First Fit and Best Fit for the same sequence. Note that in this case, Best Fit has a lower cost; however, for some sequences it is First Fit that opens less bins.

First Fit and Best Fit are members of Any-Fit family of algorithms. An algorithm belongs to this family if it avoids opening a new bin if any of the existing bins has enough space for the item. Note that Next Fit is not an Any-Fit algorithm since an item can fit in the closed bin but Next Fit does not use it. Worst-Fit is an Any-Fit algorithm if it places an item in the bin with the smallest level (the ‘emptiest bin’). An online algorithm belongs to Almost-Any-Fit family of algorithms if it is Any-Fit and it avoid worst-fit strategy. It means that, if there are more than one bin that have enough space for an incoming item, the algorithm uses places the item in any bin except for the emptiest bin. Clearly, Worst-Fit is not an Almost-Any-Fit algorithm. Similarly, Fist Fit is an Any-Fit algorithm which is not Almost-Any-Fit because the first bin which has enough space might be the emptiest bin which fits an incoming item, e.g., for sequence \( (0.6, 0.7, 0.8, 0.2) \), First Fit places the item of size 0.3 in the emptiest bin with level 0.6.

Harmonic family is another family of bin packing algorithms which classify items based on their

Figure 3: Packing of Next Fit for sequence \( \sigma = (0.9, 0.3, 0.8, 0.5, 0.1, 0.1, 0.3, 0.2, 0.4, 0.2, 0.4, 0.5, 0.5, 0.8, 0.6, 0.4, 0.5) \).
sizes. In the simplest form, the classic Harmonic algorithm divides items into $K$ classes, where $K$ is an integer parameter of the algorithm. Items in the range $(1/2, 1]$ belong to class 1, items in the range $(1/3, 1/2]$ belong to class 2, and more generally items in the range $(\frac{1}{i+1}, \frac{1}{i}]$ belong to class $i < K$. Class $K$ is special class formed by items smaller than $1/K$. The algorithm packs items of each class separately from other items using Next Fit strategy. Note that for class $i < K$, exactly $i$ items fit in the same bin. For example, only 1 item larger than $1/2$ (of class 1) fits in a bin and two items in the range $(1/2, 1/3]$ (i.e., class 2) fit in the same bin. Figure 4 show the packing of Harmonic for an input sequence.

2.1 Analysis measures

When analyzing bin packing algorithms, we are often interested in cases where the cost of $\text{OPT}$ is arbitrary large. There are input sequence of large length which can be packed in a small number of bins (e.g., when all items are very small). For these sequences, opening a constant number of extra bins can dramatically change the ratio between the cost of an online algorithm and $\text{OPT}$ (in a similar way that we observed earlier in the context of the partition problem). In summary, to compare quality of different algorithms, we restrict our attention to inputs for which $\text{OPT}$ (and hence any other algorithm) opens an arbitrary large number of bins. We define the asymptotic competitive ratio of an online algorithm as the maximum ratio between the cost of the algorithm and that of $\text{OPT}$ for the same sequence for which the cost of $\text{OPT}$ goes to infinity. This definition of competitive ratio is in contrast with absolute competitive ratio which considers the ratio among all sequences even those which can be packed into a constant number of bins. In the remainder of this section, we use term ‘competitive ratio’ to refer to ‘asymptotic competitive ratio’.

Average-case ratio is another measure that we sometimes use to compare online algorithms. Assuming item-sizes are independently and identically distributed (typically using a Uniform distribution), the average case ratio of an algorithm is the expected value of the ratio between the cost of a sequence and that of $\text{OPT}$. Average-case ratio is a performance measure which is more indicative of the typical performance of bin packing algorithms as we will see later.

2.2 Analysis of Next Fit

Let’s start our quest to analysis of bin packing algorithms with the simplest online algorithm.

Lemma 1. Next Fit has a competitive ratio of at most 2.
Figure 5: Competitive ratio of Next Fit is at least 2.

Proof. Consider the final packing of Next Fit for any sequence. We claim the total size of items in each two consecutive bins is more than one. Consider otherwise, i.e., there are two consecutive bins with total level less than 1. This means that any of the items in the second bin could fit in the first bin. Consequently, Next Fit should have not opened the second bin which is a contradiction. Let $S$ denote the total size of items in the input sequence. Since each two consecutive bins have total size more than 1, we conclude that there cannot be more than $2S$ bins in the Next Fit packing; otherwise, there will be more $S$ pairs of bins each having level more than 1 which given a total size of more than $S$ for the packed items which is a contradiction. On the other hand, $\text{Opt}$ has to open at least $S$ bins. This is because each bin of $\text{Opt}$ includes items with total size at most 1 (capacity of the bin) and hence to pack all items, at least $S$ bins are required. In summary, for any sequence with total size $S$, the cost of Next Fit is at most $2S$ and the cost of $\text{Opt}$ is at least $S$. Consequently, the competitive ratio of the algorithm is at most $\frac{2S}{S} = 2$.

The above lower bound is indeed tight:

**Lemma 2.** Next Fit has a competitive ratio of at least 2.

Proof. Consider sequence $\sigma = \langle 0.5, \epsilon, 0.5, \epsilon, \ldots \rangle$. Next Fit opens a new bin for items of size 0.5 and the next item of size $\epsilon$ results in a bin with level more than 0.5 which cannot be used for the subsequent item of size 0.5 (and hence it will be closed). Assuming $\sigma$ has length $n$, the cost of NextFit for packing $\sigma$ will be roughly $n/2$. On the other hand, $\text{Opt}$ places all items of size $\epsilon$ in one bin and opens one bin for any two item of size 0.5. The cost of $\text{Opt}$ will be roughly $n/4$. Figure 5 provides an illustration. In summary, the ratio between the cost of Next Fit and $\text{Opt}$ for $\sigma$ is at most $\frac{n/2}{n/4} = 2$. Since the competitive ratio is the maximum value of this ratio for all sequences, it will be at least 2.

From the above two lemmas, we conclude the following theorem:

**Theorem 2.** Competitive ratio of Next Fit is exactly 2.

### 2.3 Weighting argument

In order to provide an upper bound for the competitive ratio of most bin packing algorithm such as First Fit and Best Fit, we often use a weighting technique. To prove a competitive ratio of an algorithm $\text{Alg}$ is at most $c$, we take the following steps in a weighting argument:

- **Step I:** Define a weighting function $w$ which assigns a weight $w(x)$ for any item of size $x$. Typically, we have $w(x) \geq x$. 

   [Diagram of Next Fit and Opt with weights]
• Step II: Show that, for any bin in the final packing of \text{Alg} (except possibly a constant number of them), the total weight of items in \text{B} is at least 1.

• Step III: Show that, for any arbitrary bin \text{B} (in particular, a bin in the packing of \text{Opt}), the total weight of items in \text{B} is at most \(c\).

To see why the above steps guarantee a competitive ratio at most \(c\), let \(W\) denote the total weight of all items in an input sequence \(\sigma\). From step II, we know the weight of items in any bin of \text{Alg} (except possibly a constant number of them) is at least 1; hence, the number of such bins cannot be more than \(W\) (otherwise total weight of packed items will be more than \(W\) which is a contradiction). So, we have \(\text{cost(Alg)} \leq W + x\) for some constant \(x\). One the other hand, from step III, weight of any bin in the \text{Opt}'s packing is at most \(c\). In summary, the cost of \text{Alg} for an sequence of total weight \(W\) is at most \(W\) and the cost of \text{Opt} is at least \(W/c\). Hence, the competitive ratio of \text{Alg} will be at most \(W + x\ W/c\) which approaches to \(c\) for large values of \(W\) (i.e., when the cost of \text{Opt} is sufficiently large).

2.3.1 Analysis of harmonic algorithm

We use the weighting technique to prove an upper bound for competitive ratio of Harmonic algorithm. Recall that the algorithm places \(i\) items of class \(i\) (i.e., with size in the range \((1/(i+1), 1/i]\)) in the same bin for \(i < K\). The algorithm uses Next Fit strategy to pack items of class \(K\) (those smaller than \(1/K\)).

**Lemma 3.** The Harmonic algorithm has a competitive ratio of at most \(c = 1 + 1/(1 \times 2) + 1/(2 \times 3) + 1/(6 \times 7) + 1/(42 \times 43) + \text{etc.} \approx 1.691.\)

**Proof.** We take the three steps involved in the weighting argument to prove the competitive ratio is at most \(c\).

• Step I: we define the weight of items in class \(i\) (i.e., with size in the range \((1/(i+1), 1/i]\)) to be \(1/i\) for \(i < K\). For an item of size \(x\) that belongs to class \(K\) (i.e., \(x \leq 1/K\)) the weight is defined as \(K/(K-1) \times x\).

• Step II: we show that total weight of items in any bin of Harmonic, except a constant number of them, is at least 1. Note that all bins of class \(i < K\), except possibly the last bin of the class, include \(i\) items of weight \(1/i\). Hence, these bins have total weight of 1. Now, consider a bin \(B\) of class \(K\). Since all items of class \(K\) are smaller than \(1/K\), if a bin is opened after \(B\), the level of \(B\) has been larger than \((k-1)/k\) (otherwise the items in the next bin should have been placed in \(B\)). Since the weight of items placed in \(B\) is the product of their sizes and \(k/(k-1)\), the total weight of items in \(B\) is at least \((k-1)/k \cdot k/(k-1) = 1\). We conclude that all bins of Harmonic, except possibly the last bin of each class (a total of \(K\) bins) have weight at least 1.

• Step III (sketch): we show that the total weight of items in any bin of \text{Opt} is at most \(c\). Define the density of an item to be the ratio between the weight and the size of the item. For example, density of an item in class 1 is less than 2 because the weight of such item is 1 and its size is more than \(1/2\). Similarly, the density of items in class \(i < K\) is upper-bounded by \((i+1)/i\). For items of class \(K\), the density is \(K/(K-1)\). In order to maximize the total weight of items in a bin of \text{Opt}, one can show that we need to place items in the bin in a greedy manner in a decreasing order of density. In other words, we repeatedly add the item with the largest density which fits into the bin. For large values of \(K\), these items are \(1/2 + \epsilon\) (density
2), \(1/3 + \epsilon\) (density 3/2), \(1/7 + \epsilon\) (density 7/6), \(1/43 + \epsilon\) (density 43/42), and so on. These items respectively have weights \(1, 1/2, 1/6, 1/42\), and so on. So the total weight of items in any bin of \(\text{OPT}\) will be the claimed value of \(c\).

In fact the upper bound in the above theorem is tight:

**Lemma 4.** The competitive ratio of Harmonic algorithm is at least \(c = 1 + 1/(1 \times 2) + 1/(2 \times 3) + 1/(6 \times 7) + 1/(42 \times 43) + etc. \approx 1.691\).

**Proof.** Consider the following sequence:

\[
\begin{align*}
&\left(\frac{1}{43} + \epsilon, \ldots, \frac{1}{43} + \epsilon\right) \quad m \text{ times} \\
&\left(\frac{1}{7} + \epsilon, \ldots, \frac{1}{7} + \epsilon\right) \quad m \text{ times} \\
&\left(\frac{1}{3} + \epsilon, \ldots, \frac{1}{3} + \epsilon\right) \quad m \text{ times} \\
&\left(\frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon\right) \quad m \text{ times}
\end{align*}
\]

Harmonic places items of the same sizes together (they belong to the same class). Hence, it opens \(m(1/42 + 1/6 + 1/2 + 1)\) bins. We can add prefixes to the beginning of the sequence to ensure \(\text{Alg}\) opens \(c \cdot m\) bins. On the other hand, \(\text{OPT}\) places one item of each class in each bin. Hence, the cost of \(\text{OPT}\) is \(m\). Hence, there is a sequence for which the ratio between the costs of Harmonic and \(\text{OPT}\) is \(c\), i.e., the competitive ratio of Harmonic is at least \(c\).

From the above lemma we conclude the following theorem:

**Theorem 3.** The competitive ratio of Harmonic algorithm is exactly \(c = 1 + 1/(1 \times 2) + 1/(2 \times 3) + 1/(6 \times 7) + 1/(42 \times 43) + etc. \approx 1.691\).

### 2.3.2 Analysis of First Fit

Weighting arguments can be used to compute competitive ratio of other bin packing algorithms such as First Fit and Best Fit.

**Lemma 5.** The competitive ratio of First Fit is at most 1.7.

**Proof.** The proof is based on weighting function. Here we just show step I by providing the weighting function. For an item of size \(x\) define weight \(w(x)\) as follows:

\[
w(x) = \begin{cases} 
(6/5)x & \text{for } 0 \leq x \leq 1/6 \\
(9/5)x - 1/10 & \text{for } 1/6 < x \leq 1/3 \\
(6/5)x + 1/10 & \text{for } 1/3 < x \leq 1/2 \\
(6/5)x + 4/10 & \text{for } 1/2 < x \leq 1
\end{cases}
\]

With the above definition of weighting function, one can show the weight of any bin in the First Fit packing (except possibly a constant number of bins) is at least 1 (Step II). Meanwhile, the weight of any bin in the packing of \(\text{OPT}\) is at least 1.7 (Step III). We conclude that the competitive ratio of First Fit is at most 1.7.

Using a similar weighting function, one can prove that any Almost-Any-Fit algorithm has a competitive ratio at most 1.7. This implies that Best Fit is also having competitive ratio of 1.7.
2.4 General Lower Bounds

We saw in the previous section that competitive ratios of First Fit and Best Fit are 1.7 while that of Harmonic is roughly 1.69. This raises the question that how good the competitive ratio can be? Is it possible to achieve competitive ratios close to 1? The answer is No:

**Theorem 4.** The competitive ratio of any online bin packing algorithm is at least 4/3.

**Proof.** Consider the following input sequence: 

\[
\sigma = (1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon).
\]

Consider the sub-sequence \(\sigma_1\) formed by the first \(m\) items. The cost of \(\text{OPT}\) for this sub-sequence is \(m/2\) (it places two items of size \(1/2 - \epsilon\) in one bin). Let the cost of \(\text{Alg}\) be \(\alpha m\) for some \(\alpha\) so that \(1/2 \leq \alpha \leq 1\). So the ratio between the cost of algorithm and \(\text{OPT}\) will be \(\frac{\alpha m}{m/2} = 2\alpha\). Since there is one sequence for which the ratio between the cost of \(\text{Alg}\) and \(\text{OPT}\) is 2\(\alpha\), the competitive ratio of \(\text{Alg}\) is at least 2\(\alpha\).

Next, consider the whole sequence \(\sigma\). The cost of \(\text{OPT}\) for \(\sigma\) is \(m\) (it places one item of size \(1/2 - \epsilon\) together with an item of size \(1/2 + \epsilon\) in the same bin. Recall that \(\text{Alg}\) has opened \(\alpha m\) bins for the first \(m\) items (for \(\sigma_1\)). Out of these \(\alpha m\) bins, \(m - \alpha m\) bins have two items (since there are \(m\) items in the \(\sigma_1\) and at most two fit in the same bin). So, among the \(\alpha m\) bins opened for \(\sigma_1\), \(\alpha m - (m - \alpha m) = 2\alpha m - m\) bins have one item. These bins can accommodate an item of size \(1/2 + \epsilon\).

From the \(m\) items in the second half of \(\sigma\), at most \(2\alpha - m\) of them can be placed in the bins opened for \(\sigma_1\). This means that \(\text{Alg}\) has to open \(m - (2\alpha m - m) = 2m - 2\alpha m\) new bins for the rest of items in the second half of \(\sigma\). Hence, the total number of bins in packing of \(\text{Alg}\) for \(\sigma\) is at least \(\alpha m + 2m - 2\alpha m = 2m - \alpha m\), and the ratio between the cost of \(\text{Alg}\) and \(\text{OPT}\) for \(\sigma\) is at least \(\frac{\alpha m}{2m - \alpha m} = 2 - \alpha\).

In summary, the ratio between the costs of \(\text{Alg}\) and \(\text{OPT}\) for \(\sigma_1\) is 2\(\alpha\) and for \(\sigma\) is at least \(2 - \alpha\). Hence, the competitive ratio of \(\text{Alg}\) is at least \(\max\{2\alpha, 2 - \alpha\}\) which is at least 4/3. \(\square\)

The above lower bound is a relatively ‘easy’ lower bound. More complicate sequences results in better lower bounds. For example, using a similar approach as the above lemma but with a more complicated case analysis, we can improve the lower bound from 4/3 to 1.5 using the following sequence:

\[
\sigma_2 = (1/6 - \epsilon, ... , 1/6 - \epsilon, 1/3 - \epsilon, ... , 1/3 - \epsilon, 1/2 + 2\epsilon, ... , 1/2 + 2\epsilon) \rightarrow
\]

Note that in these lower bounds, we do not use an adversarial sequence. This means that, unlike what was the case for problems like paging and list update in which an adversary generates a bad sequence based on the actions of the algorithm, in the case of bin packing, the worst-case sequence is fixed among all sequences. As a consequence, unlike list update and paging where a randomized strategy could ‘limit’ the adversary in generating worst-case sequences, in bin packing, randomization cannot help to break the lower bounds. Hence, the above lower bounds hold for all bin packing algorithms, including randomized algorithms.

There are generally two restrictions for online algorithms which can be exploited to achieve lower bounds for their competitive ratios. One is the online constraint meaning that the algorithm is oblivious to the forthcoming requests/items and second is incremental constraint meaning that the algorithm should build its solution on top of the previously built solutions for prefix subsequences of the input. In the case of bin packing, it is the incremental constraint which is used in the lower bound. This means that, even if the algorithm knows the input (sequence \(\sigma\) in the proof of Theorem 4), it cannot achieve a competitive ratio better than 4/3 because of the incremental constraints. All
lower bounds for bin packing exploit the incremental constraint. This includes the following theorem which gives the best existing lower bound.

**Theorem 5.** No online bin packing algorithm can have a competitive ratio better than 1.54037.